Fall 10-3-2019

Automatic inference of causal reasoning chains from student essays

Simon Mark Hughes
DePaul University, simonhughes22@hotmail.com

AUTOMATIC INFERENCE OF CAUSAL REASONING CHAINS FROM STUDENT ESSAYS

BY

SIMON HUGHES

A DISSERTATION SUBMITTED TO THE SCHOOL OF COMPUTING,

COLLEGE OF COMPUTING AND DIGITAL MEDIA OF DEPAUL

UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPAUL UNIVERSITY

CHICAGO, ILLINOIS

2019

DePaul University
College of Computing and Digital Media

Dissertation Verification

This doctoral dissertation has been read and approved by the dissertation committee below according to the requirements of the Computer and Information Systems PhD program and DePaul University.

Name: Simon Hughes

Title of dissertation:

AUTOMATIC INFERENCE OF CAUSAL REASONING CHAINS FROM STUDENT ESSAYS

Date of Dissertation Defense:

October 3rd, 2019

Peter Hastings
Dissertation Advisor*

Noriko Tomuro
1st Reader

Jonathan Gemmell
2nd Reader

Barbara Di Eugenio
3rd Reader

Daniela Raicu
4th Reader (if applicable)

5th Reader (if applicable)

** A copy of this form has been signed, but it may only be viewed after submission and approval of FERPA request letter.*

AUTOMATIC INFERENCE OF CAUSAL REASONING CHAINS FROM STUDENT ESSAYS

Abstract

While there has been an increasing focus on higher-level thinking skills arising from the Common Core Standards, many high-school and middle-school students struggle to combine and integrate information from multiple sources when writing essays. Writing is an important learning skill, and there is increasing evidence that writing about a topic develops a deeper understanding in the student. However, grading essays is time consuming for teachers, resulting in an increasing focus on shallower forms of assessment that are easier to automate, such as multiple-choice tests. Existing essay grading software has attempted to ease this burden but relies on shallow lexico-syntactic features and is unable to understand the structure or validity of a student's arguments or explanations. Without the ability to understand a student's reasoning processes, it is impossible to write automated formative assessment systems to assist students with improving their thinking skills through essay writing.

In order to understand the arguments put forth in an explanatory essay in the science domain, we need a method of representing the causal structure of a piece of explanatory text. Psychologists use a representation called a *causal model* to represent a student's understanding of an explanatory text. This consists of a number of core concepts, and a set of causal relations linking them into one or more causal chains, forming a causal model. In this thesis

I present a novel system for automatically constructing causal models from student scientific essays using Natural Language Processing (NLP) techniques.

The problem was decomposed into 4 sub-problems - assigning essay concepts to words, detecting causal-relations between these concepts, resolving coreferences within each essay, and using the structure of the whole essay to reconstruct a causal model. Solutions to each of these sub-problems build upon the predictions from the solutions to earlier problems, forming a sequential pipeline of models. Designing a system in this way allows later models to correct for false positive predictions from downstream models. However, this also has the disadvantage that errors made in earlier models can propagate through the system, negatively impacting the upstream models, and limiting their accuracy. Producing robust solutions for the initial 2 sub problems, detecting concepts, and parsing causal relations between them, was critical in building a robust system.

A number of sequence labeling models were trained to classify the concepts associated with each word, with the most effective approach being a bidirectional recurrent neural network (RNN), a deep learning model commonly applied to word labeling problems. This is because the RNN used pre-trained word embeddings to better generalize to rarer words, and was able to use information from both ends of each sentence to infer a word's concept. The concepts predicted by this model were then used to develop causal relation parsing models for detecting causal connections between these concepts. A shift-reduce dependency parsing model was trained using the SEARN algorithm and out-performed a number of other approaches by better utilizing the structure of the problem and directly optimizing the error metric used.

Two pre-trained coreference resolution systems were used to resolve coreferences within the essays. However a word tagging model trained to predict anaphors combined with a heuristic for determining the antecedent out-performed these two systems. Finally, a model was developed for parsing a causal model from an entire essay, utilizing the solutions to the three previous problems. A beam search algorithm was used to produce multiple parses for each sentence, which in turn were combined to generate multiple candidate causal models for each student essay. A reranking algorithm was then used to select the optimal causal model from all of the generated candidates.

An important contribution of this work is that it represents a system for parsing a complete causal model of a scientific essay from a student's written answer. Existing systems have been developed to parse individual causal relations, but no existing system attempts to parse a sequence of linked causal relations forming a causal model from an explanatory scientific essay. It is hoped that this work can lead to the development of more robust essay grading software and formative assessment tools, and can be extended to build solutions for extracting causality from text in other domains. In addition, I also present 2 novel approaches for optimizing the micro-F_1 score within the design of two of the algorithms studied - the dependency parser discussed in Chapter 5 and the reranking algorithm discussed in Chapter 7. The dependency parser uses a custom cost function to estimate the impact of parsing mistakes on the overall micro-F_1 score, while the reranking algorithm allows the micro-F_1 score to be optimized by tuning the beam search parameter to balance recall and precision.

Acknowledgements

The research reported here was supported by the Institute of Education Sciences, U.S. Department of Education, through Grant R305F100007 to University of Illinois at Chicago. The opinions expressed are those of the author and do not represent views of the Institute or the U.S. Department of Education.

Contents

List of Tables

List of Figures

List of Equations

List of Algorithms

Chapter 1

Introduction

1.1 Overview

In the US, educational standards have improved as a result of the Common Core Standards and also the Next Generation science standards [107, 161]. These standards push for the development and assessment of higher-level thinking and reasoning skills, including an understanding of scientific reasoning, evaluating scientific theories, explanations and evidence, and understanding material from multiple documents in different formats. However, many middle school, high school and even college students struggle to combine information from multiple documents in science and history [15, 243].

Writing is considered to be one of the key 21st century skills and has become an important part of many national standards. The U.S. Common Core State Standards [107, 161] now require students to improve their writing skills by developing the ability to synthesize and summarize text, formulate arguments and link ideas to texts and argumentation [68]. Writing is also an important learning skill as it can help develop higher-order thinking skills such as communication and logical reasoning skills in addition to the core literacy skills of reading and writing. Writing has been shown to promote a higher level of comprehension of information from multiple sources [14, 245] because it provides opportunities to transform the source text and synthesize and integrate information [98, 244] across sources.

By forcing the writer to process information more deeply, it is also thought that writing promotes the construction of a 'situation model', a mental model of the topic which leads to better understanding than more superficial tasks [120]. To develop better writing skills it is important for students to get both practice at writing essays and receive personalized feedback [116].

However, due to increasing class sizes and the time taken to effectively grade written assignments, the dominant form of assessment in US classrooms and colleges has become the multiple choice test, in place of essay writing [239]. This is due to the ease at which the multiple choice test can be constructed and automatically graded.

To help address these problems with essay grading and make it a more practical form of assessment, considerable work has been conducted into creating automated essay assessment software (or AES, e.g. [128, 204]). Proponents of AES software claim that AES systems can perform on a par with, or in some cases better than human graders when evaluated in terms of inter-rater reliability [163, 203, 204] and level of agreement with human graders [185]. However, most of this work has been focused on traditional essay grading - summative assessments where each essay receives a single grade only. An alternative form of assessment is formative assessment - rather than providing a student with a single score they are provided with feedback throughout the essay writing process. This feedback highlights areas of potential improvement to help the students to become more proficient writers, and typically covers many different aspects of writing providing a more holistic assessment rather than a single score for the entire essay.

While time spent practicing reading and writing is strongly predictive of ability, receiving timely feedback is particularly critical to the quality of improvement in writing skills [10, 127]. Research into providing automated feedback during essay writing has primarily focused on extending traditional AES systems to provide automated feedback to students as they construct the essays rather than designing new approaches specifically to solve this problem. Existing AES systems rely heavily on surface level features of an essay - such as the word count, syntax, quality of the grammar, and lexical proficiency (variety and rarity of words used) [185, 48]. While these features have proven effective at predicting an essay's grade by measuring the student's overall writing competency, they are insufficient to effectively judge the relevancy and appropriateness of the essay response [164, 56, 247]. A similar criticism of AES systems is on the grounds of 'construct validity', specifically the algorithms fail to determine the writer's level of understanding and knowledge of the subject matter [185, 40], nor the strength and validity of their arguments [48]. As a result, the level of feedback provided by these systems is restricted to 'writing mechanics' - matters of form, structure and style of essay writing. It is not surprising then that studies have shown that providing feedback from these systems results primarily in improvements in writing mechanics but not overall quality when measured in terms of subject matter understanding, and validity of arguments [205].

To provide personalized feedback to improve higher-level writing and reasoning skills, it is necessary to understand the arguments put forth in an essay, and evaluate them for validity and completeness with respect to the essay question and source text. This requires a departure from the methods used in traditional AES systems in order to provide more effective writing feedback. While some work has been conducted into providing some forms of automated formative assessment (e.g. [68]), little work has focused on determining the level and quality of causal reasoning presented in an essay. Natural Language Processing (NLP) is a sub-field of computer science and artificial intelligence (AI), that studies algorithms for understanding and modeling natural language data. This work focuses on building an application to identify causal relations in explanatory scientific essays. Specifically, this research investigates NLP and machine learning methods for automating the construction of 'causal models' from written essays, as explained in the next section. Creating a solution to this problem would not only have many potential applications in education, but could also be adapted to extract causality from more general scientific texts.

1.2 Causal Models

In order to understand the causal reasoning in an explanatory essay, we require a method of representing the causal structure of an essay. Psychologists construct mental models which attempt to describe the thought processes and mental algorithms underlying certain tasks to better understand how humans perform them. In their 1983 book 'Strategies of Discourse Comprehension' [230], Van Dijk and Kintsch described a form of mental model of text comprehension called a 'Situation Model'. A form of situation model that specifically describes causal reasoning is referred to as a 'Causal Model'. This defines a set of core concepts for a particular domain along with causal inferences between these concepts representing cause-effect relations. These inferences can be connected to form causal chains which represent a logical sequence of inferences in an explanatory essay. Causal models can therefore be used to represent the causal reasoning present in an essay in a human and machine readable format. Given a set of essays annotated according to such a model, a machine learning system can then be created to recreate these mental models automatically from the essay texts. Causal models and the essay annotation scheme will be discussed in more detail in chapter 2.

1.3 Problem Statement and Research Questions

Science education has focused on explaining causality for some time [242, 26, for example]. However, automatically detecting causality in scientific essays has received little attention in the literature. Research into causality detection has instead focused on different kinds of texts [33, for instance]. Presumably due to the difficulty of the task, most work in this area has attempted to address a simpler problem - detecting the presence or absence of causal relations only between certain sub-types of causal relation, such as between two noun-phrases [11], or between nouns and verbs only [79]. To overcome these limitations in the types of causal relation that can be detected, this research has focused on developing techniques to automatically infer complete causal models for scientific essays directly from essay texts. As previously described, a 'causal model' consists of a series of concepts linked together through causal relations. To automatically construct a causal model of an essay, a system would need to first identify the core essay concepts referenced in the essay, and then determine the causal relations between these concepts. To accurately identify concepts and causal relations between concepts, the system would also need to resolve anaphoric references within the essay text. Finally, the system would need to analyze the essay holistically to determine how all the concepts are connected to one another via causal relations in order to construct a causal model of the entire essay. Consequently, this problem has been decomposed into four simpler sub-problems:

1. How do we automatically label words with their associated essay concepts from a pre-defined causal model?

2. How do we determine the causal links between essay concepts?

3. Can we improve the accuracy of the word-tagging and causal relation extraction models by first resolving coreferences within the text?

4. Can we utilize the overall structure of the essay to improve the system's accuracy?

For each of these problems, the current literature was used to determine the best approach or approaches that were most likely to solve the problem, and re-phrase each problem as a specific research question in order to incorporate the different approaches. These will be described in the following sections.

1.3.1 Research Question 1

Problem 1 states:

> "How do we automatically label words with their associated essay concepts from a pre-defined causal model?"

This is a word labeling or word-tagging problem - for each word in a sentence, assign zero, one or more concepts from the causal model to that word. This is related to a number of sequence labeling problems which are common in the field of NLP, for example part-of-speech tagging and named-entity recognition. The most commonly used approach for solving a sequence labeling problem is to use a linear chain probabilistic graphical model (discussed in section 3.4.2.1). These are a simple form of probabilistic graphical model that are used to solve sequence labeling problems because they are able to model both the probability of the label given the current and previous words, and also given the previous labels. Two of the most common examples of this type of model include Hidden-Markov Models (HMM) and Conditional Random Fields (CRF), and both have been successfully applied to many NLP sequence labeling tasks, including part-of-speech tagging, named entity recognition, information extraction and text segmentation [129]. In general, the CRF algorithm out-performs the HMM because it does not suffer from the 'label-bias' problem (see section 3.4.2.1). While the strength of these models lies in their ability to model dependencies between consecutive labels, they are trained using the previous labels assigned to the training data, which are always 100% correct. They cannot be trained using their own noisy predictions for the previous labels, and thus fail to accurately model the unreliability of their own predictions.

A simpler approach to solving this problem is to use a window-based tagging model - use a window of words around the word to be labeled as features to train a traditional classifier (see section 3.4.2.3). The advantage of this approach is that any traditional machine learning algorithm can be used to build a tagging model, and the algorithm can take advantage of any contextual features from the words within the window. However, the main disadvantage of this approach is that unlike the HMM and CRF model, it is unable to model dependencies between the consecutive labels in the training data. In addition, it is unable to make use of any information outside of the word window, and thus cannot model longer-range dependencies.

Online tagging models exist that can be trained iteratively to make predictions using both the contextual information from the sentence and their previous label predictions. These iterative machine learning methods have been show empirically to outperform alternative approaches that rely solely on the training data labels [45]. Two examples of popular iterative tagging models are the averaged perceptron, and a recurrent neural network. The averaged perceptron has been shown to out-perform a specific form of linear chain probabilistic model called a Maximum-Entropy Markov Model on part-of-speech tagging and noun-phrase chunking problems [35], and is commonly applied to the more challenging problem of natural language parsing [37, 43] (see section 3.4.5). A recurrent neural network (RNN) is a type of neural network capable of handling sequential prediction tasks by maintaining an internal state that is updated as each new input is processed. This allows it to model longer-range dependencies than is possible with simpler model types, such as a CRF model or an averaged perceptron, and it can be trained in an iterative fashion using its own prior predictions. RNN's have been successfully applied to a wide range of complex sequence labeling tasks [78, 86], for example they have achieved state-of-the-art performance on unsegmented connected handwriting recognition [87], named entity recognition [27, 170], language modeling [146, 248, 170] and speech recognition [85].

Considering these different techniques, the following research question was created to address problem 1:

> "Which of the following model types is the most effective machine learning model to automatically label words with their associated essay concepts from a pre-defined causal model?"
>
> A Window-Based Word Tagging Model
>
> B Conditional Random Field
>
> C Hidden Markov Model
>
> D Averaged Perceptron
>
> E Recurrent Neural Network

For the reasons stated above, I predicted a recurrent neural network would out-perform the other algorithms on the sequence labeling task.

1.3.2 Research Question 2

Problem 2 states:

> " How do we determine the causal links between essay concepts?"

This problem requires determining, for a given sentence, whether or not a causal relation exists between any two concept codes discovered in that sentence. The prior literature on causal relation extraction focuses on the use of lexico-syntactic patterns to detect specific types of causal relations that can occur in open domain text. To solve this problem however, we need to be able to detect any type of causal relation that can occur only between any two concept codes. The solution could not be restricted to specific types or categories of causal relation, and should only care about domain specific relations that are present within the causal model. I therefore needed a more flexible approach that could be trained on data specific to this particular problem.

A simple solution would be to adapt the optimal word tagging problem determined from Research Question 1 to detect causal relations by treating that problem also as a word-tagging problem, where the tags to be predicted are causal relations instead of concept codes. However, the causal relations that occur in the essays can span the length of the entire sentence, and the occurrence of one causal relation can also be influenced by the presence of other concept codes within the sentence that are not part of the causal relation. To make use of this information, a stacking or 'stacked generalization' approach can be applied [246]. Stacking takes the predictions from a number of base-classifiers and feeds them as inputs to a meta classifier which can make predictions in light of the predictions of all of the base-learners (see section 3.3.1.5). To use stacking to solve the causal-relation extraction problem, the optimal word labeling model's predictions about which concepts exist within a sentence were used as inputs to train a stacked model.

The disadvantage of these two approaches is that they treat each causal-relation as a separate label to be predicted. A lot of the observed causal relations are very rare in the training data, making it hard for the model to learn to accurately predict these relations without a much larger amount of positive training examples. A better approach would be to build a system that can treat this problem as a binary classification problem - given a pair of concept codes, is there a causal relation between them? One analogous problem from the domain of NLP is dependency parsing, a form of natural language parsing that detects a series of binary dependencies between words in a sentence. Framing the problem as one of detecting binary dependencies between concept codes, the predicted concept codes could be used to build a dependency parser. One of the most successful methods of building a dependency parser is to use a conditional-history, transition-based based parsing model that parses a sentence by making a sequence of parsing decisions, each decision conditioned on the remaining input and all prior parsing decisions (see section 3.4.3). At the center of any parsing problem is a difficult search problem - given the many different ways in which a sentence can be parsed, determine the most likely parse tree. The SEARN algorithm has been shown to outperform a number of structured learning algorithms, including the averaged perceptron, CRF, SVM$^{\text{STRUCT}}$, and M^{3}N, on a number of NLP structured learning tasks [43, 44] (see section 3.4 for a description of structured learning). SEARN treats a complex problem such as dependency parsing as a sequence of search decisions, and trains a separate machine learning model to make each decision.

One important aspect of detecting causal relations is understanding the context of how each concept code is used in a sentence, and how the author is connecting the two concepts. In order to do this effectively, the model needs to be able to process the whole sentence and learn long distance relationships between words. Recurrent Neural Networks (RNNs) have achieved state-of-the-art results in tasks like sequence labeling and language modeling [146, 248, 170] where learning long term dependencies between words and phrases are important for achieving good results. Furthermore, RNN's are able to take advantage of neural word embeddings which have been pre-trained on a large external corpus. Neural word embeddings capture semantic information about the meaning of each word in the form of a vector representation, allowing the RNN to better generalize on out-of-vocabulary words not present in the training data. See section 3.5 for more information on RNN's and neural word embeddings.

Incorporating these different approaches, Research Question 2 became:

> "What is the most effective approach for determining causal links between concepts from the algorithms below?"
>
> A Extending the optimal tagging model from Research Question 1 to detect causal relations
>
> B Creating a Stacked Model using the predictions from the optimal word tagging model from Research Question 1
>
> C Transition-Based Parsing Model
>
> D Recurrent Neural Network

Due to its strong performance in solving a wide range of complex NLP problems, I believe that using the SEARN algorithm to build a transition-based dependency parser would be the most effective solution for detecting causal-relations.

1.3.3 Research Question 3

Problem 3 states:

> "Can we improve the accuracy of the word-tagging and causal relation extraction models by first resolving coreferences within the text?"

Anaphoric references are common throughout the research data set as the students often refer to essay concepts mentioned earlier in the sentence, or discussed in previous sentences using anaphoric references. The system's performance could therefore be improved by attempting to resolve anaphoric references to detect certain concept codes within an essay. To resolve anaphoric references, two publicly available coreference resolution libraries were used to detect and resolve coreferents - the Berkeley and Stanford coreference resolutions systems. The Berkeley coreference resolution parser is part of the Berkeley Named Entity Resolution system, which demonstrated state-of-the-art performance on the CoNLL 2012 shared task [59] at the time of the paper's publication, attaining an F_1 score of 61.71. The Stanford neural coreference system [31] is the most accurate model from the Stanford CoreNLP library on the CoNLL 2012 shared task at the time of writing, and achieved an F_1 score of 65.73, improving on the Berkeley parser's accuracy on that dataset. Because the performance of any coreference resolution system depends largely on the dataset evaluated, both systems were evaluated on our data.

The accuracy of the word tagging and causal-relation extraction models could be evaluated when using each of these coreference resolution systems, resulting in the following Research Question:

> "Does the accuracy of the word-tagging and causal relation extraction models improve when coreferents are resolved using either of the following two state-of-the-art coreference resolution systems?"
>
> A The Stanford Coreference Resolution System
>
> B The Berkeley Coreference Resolution System

Given the relative frequency of anaphors in the essay data, I predicted that using one or both of these coreference parsers would improve the performance of both the word-tagging and causal relation extraction models, provided at least one of these parsers was effective at detecting coreferences within the essay data. I also expected the Stanford parser to produce a greater improvement than the Berkeley parser due to its stronger performance on the CoNLL-2012 data.

The Stanford Coreference Resolution System is available to download from [90], while the Berkeley Named Entity Resolution System can be downloaded from [89].

1.3.4 Research Question 4

Problem 4 states:

> "Can we utilize the overall structure of the essay to improve the system's accuracy?"

The first two research questions look at determining the concept codes associated with the words in a sentence, and the causal relations occurring within a sentence, and do not use information outside of the sentence when making classification decisions. While Research Question 3 looks at anaphoric references occurring within and between sentences, it does not utilize the overall structure of the essay. The location of a particular sentence within the essay and also within the current paragraph, as well as the concept codes and causal relations that are discussed in the other sentences of the essay likely influence the codes and causal relations that exist in that sentence. This problem subsequently asks whether we can more accurately create a causal model by making use of this additional information.

To solve this problem, it could again be framed as a problem of learning to make a sequence of decisions, and similarly apply a conditional-history, transition-based based parsing model to solve this problem. However, unlike in Research Question 2 where we limited the parsing history to that of the current sentence, for this problem the conditional history could be extended to encompass all previous and subsequent sentences in the essay. The optimal model for research question 2 was extended to solve this problem by including additional structural features from the essay, as well as a larger conditional history that spans the rest of the essay.

While transition-based parsing models are one of the most successful and widely used types of natural language parsing model, another very successful approach is to use a reranking model. First a generative model is used to generate a number of likely parses for a sentence, or in this case a set of likely causal models for an essay. Then a second model is used to rerank the candidates according to their likelihood given the input sentence, or in this case the essay. This can be a very powerful approach, because the reranking model can use a number of new features computed from the generated parse trees, in addition to the original features extracted from the essay text, to rerank the parses. Unlike the transition-based parser which uses local features to make a sequence of parsing decisions, the reranking approach is a global model, which is able to evaluate the consistency of entire causal models with that of the training data to determine which model is most probable. The main limitation of this approach is that it is only able to consider causal models produced by the generative model. If the most accurate causal model is not produced by the generative model, it can not be considered by the reranking model.

Incorporating these two different techniques, Research Question 4 becomes:

> Can a more accurate causal model of an entire essay be constructed by using information from the whole essay using either:
>
> A A Transition-Based Parsing Model to parse the entire essay
>
> B A Reranking Model to rerank possible causal models

Because a reranking model is able to evaluate each generated causal model as a whole rather than constructing it iteratively, I predicted this would be the more successful approach at solving this problem.

1.4 Contributions of this Work

This research presents the first system to extract causal relations from essays according to a pre-defined causal model of the essay topic. Furthermore, this work aims to further advance the techniques of causal relation detection from detecting the presence or absence of individual causal relations to automatically inferring chains of causal reasoning, and building complete causal models from a text document. To the best of my knowledge, this is the first piece of work to attempt to tackle these two problems, and also the first study investigating approaches to detecting causality in explanatory scientific essays.

In recent years, advances in deep learning techniques have produced big improvements in the areas of speech recognition, image classification, and sentiment analysis, as well as numerous other applications [51, 123, 206]. However, little work has attempted to apply deep learning techniques to identify complete causal chains or causal models in scientific essay text. This research compares the efficacy of deep Recurrent Neural Networks for solving these problems with that of a number of more traditional machine learning techniques.

1.5 Potential Applications

The final system outputs the set of concepts found in the essay, identifying where in the original essay text these concepts where located. In combination with the causal model, this could be used to determine which essay concepts were omitted by the student altogether to bring these concepts to their attention. The system also extracts individual cause-effect pairs mentioned in the essay. This could help identify invalid inferences where the cause and effect are not connected in the causal model. It could also help identify incomplete explanations, where the student has omitted intervening concepts when specifying a cause and effect relation. The system output could also be analyzed to determine the order in which the causes and effects are introduced within the essay, which could be used to evaluate whether or not the explanations in the essay were presented in a logical order, and if not it could be used to suggest a clearer and more structured explanation. Finally, the system could be used to present example sentences and sections of essays from other students that more completely answer parts of the essay question.

This analysis could be used in a wide range of teaching applications, such as providing immediate feedback to the learner whilst writing an essay, or to improve upon existing systems for providing formative assessments. Furthermore, it could help address the issue of construct validity in AES systems, and could be used to assist manual essay grading by helping a teacher more rapidly identify which essays have the most problems and may need to be manually graded.

There are also additional potential applications for this research outside of the educational domain. There are many domains where causal models already exist alongside a wealth of unstructured textual data on the domain, for example determining the mechanism of action of a new drug from a scientific paper, or identifying the key arguments in a legal case from a legal document. Given a causal model of the domain and sufficient annotated text documents, this system could be adapted to generate causal models from text for a new domain.

This system could also advance the field of Natural Language Processing (NLP). Traditional question answering (QA) systems are mostly restricted to answering simple questions, as they are unable to follow chains of reasoning. If a QA system is able to build a causal model of a piece of text, it could answer more complex questions by analyzing a causal model of that text. Dialog systems, where a user interacts with a system using a natural language interface, could also benefit from this work as they could build a causal model of the user's problem by combining information across the entire dialog it has had with the user. This could also guide the dialog agent to ask more appropriate questions by analyzing the causal model to look for gaps in its current knowledge of the problem. Finally, this system could be used to enhance the field of information extraction by extracting more complex causal relations from text documents.

The remainder of this thesis will be laid out as follows. Chapter 2 will describe in more detail the causal model, along with the process by which the essays were collected and annotated. Chapter 3 will describe related research in the areas of text classification, sequence learning and causal relation detection that are relevant for this topic, including an array of machine learning techniques applicable to this task. Chapters 4 to 7 will then describe the research conducted to solve research questions 1 to 4.

Chapter 2

From Student Essays to Causal Models - The Dataset

Situation models are an important form of mental model principally used in psychology for modeling how people acquire knowledge and understanding from reading. In this chapter, I will describe a specific form of situation model, a ‘causal model’, that focuses solely on the causal inferences present in a body of text. I will describe how this form of model is identified from an essay by an annotator and then used to evaluate the quality and completeness of the inferences demonstrated by the student through their writing. Automatic extraction of a causal model from student essays will form the focus of this research.

2.1 Situation Models

The term ‘situation model’ was first defined by Van Dijk and Kintsch in 1983 [230] to attempt to create an abstract model of the thought processes underlying text comprehension. To develop an understanding of a piece of text, it is thought that the reader accumulates knowledge of the topic as they read each sentence, combining new knowledge with that acquired from previous sections of the same material [234]. Following the initial paper in 1983, a number of different situation models have been described, including the ‘construction-integration model’ [119], the ‘landscape model’ [229], and the ‘resonance model’ [77]. These models have shown that text comprehension cannot be the sole product of inferences made within the text. Instead the reader has to also build upon their own prior knowledge to construct new factual information from the material that is relevant to their individual experiences and knowledge of the world [234].

A situation model is defined as consisting of 5 different dimensions of information within the text to which readers attend when forming a mental model of the text: spatial information, temporal information, causality, intentionality and finally information about the main protagonist(s) [259]. According to the theory, when information from one of these 5 dimensions is extracted from the text, the user updates their current situation model to integrate this new information. Studies into 'priming effects' lend compelling evidence in support of this theory of text comprehension. Experiments have shown that after reading a passage of text, information related to the text that falls into one or more of these 5 different dimensions is more readily available to the reader (it is 'primed') compared to other types of information [259]. This has been measured using various experimental techniques, including the time taken to make certain verbal utterances or recognize words that were primed following reading, compared to words that were not primed. In this research, I will focus on a different form of situation model that focuses solely on the causal dimension, called a 'causal model'.

2.2 Causal Models

A causal model is a type of situation model that focuses on modeling the causal inferences present in some written material, for instance, a causal model of some scientific phenomena represents the relevant underlying scientific processes at work, and the different cause-effect relations that occur between them. Such a model shows how various initiating factors lead, via a number of intervening concepts, to the final outcome.

Two examples of a causal model are shown in Figures 2.1 and 2.2 below. Figure 2.1 shows a causal model for the causes of coral bleaching (a whitening of coral reefs as they die). In this model, there are two different initiating factors - Decrease in Trade Winds and Storms / Rainfall. These lead via two different causal chains and various intervening factors to the final outcome Coral Bleaching. Figure 2.2 shows a causal model describing the different factors that lead to an increased risk of getting skin cancer. In the Figure, the two initiating factors - Closer to the Equator, and - Increased Melanin initiate two separate causal chains, both resulting in a single final outcome - Increase in Skin Cancer Risk via a number of different intervening factors, such as - Increase in Direct Sunlight and - Decrease in Protection.

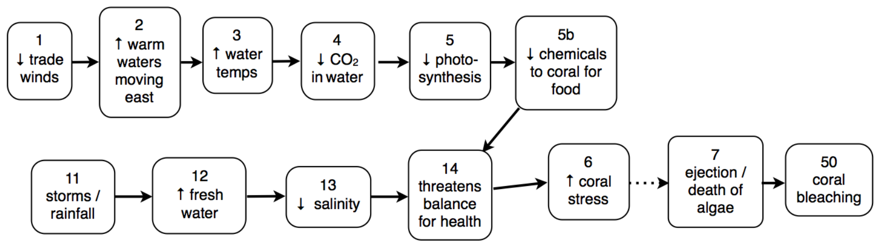


Figure 2.1: Causal model for Coral Bleaching

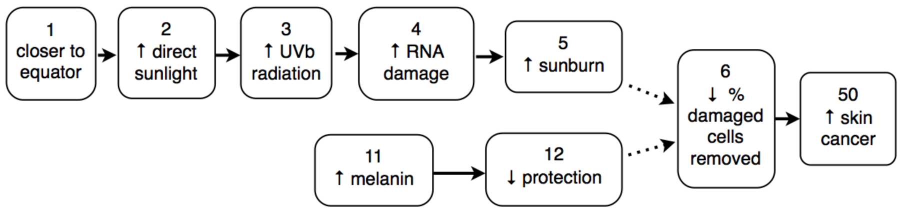


Figure 2.2: Causal model for Skin Cancer

Both of these models include one or more causal links not explicitly mentioned in the source text. These are represented by the dotted lines linking two concepts in the Figures, for example the causal relation between concepts 6 and 7 in Figure 2.1. The students were expected to infer these causal links based on their overall understanding of the source material.

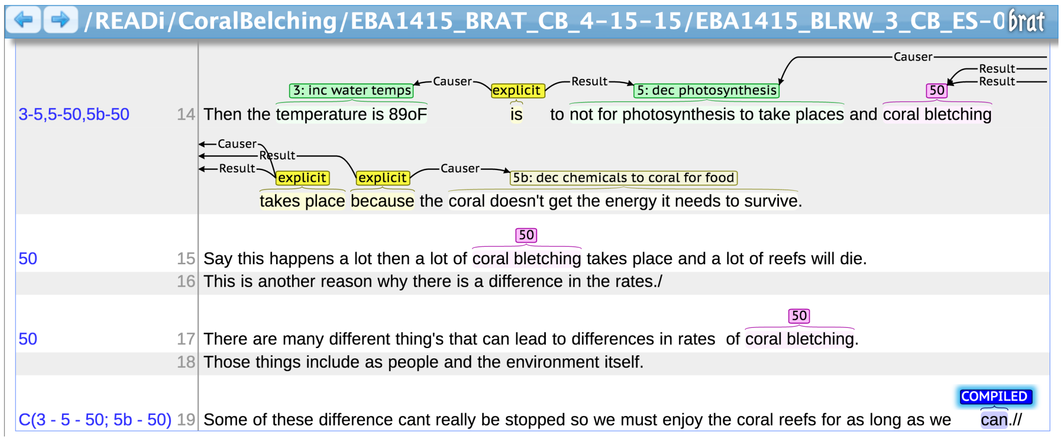
2.3 The Construction of Causal Models for Explanatory Essays

In conjunction with the READI grant [70] (funded by the National Center for Education Research), causal models were created to evaluate how effectively students are able to synthesize and combine information from different source texts. Three different domains were studied as part of this research - science, history and english literature, and the READI grant was created with the goal of improving adolescents' skills at reading multiple texts to produce discipline specific explanations in the case of literature, or causal explanations in the case of science and history. The two causal models described above were created as part of this research. This research project will focus solely on the aspects of this research related to improving causal explanations of scientific phenomena, and will use the document sets collected during the 2014-15 school year in the science domain only.

Middle school and high school science students were asked to read a set of documents, and asked to explain the principal underlying causes of some scientific phenomena by integrating information across several different sources. The writing prompts for each essay topic can be seen in appendix A. A causal model was first constructed for each essay topic that described the principal causal chains underlying the phenomena, and then used to construct a set of source documents (see appendix B for examples of the source documents). The source documents were created from reputable sources (e.g. the US Geological Survey, the NASA earth laboratory, and online science textbooks) and were constructed to facilitate the evaluation task; no individual source could be used on its own to explain the scientific phenomena. This forced the students to combine information from the different documents when constructing the causal explanations. Each document was written at a level appropriate for the target audience, and separate documents were presented on separate pages without staples or numbering so that the students could more easily compare documents from different sources.

Initially, a pilot study was conducted to help the researchers understand and evaluate all aspects of the experimental design, including the creation of the causal model and accompanying documents sets, as well as the annotation processes and machine learning approaches used to automate the annotation process [95]. Guided by the findings of this pilot study, two subsequent studies were conducted, the first on the topic of coral bleaching, and asked the students to "explain how and why coral bleaching rates vary at different times", and the second on skin cancer, and asked students to "explain how and why rates of skin cancer differ around the globe". See Figures 2.2 and 2.1 for the causal models that were created for these two different topics. Each set of documents began with a short background document (255 and 273 words) to establish the main outcome, and provide the necessary framing, background material and vocabulary to answer the essay question. This background document was followed by 4 additional documents consisting of a number of different document types (images, maps, graphs, and descriptive text). Students were instructed to write their essays using the sources provided, and were given several hints to ensure they understood the goal of the task (e.g. "you have to piece together important information", and "you are the one making connections across sources"). The 2 studies focused on 10th grade high school students only, and the same 1,300 students were used for both studies. Three coders were trained as annotators of the essay data; one coder was trained for each of the two topic, and a third was trained on both topics. Inter-rater reliability was high for each each of the datasets, with a κ values ranging between 0.76 and 0.97 for the different studies. A detailed description of the annotation procedure can be found in appendix C, and examples of some essays from both datasets can be found in appendix D.

In Figure 2.3 below you can see an example of an annotated coral bleaching essay, showing a sequence of three causal relations forming a causal chain. First there is a code 3, the cause - Increase in Water Temperatures leading to the effect, code 5 - Decrease in Photosynthesis. This forms part of a causal chain with code 5 in turn acting as the cause of effect 5b - Decrease in Supply of Chemicals to Coral for Food. Finally, there is a third causal relation, that of code 5b leading to code 50 - Coral Bleaching. This last causal relation overlaps the 5$\rightarrow$5b causal relation, and the effect precedes the cause in the sentence. In each causal relation, the cause and effect codes constitute a sequence of multiple words, and the cause is connected to the effect via an 'explicit' connector concept code. This demonstrates how one sequence of words can have multiple overlapping causal relations, and shows that the cause, explicit and effect concept codes do not always appear in the same order within a sentence.



3-5,5-50,5b-50 14 Then the temperature is 89oF is to not for photosynthesis to take places and coral bletching takes place because the coral doesn't get the energy it needs to survive.

50 15 Say this happens a lot then a lot of coral bletching takes place and a lot of reefs will die.

16 This is another reason why there is a difference in the rates./

50 17 There are many different thing's that can lead to differences in rates of coral bletching.

18 Those things include as people and the environment itself.

C(3 - 5 - 50; 5b - 50) 19 Some of these difference cant really be stopped so we must enjoy the coral reefs for as long as we can.//

Figure 2.3: An Annotated Coral Bleaching Essay

Overall, the students were successful at identifying some of the core essay concepts, but struggled to integrate information between sources. For example, 74.3% of all essays contained one or more concept codes from the causal model, and 58.6% of the essays contained some causal chain. However, only 30.9% of the essays contained a causal chain with one or more intervening factors between a cause and the final outcome. Consequently, most of the causal chains observed in the essays linked a cause directly to the final outcome. For example, in a coral bleaching essay a student may state that an Increase in Coral Stress causes Coral Bleaching, omitting the intervening factor Ejection/Death of Algae (see Figure 2.1). One possible explanation for the small number of essays with longer causal chains is that creating such a chain required the students to combine information from multiple sources, indicating this was the most challenging form of causal relation for them to identify from the source text.

2.4 Analysis of the Dataset

The number of essays, sentences, words, and the vocabulary sizes (unique words) are listed in Table 2.1 below. Note that the vocabulary size was computed after the application of a simple spell checker.

Table 2.1: Dataset Sizes and Vocabulary

Dataset	**Essays**	**Sentences**	**No. Words**	**Vocabulary**
Coral Bleaching	1,127	10,198	167,656	4,770
Skin Cancer	1,088	10,670	180,899	4,702

The essays from each dataset vary greatly in length, with the skin cancer essays being slightly longer and using more unique terms (see Table 2.2 and Table 2.3). While the skin cancer essays have more unique words per essay, on average, there are slightly fewer unique words in that corpus: 4,702 words, compared with 4,770 unique words for the coral bleaching corpus.

Table 2.2: Coral Bleaching Essay Statistics

	Min	**Max**	**Mean**	**Median**	**Std. Deviation**
Word Length	1	461	148.8	142.0	81.7
Unique Words	1	226	83.9	83.0	37.8
Sentence Length	1	31	9.0	8.0	5.2

Table 2.3: Skin Cancer Essay Statistics

	Min	**Max**	**Mean**	**Median**	**Std. Deviation**
Word Length	4	479	166.3	157	82.4
Unique Words	4	215	90.2	88	36.2
Sentence Length	1	36	9.8	9	5.0

2.4.1 Concept Code Analysis

There are 13 unique concept codes in the coral bleaching essays, and 9 in the skin cancer essays and there is a small amount of overlap between the essay concepts assigned to each word in both datasets. For the coral bleaching essays, out of 167,656 words, there are 52,014 words assigned concept codes (31%), 12 words of which have multiple concept codes associated with them (0.007%). For the skin cancer dataset, out of 180,899 words, 43,024 words were assigned concept codes (24%), and just 2 words have multiple concept codes (0.001%). At the sentence level, the proportion of sentences with multiple codes is much higher. In the coral bleaching dataset, 7,085 out of 10,198 sentences (69%) have one or more concept codes, and 3,132 sentences (31%) have multiple codes. For the skin cancer dataset, 6,671 sentences out of 10,670 (63%) have concept codes and 4,555 (43%) have multiple concept codes. Comparing both datasets, a greater percentage of words and sentences in the coral bleaching dataset were assigned concept codes than in the skin cancer dataset. The length of each concept code reference in the essays was shorter for the skin cancer dataset, with an average of 3.1 words per code, compared with 4.0 words per code for the coral bleaching dataset. These data are summarized in Table 2.4 below.

Table 2.4: Concept Code Statistics

	Coral Bleaching	**Skin Cancer**
Number of Concept Codes	13	9
MEAN Number of Words Per Concept Code	4.00	3.10
% Words with Concept Codes	31.03	23.78
% Words with Multiple Concept Codes	0.0072	0.0011
% Sentences with Concept Codes	69.48	62.52
% Sentences with Multiple Concept Codes	30.71	42.69

To determine if there are any statistical dependencies between the concept codes occurring in a sentence, we can calculate the pointwise mutual information (PMI) between pairs of concept codes as they occur within sentences in the dataset. PMI is computed as follows:

$$pmi(x;y) = \log \frac{p(x,y)}{p(x) \cdot p(y)} \tag{2.1}$$

PMI divides the joint probability of two events occurring - $p(x,y)$ by the probability of these 2 events occurring by chance - $p(x) \cdot p(y)$, and then takes the natural log of this number. PMI values greater than zero denote events that are more likely to occur together than by chance alone, whereas negative PMI values indicate two events that are unlikely to occur together (they occur together less frequently that would be expected by random coincidence). Given the structure of the causal model, it seems likely that if 2 concept codes exist in a sentence, that they are adjacent codes in the causal model, i.e. there is a direct causal relation between them. If we examine the PMI values for adjacent concept codes in each model, we see that there are positive PMI values for all but one pair of codes in each model (see table 2.5 and table 2.6 below):

Table 2.5: Coral Bleaching Concept Code Dependencies

Concept Code	Adjacent Code	PMI
1	2	2.23
2	3	1.53
3	4	1.73
4	5	2.11
5	5b	2.11
5b	14	-0.69
6	7	2.24
7	50	1.20
11	12	3.93
12	13	3.58
13	14	2.59
14	6	2.34

Table 2.6: Skin Cancer Concept Code Dependencies

Concept Code	Adjacent Code	PMI
1	2	1.83
2	3	1.19
3	4	1.54
4	5	1.39
5	6	1.69
6	50	1.10
11	12	2.82
12	6	-1.50

From this we can conclude there is quite a strong dependency between the adjacent concept codes, indicating that the ordering of concept codes within the essay is not random. In other words, the occurrence of one concept code in a sentence makes the appearance of an adjacent concept code from the model much more likely. It is interesting to note that the only negative PMI values between adjacent codes in each causal model are for one of the pair of codes that occur when the two different paths in each model combine, between codes 5b and 14 for the coral bleaching dataset, and between codes 12 and 6 for the skin cancer dataset. The causal connection between codes 12 and 6 in the skin cancer model is also an implied causal relation; it is not stated explicitly in the source texts, making this a more challenging causal relation for the students to determine.

In terms of the coverage of concept codes, 22% of the 13 coral bleaching concept codes were included on average in each essay, and 36% in the skin cancer dataset. However, not all concept codes occurred at the same frequency in the dataset, some were much more prevalent than others, resulting in an imbalanced dataset as seen in Tables 2.7 and 2.8 below. In this calculation, a sentence was assigned a code if that code occurred one or more times anywhere in the sentence.

Table 2.7: Percentage of Words and Sentences Assigned Each Concept Code in the Coral Bleaching Dataset

Coral Bleaching		
Concept Code	**Percentage of Words**	**Percentage of Sentences**
1	3.33	12.69
2	0.85	2.27
3	5.36	14.61
4	1.88	5.38
5	1.46	6.26
5b	1.44	2.69
6	0.88	4.00
7	3.03	9.17
11	0.63	3.67
12	0.51	1.23
13	1.34	4.92
14	1.47	3.34
50	8.86	39.36

Table 2.8: Percentage of Words and Sentences Assigned Each Concept Code in the Skin Cancer Dataset

Skin Cancer		
Concept Code	**Percentage of Words**	**Percentage of Sentences**
1	3.38	13.46
2	3.33	17.22
3	2.30	12.69
4	1.86	7.71
5	2.82	21.94
6	2.88	9.20
11	0.33	2.98
12	0.52	6.08
50	6.36	29.29

2.4.2 Causal Relation Analysis

Because each causal relation consist of 2 concept codes, there are a lot more unique causal relations than concept codes and fewer sentences contain causal relations than concept codes. There are 86 unique causal relations in the coral bleaching dataset, and 49 in the skin cancer dataset. 35,303 out of 167,656 words (21%) and 2,781 out of 10,198 sentences (27%) of sentence are assigned causal relations in the coral bleaching dataset. In the skin cancer dataset, 53,742 out of 180,899 words (29%) and 4,483 sentences out of 10,670 sentences were assigned causal relations, so a greater percentage of words and sentences were assigned causal relations than in the coral bleaching dataset. This differs from the concept codes where a larger proportion of words and sentences in the coral bleaching dataset had concept codes assigned. 4% of words and 12.5% of sentences had multiple causal relations as opposed to 2.9% and 6.6% for the coral bleaching dataset, so there is a much higher degree of overlap for the skin cancer dataset. There is also a higher degree of overlap for the causal relations compared to the concept codes for both datasets. The average number of words assigned to a single causal relation is shorter for the skin cancer dataset than is observed with the concept codes; 10.0 words compared to 11.17 words for the coral bleaching data. The average number of words constituting a causal relation is over twice the average length of a concept code because each causal relation involves two concept codes and usually one or more words linking the two codes in the relation. These datapoints are summarized in Table 2.9 below.

Table 2.9: Causal Relation Statistics

	Coral Bleaching	**Skin Cancer**
Number of Unique Causal Relations	86	49
MEAN Number of Words Per Causal Relation	11.17	10.00
% Words with Causal Relations	21.06	29.71
% Words with Multiple Causal Relations	2.91	4.03
% Sentences with Causal Relations	27.27	42.02
% Sentences with Multiple Causal Relations	6.63	12.50

In addition, the causal relations show a similar class imbalance to the concept codes, although each causal relation is much rarer. Please refer to Appendix F for the relative frequencies of the causal relations at the word and sentence level.

2.5 Anaphora and Coreference

In linguistics, *coreference* is the phenomenon where two or more statements in text refer to the same entity [139]. *Anaphora* is a type of coreference where a word such as a pronoun refers to an entity that occurs earlier in the same sentence or document, called the *antecedent*. Anaphora resolution presents a challenging problem for machine learning algorithms when working on NLP problems; the algorithm must be able to infer the antecedent, being referred to by the anaphor in order to make correct classification decisions. To understand the relative importance of anaphora in the different datasets, anaphoric references were annotated as part of the annotation procedure. Table 2.10 below shows the relative frequencies of anaphoric references within both datasets.

Table 2.10: Anaphora Statistics

	Coral Bleaching	**Skin Cancer**
Essays with Anaphora tags	21.02%	29.78%
Sentences with Anaphora tags	3.12%	4.10%
Words with Anaphora tags	0.22%	0.32%
Concept Codes with Anaphora Tags	3.25%	3.84%
Causal Relations with Anaphora Tags	6.64%	6.70%
Causal Relations Spanning Sentences	3.83%	5.39%

When writing the essays, a common writing style was to develop an explanation of the causal phenomena over several sentences. Consequently, a number of causal relations described by the students refer to concept codes that are either causes or effects mentioned in previous sentences, but were not marked as explicit anaphoric references. These are shown in Table 2.10 as 'Causal Relations Spanning Sentences'. 3.83% of the coral bleaching causal relations and 5.39% of the skin cancer causal relations were of this form. Thus, any machine learning algorithm designed to automatically infer causal relations from these datasets will be more accurate if it can accurately resolve these different types of coreference.

Coreference is a general linguistic phenomenon, and is not specific to this type of problem or dataset. Therefore. this research can provide insight into how coreference is used when explaining cause and effect relationships, and in constructing a causal chain, and has implications beyond this dataset and this specific set of problems.

Chapter 3

Related Work

The 3 different research questions outlined in Chapter 1 present complex problems, which are not amenable to traditional machine learning regression and classification problems. Research Questions 1 and 2 are types of sequence labelling problems, where the task is to assign a label to each word in a sentence, while Research Question 4 involves predicting a causal model - a complex graph structure. Both of these types of problem are structured learning problems, a type of machine learning problem where the output has a complex structure such as a sequence or a graph structure. Furthermore, the 2 sequence labelling problems also present examples of multi-label classification problems, a special form of classification problem where there can be varying numbers of labels for each data point. In this Chapter, I will discuss the main techniques for solving these kinds of problem. However, I will start by discussing the previous literature on causal relation extraction, and why the techniques described are ill-suited to solving the problems outlined in this research.

3.1 Causal Relation Extraction

Science education has focused on explaining causality for some time [242, 26, for example]. However, automatically detecting causality in scientific essays has received little attention in the literature. Research in this area has instead focused on different kinds of texts. In 1987, Cohen [33] outlined a set of problems that he believed needed addressing to fully understand argumentative discourse. These included defining a set of linguistic clues that define the structure of an argument, the relative importance of a coherent structure in argumentative discourse, and the need for a pragmatic analysis when the participants differ in their beliefs. Thirty years later, a workshop at the

SemEval-2007 conference focused on relation extraction, and included a task for "the classification of semantic relations between nominals", which included the detection of causal relations as a sub-task [80]. The highest F1 score on this category was 0.82 with an accuracy of 77.5% in a system that used features based on WordNet, VerbNet, lexico-syntactic features and dependency-parse features.

The Semeval 2007 causal relation task typifies the prior work in this area, which has focused on detecting specific sub-types of causal relation that only occur between certain specific grammatical constructs, and has relied primarily on the use of surface lexical features and syntactic patterns. Little work to date has focused on detecting arbitrary causal relations in text. This is mainly because arbitrary causal relations are hard to annotate, and systems designed to detect them have historically achieved very poor performance [180]. In 2002, Girju and Moldovan [79] used lexico-syntactic patterns to detect causal relations between two noun phrases of the form <NP1 verb NP2>, where the verb was a simple causative. They achieved an accuracy of 65.6% compared to the average of two human annotators. In 2008, Blanco et al. [11] used lexical, syntactic and semantic features to train a machine learning model to detect causal relations, attaining an impressive F1 score of 0.895. However, as with other studies in this area, they restricted the type of causal relation detected, in this case focusing on explicit and marked causal relations that consisted of a verb phrase, with a relator and a clause. In a very different approach, Rink et al [183] built a system that detected causal relations containing verb events joined with a conjunction. In a novel approach, they created graphical models of sentences that encoded both syntactic and hypernym information, as well as dependencies from a dependency parser. They then extracted sub-graph patterns that occurred in causal sentences and used a constraint satisfaction solver to detect these patterns from new sentences, attaining an F1 score of 0.39 on this type of causal relation.

A number of authors have attempted to combine pattern matching and machine learning approaches to benefit from the advantages of both approaches. To move beyond surface lexical features, in 2014 Riaz and Girju [182], [181] used Integer Linear Programming [188] to combine verb and noun semantics with the predictions of a supervised machine-learning model trained only on linguistic features. They focused on detecting only causal relations between noun and verbs, achieving an F1 score of 0.41 and an accuracy of 80.73%, a 15% improvement over using linguistic features alone. A more common approach is to use the syntactic patterns as training data in a machine learning system. In [212], the authors manually defined lexico-syntactic patterns over words to identify possible sentences involving causal relations. They then trained a Naïve Bayes classifier to determine which sentences contained causal relations, attaining an F1 score of 0.64. However, they focused only on causal relations where the cause and effect were both present, and indicated by specific linguistic units.

Additionally, there has been some work in this area from within the medical domain. In 2000, Khoo et al [117] manually constructed a set of graphical patterns that indicated causal relations, and matched these against syntactic parse trees to extract causal relations, and identified the cause and effect by filling in specific slots in the patterns. They achieved an F1 score for detecting causal relations of 0.681, with F1 scores of 0.497 and 0.512 at detecting the specific causes and effects respectively. Chang and Soi [21] in 2004 designed a system to identify causal relations existing between two events expressed as noun phrases. While other work in this area has focused on extracting causal relations within a sentence, they extended their system to detect inter-sentence causal relations. Using an unsupervised approach for detecting cue-phrases and lexical pairs from the training data in conjunction with a Naive Bayes classifier, they attained a precision of 0.746 on detecting inter-sentence causal relations. In 2006, Giuliano et al used shallow linguistic information to train an SVM classifier to extract gene and protein interactions [81]. They used a combination of two different kernels in the SVM classifier, one that used the whole sentence and a second that used only the local context around the entities involved. Fundel et al [73] also worked on extracting regulatory interactions between genes and proteins, building the RelEx system. This used patterns extracted from dependency parse trees to identify these interactions, achieving an F1 score of 0.8 on extracting these domain specific relations. More recently, in 2010 Volkova et al [236] used syntactic patterns and part-of-speech tagging to identify semantic relations in biomedical text, including causal relations to automatically expand a manually constructed ontology. Similar to other work, only causal relations matching specific syntactic patterns could be identified.

The causal relations present in the student essays are not restricted to types of causal relation that only exist between certain grammatical constructs. Instead, the casual relations can take any grammatical form, are specific to the essay question, and consist of scientific explanations relevant to the essay question. Furthermore, some causal relations span multiple sentences, referencing concept codes described much earlier in the essay, and thus contain long distance dependencies between the annotations present. Detecting these causal relations is not possible when using syntactic patterns that rely on the grammatical structure of a sentence, or when using only simpler lexical features and patterns. To effectively detect these more complex types of causal relation, techniques that can leverage short and long-term dependencies between the concepts detected in an essay were needed. In addition, the final system needed to be able to predict complex graph structures (the causal model of each essay), and could not be restricted to detecting the presence or absence of causal relations in sentences. The next sections will discuss some different approaches that are better designed to address these issues.

3.2 Discourse Analysis

Discourse analysis is a sub-field of NLP and computational linguistics that studies coherent sequences of natural language utterances, such as sentences, propositions, or conversational dialog (written or spoken) where two or more people take it in turns to communicate. Discourse analysis covers both 'local' structures of discourse (sentences, propositions, etc) and so-called 'global' structures such as the overall topics and schematic structure of a conversation or an article of text [113]. The study of discourse analysis and modeling has focused primarily on discourse relations, also referred to as coherence relations and rhetorical relations. Discourse relations are possible relations or connections between utterances in a discourse. One example, from [99], is "John bought an Acura. His father went Ballistic". This is an example of a 'Result' relation, where the first sentence causes or could cause the second. This represents a form of causal discourse relation, although many other types of relation exist, for instance 'Elaboration', where the second utterance elaborates and expands on the information in the first.

3.2.1 Rhetorical Structure Theory

Expanding on this idea, Mann and Thompson developed Rhetorical Structure Theory (RST) in 1988, which proposes a hierarchically structured organization of texts based on a set of discourse relations between text spans [138]. A related idea is the notion of coherence. Coherence in natural language describes how logically and semantically related two utterances are. It is widely believed that coherence in natural language arises through these types of discourse relations between discourse units. Most RST relations denote 2 types of text segment - a central segment called a 'nucleus' and a more peripheral segment called a 'satellite' [113], illustrating the fact that most discourse relations are asymmetrical. For example the text "I love to collect classic automobiles. My favorite car is my 1989 Duryea" is an example of the 'Elaboration' relation. In this example, the first sentence is the nucleus and the second is the satellite, which is interpreted in light of the nucleus [113].

3.2.2 Automated Discourse Parsing

RST and discourse analysis have given rise to the field of discourse parsing, where parsing models are developed to produce 'discourse trees' - a hierarchical representation of a piece of text, structured according to the discourse relations contained within. Automated approaches to discourse parsing have a long history. Soricut and Marcu [213] in 2003 developed the SPADE system that used probabilistic models for sentence-level discourse parsing that utilized lexical and syntactic features derived from a lexicalized syntactic tree parsed from each sentence. SPADE had a number of limitations however; it relied solely on lexical and syntactic features, used a generative rather than a discriminative approach to estimate the model parameters, and assumed independence between the label and the structure while modeling a constituent [111]. Subba and Di Eugenio (2009) [217] developed a shift-reduce parser that utilized an inductive logic programming classifier (ILP) to identify the type of discourse relation. The ILP classifier learns a set of first-order logic rules using features derived from a semantic parser that identifies a number of rich linguistic features, including *compositional semantics* features.

Later, in 2013 Joty et al [112] improve on their earlier discourse parsing model [111] by building a two-phase parsing algorithm using a dynamic CRF model. First, they construct an intra-sentential discourse parser to produce a discourse tree for each sentence. Then a multi-sentential parser combines the sentence-level discourse trees to produce a text-level discourse tree. The strength of this approach lies in the fact that their model jointly models both structure and discourse relations, allowing it to capture dependencies between these two aspects. It achieves this by separately modeling the probability of two adjacent discourse units being connected, and the probability of two connected units belonging to a particular discourse relation. The result is a joint model utilizing both of these types of relationship. A CKY-like parsing algorithm is then used to find the globally optimal discourse tree [67]. Their system achieves an overall accuracy in relation assignment of 55.73%. The main limitation of this approach is the high order of the time complexity - $O(n^3)$, where n is the number of discourse units. This makes using this model impractical for larger documents. To address this problem, Feng and Hirst (2014) develop a more accurate discourse parser that has linear-time complexity [67]. Inspired by Joty et al's model, they also build a two phase parsing model consisting of an intra-sentence parser followed by a multi-sentential parser. Their system uses a greedy bottom-up parser that uses a cascade of two CRF models, one for each parsing task, and is capable of a linear time complexity, in terms of the number of discourse units. In addition they use a novel post-editing approach that modifies the parse tree by considering information from constituents on multiple-levels to further improve their accuracy. Their system achieves an overall accuracy of 58.2% in relation assignment.

3.2.3 Detecting Argumentative Discourse Relations

A lot of previous work on discourse parsing has focused on detecting argumentation. In this context, argumentation is seen as the author making a claim in the text, and then presenting evidence, or the premise, that they believe supports the claim. Although a claim could involve stating a cause and effect relationship, argumentation in this context is not restricted to the identification of causal relations, as they form just one of many different types of argumentative discourse relation. One of the first studies in this area was done by Teufel in 1999 [220]. She classified sentences from scientific articles as one of seven rhetorical roles, including claim, result and purpose, achieving an F1 score of 0.46 using lexical, syntactic and structural features. Later, in 2012, Rooney et al identified claims, premises and non-argumentative text in the Araucaria corpus, and achieved an accuracy of 65% at this task.

Some authors have focused on detecting argumentative discourse structures in student essays. In 2001, Burstein et al [18] improved the accuracy of the e-rater AES software by building a separate module for detecting argumentative discourse relations. The agreement between the argumentation module and the human raters was 82%, 13% higher than the 69% accuracy achieved using a bag-of-words approach that ignored argumentation. In 2014, Stab and Gurevych [216] used a two-step classification approach for identifying argumentative discourse structures in persuasive essays. They trained an initial set of classifiers to first identify argumentation components, such as claims and premises. To then identify the structure of the argumentative discourse, they trained a second classifier to determine whether a pair of argument components were support or non-support (i.e. supported or did not support the claim). They used a variety of different features, including structural, lexical, syntactic and contextual features, and attained an F1 score of 0.726 for identifying argument components and 0.722 for identifying argumentative relations. In both the Stab and Gurevych and the Burstein et al papers, the authors found that the presence of modal verbs such as *would*, *could*, *should* and *might* were useful features in detecting argumentative relations.

Discourse analysis has also been applied to build more effective essay tutoring systems. In 2003, Burstein et al [19] built a discourse analysis system designed to help students improve their essay writing by identifying discourse elements in their essays. Similar to the other related work in this area, they focused on persuasive and informative essays where students were required to state their opinion on a topic and provide arguments in support of that opinion. While other work has primarily focused on argumentative discourse, this system annotated discourse units as introductory material, thesis, main ideas, supporting ideas, conclusion, title, and other. The authors used a discourse parser to parse discourse relations from the RST framework and trained a number of different models, including a decision tree classifier and a probabilistic model. They evaluated their system on essays written on topics that were different from the training data to see how well the system generalized to new domains, and achieved an F1 score of 0.79 on these essays.

3.3 Multi-Label Classification Problems

The simplest form of machine learning classification problem involves predicting whether or not the input belongs to a single class, and this is referred to as a binary classification problem. For more than two classes of labels where a single label is predicted, the problem is referred to a a multi-class classification problem. A more complex variant of this is where there can be a varying number of labels for each example in the dataset, which is called *multi-label classification* (MLC). One example of MLC is generating a set of keywords to use to tag a document with. When a document is uploaded to a website, the site may ask for a list of different keywords describing the core topics and ideas contained within the document to aid search and discovery. For any given document, it could have zero, one or several different keywords describing its contents. Other examples include detecting objects in images, identifying emotional expressions on faces [135], and assigning genres to a movie on the IMDB website [24]. Multi-label classification presents a problem for traditional machine learning algorithms which are usually designed to output a single label for every row, and are unable to learn dependencies between the output labels. The first two research questions described in Chapter 1, labeling words with their associated concepts and detecting causal relations within sentences, both present examples of MLC problems. In this section I will describe the two main types of approach for addressing this sort of problem - problem transformation methods, and algorithm adaptation approaches [25].

3.3.1 Problem Transformation Methods

One approach to solving multi-label classification problems is to transform the problem in such a way that traditional classification algorithms, such as support vector machines and logistic regression, can be applied to solve it. A number of transformations have been proposed to solve this problem, including *binary relevance*, *BR+*, *classifier chains*, the *label powerset method*, and *stacking* [25, 49]. These techniques transform the problem in a number of different ways, from breaking down the MLC problem into a series of binary classification decisions, to training a separate classifier for each unique combination of labels in the training data. Each transformation method has certain advantages and limitations that tend to make it better suited to solving certain types of MLC problem.

3.3.1.1 Binary Relevance

In *binary relevance* (BR), a multi-label classification problem with L labels is broken down into L separate binary classification problems, and a separate binary classifier trained separately for each label. This method has been shown to be effective at multi-label classification by multiple authors [106] and [135]. One limitation of binary relevance is that it assumes independence between labels and cannot make use of dependencies between the output labels [136]. The other approaches to multi-label classification discussed in this section attempt to address this problem by utilizing correlations between the output labels.

3.3.1.2 BR+

BR+ modifies the binary relevance method to take into account dependencies between labels during learning [25]. First, an initial set of classifiers are trained using the BR approach. Then a second set of classifiers are trained, one per class, on the original dataset but with the labels predicted by the first set of classifiers used as additional features, excluding the label to be predicted [3]. When the algorithm is tested on unlabeled data, the dataset is augmented with the predictions from the initial BR classifiers.

3.3.1.3 Classifier Chains

Classifier chains also extend the binary relevancy method to incorporate label dependency information while learning. L binary classifiers are trained, one per class, in a chain where each classifier is trained on a separate copy of the training data containing all the labels predicted by the previous classifiers in the chain. When the classifier chain is trained, the true labels from the training data are used, but for new unlabeled data points, the predictions made by the classifiers earlier in the chain are used to augment the data [179, 3]. One disadvantage of this approach is that each classifier can only take advantage of dependencies between its target class, and the target classes of classifiers earlier in the chain, thus assuming independence between a class and the classes predicted later in the chain [25]. Therefore the order in which the classifier chain is trained is very important, and not all dependencies between labels can be utilized by the algorithm. In order to solve this problem, it is possible to train an ensemble of classifier chains, where each model in the ensemble is trained on a subset of the training data, using a random ordering of classifiers in the chain [184]. However, this increases the computational complexity of this approach.

3.3.1.4 The Label Powerset Method

The *label powerset (LP)* method addresses the independence assumptions of the *binary relevance* transformation by treating every unique combination of labels in the training dataset as a separate new target class to be learned. A separate binary classifier is then trained for each unique combination [226, 49]. The disadvantage of this approach is that if there are a lot of target classes, the number of occurrences of each unique combination of classes is often very rare, even in large datasets [225]. With only a small number of training examples for each class, it becomes very hard for most machine learning models to achieve low generalization error on new datapoints. Also, the number of generated target classes can produce challenges in terms of the computational complexity of the resulting algorithm [226]. Specifically, this technique creates up to $2^{|L|}$ binary classification problems, typically the number of classifiers growing exponentially with $|L|$ [25]. Furthermore, it is also likely that new data points will have unique combinations of class labels not seen in the training data. Classifying these data points with 100% accuracy is impossible with this approach.

3.3.1.5 Stacking

One approach that attempts to address the limitations of the binary relevancy, label powerset and classifier chains approach is to use *stacking* [246], a form of meta-learning. Stacking is an ensemble method, where the predictions from a number of base classifiers are used to train a separate meta-classifier on the same supervised learning task. Stacking has been shown empirically to be particularly effective for solving multi-label classification problems [145]. In the context of MLC, the meta-classifier is trained on the outputs of a set of models trained using the binary relevance method [49], where each model predicts a separate class. The main advantage of this technique is that the stacked model is able to correct the initial predictions for a class in light of the initial predictions of all the other classes [227, 184], thus overcoming the label independence assumptions of both the binary relevancy method and the classifier chains approach. This can also be thought of as a form of multi-task learning, where training a classifier on several independent tasks improves its performance on all tasks. Stacking is usually more computationally efficient than the label powerset method, requiring fewer classifiers to be trained (two per label, instead of one classifier for every unique combination of labels), and does not suffer from the problem of rare target classes. The features used to train the meta-classifier are also an important consideration when training a stacked ensemble classifier. Using the confidence estimates of the base learners in place of the binary label predictions has been shown to improve the classification accuracy of the meta-classifier [221, 198].

While problem transformation approaches focus on adjusting the problem so that traditional machine learning classification algorithms can be used, other research has focused on adapting the algorithms themselves to perform multi-label classification.

3.3.2 Algorithm Adaptation Methods

A number of approaches have been used to modify existing machine learning classifiers so that they can handle multi-label classification problems. Boosting approaches are one common approach to algorithm adaptation. *Adaboost.MH* extends Adaboost to minimize Hamming loss over the set of output labels and *Adaboost.MR* extends Adaboost to find the group of labels with the optimal ranking [196]. Another common approach is to adapt the k-nearest neighbor algorithm (kNN) for this problem. BRkNN first performs binary relevance to get a set of predictions for each class, and then uses the kNN algorithm to find the most similar k set of neighbors in the training data based on output labels [214]. The output labels are aggregated across the nearest neighbors to determine the final set of predicted classes. ML-kNN computes the output labelset based on the prior and posterior probabilities for each of the k nearest neighbor labels [254]. In addition, a number of other traditional machine learning algorithms have been extended to handle the multi-label classification problem, including neural networks [253], decision trees [233], and support vector machines [12, 82, 62]. This research will focus on applying problem transformation methods to label words with concepts and detect causal relations rather than adapting existing algorithms to tackle these problems. This will allow more rapid experimentation with a wider range of existing algorithms than would be possible when adapting individual algorithms to solve these problems. For a more complete review of algorithm adaptation methods for multi-label classification, see [149].

Multi-label classification presents a more challenging problem than multi-class classification because the output labels depend on one another as well as on the inputs. However, the output is still restricted to a set of labels. There exists a wide range of more complex problems that cannot be solved by MLC alone because the output consists of complex structured objects, such as graph structures or natural language sentences. I will cover this topic in the next section, before describing the field of deep learning, which consists of a set of algorithms especially well adapted to process unstructured data such as text documents, and which can also be used to produce more complex structured outputs.

3.4 Structured Learning

Most machine learning algorithms are designed to solve problems where the output is a single label, and thus could be deemed "simple" [43]. Simple problems include regression problems, where the label is a single continuous number, and both binary and multi-class classification problems where the label is categorical. However, there are a set of problems where the output is not "simple" and is said to contain structure in some form, such as sequences, strings, lattices, trees or more general graph structures [153]. This type of problem is called *Structured Learning*, or *Structured Prediction*. Usually, the structures predicted involve dependencies between different labels in the output [55], as is common with some multi-label classification problems. However, it is the additional structure that exists between these labels that differentiates structured learning from multi-label classification. For instance, when translating a sentence, the order of the words in the output matters, not just which words are translated. This structure results in an output space that is exponential for most problems [153], and will contain many structures not observed in the training data, which is what makes this such a challenging problem.

The literature as a whole lacks a consistent definition of structured learning; it is typically defined in terms of the sort of problems it encapsulates

[43]. Some examples of these kinds of problems include machine translation, natural language parsing, part-of-speech tagging, creating textual descriptions of visual scenes [55], image segmentation, and document summarization [45]. Structured learning therefore encompasses a wide variety of complex NLP tasks as well as problems from other domains such as computer vision. In each of these types of problem, the number of predicted labels varies over different data points [43], and often the output can be represented in the form of a graphical structure, for example a parse tree (e.g. natural language parsing), a tag sequence (e.g. part-of-speech tagging) or a bipartite graph (e.g. sentence alignment in machine translation).

In 2006, Hal Daumé III attempted to provide a more formalized definition of Structured Prediction in his thesis on the topic [43]. He describes two conditions that together are necessary and sufficient to identify a structured prediction problem. Firstly, the training data contains strong correlations between the output labels. Secondly, the loss function does not decompose simply over the output vector such that it can be broken down into a relatively small number of classification problems. To put this more formally, there is no polynomial decomposition of the loss function over the output labels, and the loss function is not invariant over identical permutations of the output labels [43]. This second condition is necessary to differentiate structured prediction from multi-label classification, for which only the first condition holds.

Three of the research questions outlined in Chapter 1 are examples of structured learning problems; Research Questions 1 and 2 are both sequence labeling problems, while Research Question 4 involves predicting a causal model, which takes the form of a graphical model. Building machine learning systems for structured prediction presents many additional challenges, including how to efficiently integrate learning and inference, and how to assign credit for partially complete solutions. In the next section I will discuss the main challenges in building structured prediction systems, and then describe some of the existing solutions.

3.4.1 Challenges in Structured Learning

At the heart of any structured learning problem is the following computation:

$$\hat{y} = \underset{y \in Y}{\arg\max} f(x, y; w) \tag{3.1}$$

This calculation chooses the structure $\hat{y}$ from the set of all possible structures Y that maximizes some function f over the set of all inputs x

in some dataset, where the structured model is parameterized by the weight vector w [46]. The weight vector w represents all the parameters learned by the machine learning model used to solve the structured prediction problem, whether this is a simple logistic regression model, or a non-linear deep neural network consisting of many layers.

In most 'simple' classification problems, this computation is tractable as Y lacks structure, and the algorithm only has to choose one from a handful of potential classes. However, in structured learning, there can be a large number of possible structures that y can take, many of which will not have been observed in the training data. In a natural language parsing problem, for example, the argmax computation would potentially have to iterate over all possible parses for a given sentence. Similarly, in machine translation problems, the model has to consider all possible translations in the target language. Consequently, structured learning algorithms involve an 'inference phase' or 'search phase' not present in machine learning problems where only a single label is predicted. Similarly, some approaches to multi-label classification use a separate model to search the space of possible output vectors, for instance the BRkNN algorithm [214] and the ML-kNN algorithm [254] discussed earlier in section 3.3.2.

The inference phase is responsible for solving the argmax computation shown in equation 3.1. In the general case, this can involve an exhaustive enumeration of all possible outputs and this can be computationally intensive or even intractable for real problems with complex structures [37, 54, 219]. However, empirically most possible outputs are unlikely when conditioned on the inputs, allowing for more compact models than would otherwise be possible [219]. In order to avoid an exhaustive search of all potential output structures, approximate inference algorithms are usually used including dynamic programming algorithms such as the Viterbi algorithm [76, 126]. Re-ranking approaches are another common solution where the top N candidate solutions are re-ranked by a second model [35, 36], although this assumes a baseline model with reasonable computational performance. Alternatively, structured learning can be re-framed as a search optimization problem, solving both learning and inference in a single approach, as discussed in section 3.4.8 later in this Chapter.

The choice of loss function has a large impact on the design of a structured learning algorithm. For a 'simple' classification problem, distinguishing between a correct and incorrect prediction is straight forward, the answer is either right or wrong and a simple loss function such as log loss or 0/1 loss can be applied. However, when the output is 'complex', the problem is typically more complex than a binary decision because we may wish to distinguish between an answer that is almost correct, and one that is completely wrong [54]. For example, in a part-of-speech tagging problem, if the model makes a mistake on a single word, we may want the loss to be lower than if it predicted the tags for all words incorrectly. One example of this sort of loss function is *Hamming loss*, which measures the fraction of incorrect labels in the output. The choice of loss function is critical to effectively solving the classification problem because providing partial credit for a nearly correct answer can be very useful in guiding the learning process towards the optimal solution during the training phase [45], provided the learning algorithm can take advantage of this form of loss function.

One of the principle challenges to building effective structured learning algorithms is taking advantage of the dependencies between the labels in the outputs. As features in the outputs are not independent, exploiting these dependencies is critical for good classification accuracy. For instance, in fraud detection, the sequence of financial transactions is very important for detecting fraud; looking at an individual transaction in isolation is often insufficient to determine whether it is fraudulent. However, depending on the specific problem, the possible number of interactions between the output labels can be exponential in nature [219]. One approach to solving this problem is to focus on local interactions between labels, for instance in a named entity recognition problem, the system may only examine relations between the current and last two named entities. However, this may not be sufficient where there are long distance relationships between the output labels [54], such as in a parsing problem where relations may exist between words at the start and end of the sentence. Some approaches avoid learning dependencies altogether, instead training a series of local classifiers and enforcing constraints over the predicted outputs [173].

Computing the entire joint distribution between the input and output labels for a structured prediction problem is infeasible for most practical problems [54]. For sequence learning, the problem can be solved by conditioning each prediction on both the previously predicted labels, and the inputs observed up to that point in the sequence. However this approach can also be adapted to solve any structured prediction problem, by decomposing the problem into a sequence of decisions which guide the construction of the structured output [45, 178]. In this method, each subsequent decision is conditioned on some or all of the previous decisions made by the classifier, in addition to the current inputs. For example, in a part-of-speech tagging problem, the current word's tag depends on the tags of the previous words, and in a shift-reduce dependency parser, the parsing problem can be decomposed into a sequence of shift-reduce actions, for example in [156].

As this approach relies on using predictions made earlier by the same model, this can cause challenges when training the model, because unless those predictions are 100% accurate, the model will be learning from unreliable data points. This is particularly problematic at the start of the training phase, when the initial predictions will be very noisy because the model has not had chance to learn effectively from the data. If instead the labels from the training data are used to simulate perfect predictions, for instance in [126], this can also lead to problems at test time as the algorithm has to rely on its own imperfect prior predictions when classifying unlabeled data. The model therefore needs to be robust enough to correct future decisions in the presence of earlier mistakes, and if it is only trained as if it has never made a mistake, then this is not possible. A more robust approach that has been show empirically to produce better results [45] is to start training the algorithm using the perfect predictions derived from the training data, and gradually switch during training to use the model's own noisier predictions. This approach was pioneered with the SEARN algorithm [45], which will be covered later in this Chapter. An alternative solution is to use an online algorithm such as the structured perceptron of Collins [35]. An online algorithm can be trained using its own previous predictions, and because it is trained iteratively, the algorithm gradually relies more and more on these predictions as training proceeds and they become more accurate.

3.4.2 Sequence Labeling

Sequence labeling is one of the simplest forms of structured prediction, where the input is a sequence of observations of some kind, and the output is a sequence of labels drawn from a fixed alphabet. For each element in the input sequence, a single output label is predicted. Often these labels are referred to as states, and the problem is viewed as one of predicting transitions between different states. One example of this sort of problem is part-of-speech tagging, where each word in a sentence is labeled with its corresponding part of speech - noun, verb, adjective, etc. Another example would be determining which of the transactions in a sequence are fraudulent. In both cases, the labels or states can not be reliably assigned to each observation without a knowledge of both the previous observations and the states, for example the word *bank* can be either a verb ('they *bank* at the building just down the road') or a noun ('they went to the *bank*'). Similarly, determining if a single transaction is fraudulent depends on the history of previous transactions that describe a person's usual spending habits, and also depends on which of these transactions were deemed fraudulent or not.

A number of different algorithms and methods have been developed to solve the sequence labeling problem. In this section I will cover three of the more common categories of techniques, linear chain probabilistic graphical models, sliding window methods, and recurrent neural networks.

3.4.2.1 Linear Chain Probabilistic Graphical Models

A probabilistic graphical model is a probabilistic model that expresses the conditional dependencies between random variables using a graph structure. A number of linear-chain probabilistic graphical models have been developed for sequence labeling, including hidden Markov models (HMMs) [175, 241], maximum entropy Markov models (MEMMs) [142] and conditional random fields (CRFs) [126]. These models have been applied to many NLP sequence labeling tasks, including part-of-speech tagging, named entity recognition, information extraction and text segmentation [129].

All of these algorithms are restricted by the Markov property - observations in the sequence are labelled based on only a small number of previous labels or tags, which is necessary for most problems to be tractable [129, 45]. MEMMs are a discriminant version of HMMs, combining the aspects of an HMM with a maximum entropy model. This also allows them to incorporate additional features that make use of multiple previous observations and states, something that is not possible with the standard HMM due to its generative nature [54]. Consequently, MEMMs are typically more accurate than HMMs in practice, and are usually faster in the training and decoding phases than both HMMs and CRFs. However, MEMMs suffer from the label-bias problem where states with low probability transitions effectively ignore the observation data [54, 129].

In order to overcome the label-bias problem, CRFs were created in [126], and are generally more accurate than both HMMs and MEMMs [129]. Lafferty, et al. compared the accuracy of the HMM, MEMM and CRF models on a part-of-speech tagging problem [126]. When they constrained the MEMM and CRF to use the same features as the HMM, they found the error rates were 5.69% for the HMM, 6.37% for the MEMM and 5.55% for the CRF. The authors then added additional spelling features that the HMM wasn't able to utilize, and found the error rates dropped further to 4.81% for the MEMM and 4.27% for the CRF [54]. This result argues that the MEMMs' discriminative nature leads to its improved performance over the HMM, while the CRF is more accurate overall because it is a discriminative model that does not suffer from the label-bias problem. However, in all of these approaches, the models are trained using known labels taken from the training data. As a result, they all struggle to deal with the uncertainty of relying on their own predictions when classifying new unlabelled data points.

3.4.2.2 Transformation Based Learning

In 1993, Eric Brill presented the 'Brill Tagger' for part-of-speech tagging as part of his PhD thesis 'A corpus based approach to language learning' [13]. The Brill tagger is an example of 'Transformation-Based Learning'. The Brill tagger initially assigns each word its most common tag from the training corpus. Then the tagger applies a series of transformation rules, that will correct a specific tag to some other tag under a specific context. This context can include the subsequent or preceding words and\or word tags in the sentence. Given a set of manually defined transformation rules, rules learned using a statistical approach, or a combination of both, the system determines the subset of rules which result in the biggest reduction in the tagging error on the training data. Consequently, this technique is also referred to as 'error driven learning'. One advantage of type of tagging model over probabilistic models and other statistical models is that the transformation rules are easier for a human to interpret and understand compared to the parameters learned by these probabilistic models. Hasan et al [93] compared the performance of the Brill Tagger to that of a HMM and unigram and bigram tagging approaches and found the Brill Tagger to out-perform the other techniques on part-of-speech tagging tasks involving three different African languages.

3.4.2.3 The Sliding Window Method

The sliding window method converts the problem of sequence labeling into a supervised classification problem. For every label to be predicted in the sequence, for a window of size w, the current observation and $(w-1)/2$ observations before and after the current label are used to make the prediction. Null tokens are added to the start and end of the sequence so that all windows are of a fixed width. The model then predicts the current label based on the surrounding observations. The main advantage of this approach is that any machine learning classifier can be used to solve this problem. However, because only the observations are used as features to train the model, this approach is unable to take advantage of dependencies between labels [54].

A number of studies have shown this to be an effective approach to sequence labeling tasks. In [105], the authors used a sliding window tagging model to attain a weighted mean F1 score of 0.73 on the task of assigning words to different concepts in student science essays. Qian and Sejnowski

[174] trained a neural network with a 15-letter sliding window of amino acid sequences to predict the secondary structure of a protein. Eskin, et al. used a sparse Markov transducer to compute a dynamic siding window over system call traces to create an intrusion detection system [64]. Sanchez et al [194] found that a simple window-based part-of-speech tagger with a window size of 3 out-performed a HMM tagger on the same dataset.

3.4.2.4 Recurrent Methods

One way to improve the sliding window method is to incorporate the previously predicted labels as additional inputs to the model [54]. For a window of size w, the model's $(w-1)/2$ previous predictions are typically used to predict the current label, in addition to the sliding window of observations. Bakiri, et al. [2] applied a recurrent decision tree with a 7-letter window to build a system to pronounce English letters. They compared the accuracy of a decision tree trained with a recurrent sliding window to one trained only on a sliding window of observations. The task was English pronunciation, and they found the recurrent sliding window method to be around twice as accurate at predicting word pronunciations than the sliding window technique. They also found that training the classifier in a right-to-left direction was more accurate than training it from left-to-right. As with the sliding window method, the key advantage of this approach is that it can use any supervised classification algorithm.

The important question that arises when using this approach is: What labels should be used when training the model? The 'ground truth' - the actual labels from the training dataset can be used, but this can introduce errors when running the algorithm on new data as the algorithm has been trained only on correct labels, as discussed in section 3.4.1. Alternatively, a sliding window model can be trained first, and its predictions used to train a recurrent sliding window model [54]. This produces noisier predictions for previous labels which are closer to what will be observed when the algorithm is run on new data. A similar approach is used in a recurrent neural network (RNN), as discussed later this Chapter in section 3.5.

3.4.3 Data Driven Natural Language Parsing

Another common application of structured learning algorithms is in constructing discriminative natural language parsers that can learn from a labeled dataset. A classifier-based parser typically consists of three principal components: a parsing algorithm which defines the parsing procedure as a sequence of simple parsing decisions; a feature extractor which extracts a set of features representing the current state of the parse; and finally a classifier which maps the parsing states represented by the feature extractor to parsing actions that are used by the parsing algorithm [156]. Most structured learning approaches to natural language parsing fall into the category of conditional-history based parsing models. Similar to the sequence learning algorithms described in the section 3.4.2.4, conditional-history parsers make a sequence of parsing decisions, each decision conditioned on all of the prior parsing decisions made by the parser, in addition to the words present in the sentence. This is achieved by using the parser's previous parsing decisions as additional inputs into the feature extractor. History-based parsing models were first introduced by Black in 1992 [9], and subsequently adopted by several authors, for example [177] and [34].

More recently, conditional-history based techniques have been developed for dependency parsing in the form of transition-based parsers. Dependency parsing is an approach to syntactic analysis centered around the idea of a dependency grammar. In a dependency grammar, a dependency is the idea that words are connected to each other by directed links, representing asymmetric binary relations. A dependency parse of a sentence thus consists of a sequence of binary dependency relations defined between the words in a sentence to form a graph structure. Transition-based dependency parsing represents the parsing problem as a sequence of decisions that read words in sequence from an input stream and combine them iteratively to form a dependency parse-tree [60]. A number of different machine learning algorithms have been used to train transition-based dependency parsers, including SVMs [159], the structured perceptron [257] and a two-layer neural network [23]. Since the rise in popularity of deep learning approaches, recurrent neural network models, such as LSTM networks (long-short term memory), have recently been used to train shift-reduce transition-based parsers and have achieved state-of-the-art accuracy [250, 60] while retaining a high parsing rate. One recent advancement incorporates an attention mechanism into a neural-network based shift-reduce parser [57]. Graph based dependency parsing models also exist that incorporate a bi-directional LSTM POS tagger directly into the dependency parser to improve parsing performance [152].

3.4.4 Bayesian Networks

A Bayesian Network is a probabilistic graphical model that uses a directed acyclic graph to represent the conditional dependencies between a set of random variables. Bayesian networks have been applied to a number of different structured learning problems, including a variety of NLP problems. In 2003, Peshkin et al applied a Dynamic Bayesian Network (DBN) to build a part-of-speech (POS) tagger [169]. A DBN is a Bayesian network unwrapped in "time" (i.e. over a sequence of items) allowing it to represent dependencies between adjacent items. Representing morphological features from the target word, as well as contextual features in a DBN, they were able to achieve state-of-the-art performance on unknown words (i.e. words absent from the training data) at the time of publication. In 2005, Savova and Peshkin also used a DBN to construct a dependency parser for English [195]. Their system modeled the dependency parsing process as a sequence of local decisions: for each word in the sentence, determine whether to create a dependency to the left or right, or defer that decision until later. Each decision made by the network depended on a set of local features: the word, the POS tag of the word, the words to the immediate left and right and their POS tags, and the number of dependents currently linked to the current word and that of its neighboring words. Their system achieved an accuracy of 79% correct link attachments for directed dependencies, and 82% for undirected.

In addition to POS tagging and dependency parsing, Bayesian Networks have also been used for word sense disambiguation (WSD). A lot of words have multiple different meanings depending on the context in which they are used. For instance, the word bank can mean 'an organization which stores and lends money' or 'a slope of land'. WSD involves determining the correct meaning, or 'sense', for a word given its context. In 2000, Chao et al used Bayesian networks as part of a WSD system. They used Bayesian networks to represent knowledge extracted from the training data to model the selectional preferences of adjectives, and used them to disambiguate the word senses of unseen adjective-noun pairs on open domain (unrestricted) text. They reported an accuracy of 81.4% at this task, which was competitive with other approaches at the time.

3.4.5 The Structured Perceptron

The perceptron algorithm was originally invented by Rosenblatt [186], who built on the earlier ideas of McCulloch and Pitts [143] to build a computational model of how a biological neuron learns. The perceptron has since been shown to be competitive with a number of classification algorithms such as support vector machines [35]. In his seminal 2002 paper [35], Collins showed how Rosenblatt's perceptron algorithm could be extended to handle multi-class classification problems. Furthermore, he showed it could also be adapted to solve structured learning problems by introducing a $GEN(x)$ function. This function iterates over all possible output structures for a given input x, allowing the perceptron to rerank these structures to determine the optimal prediction. Collins went on to show that the structured perceptron, as it has since become known, out-performed a maximum entropy tagging model on 2 sequence labeling tasks, part-of-speech tagging and noun phrase chunking.

In the same paper, Collins also introduced the averaged perceptron, a variant of the voted perceptron introduced in [71]. The voted perceptron stores all weight updates that occurred during training, and takes a weighted majority vote over these updates to make a prediction. The weight of each update is based on how long the updated weights survive during training before a subsequent change in the weights occurs. The averaged perceptron instead computes and stores a single value for each of the perceptron's weights as the average over all updates, resulting in a model that is equivalent to the voted perceptron, but requires less storage space, and is more computationally efficient when making predictions. Collins showed that on both of the sequence labeling tasks, the averaged perceptron achieved higher classification accuracy on the test data than the regular perceptron algorithm. This indicates that the procedure of averaging the weights acts to regularize the model and reduce overfitting. The structured perceptron has since been shown to be effective at solving a variety of more complex NLP structured learning tasks such as dependency parsing [156, 102], coreference resolution [8] and machine translation [252].

One limitation of the structured perceptron is that it requires searching over all possible output structures in order to solve equation 3.1. In many problems this may be intractable, or all of the possible outputs may not be analytically available [43]. To address this problem, Collins and Roark [37] proposed a modification to the structured perceptron which replaces the argmax from equation 3.1 with a beam search algorithm, to give:

$$\hat{y} = BeamSearch(x; w) \tag{3.2}$$

Beam search is a modification of the best-first search algorithm, which searches through the search space by expanding the most promising nodes first, according to some heuristic [192]. Beam search modifies best-first search by keeping only a fixed number of 'best' candidate solutions around [192]. The key insight in the incremental perceptron is that as soon as the beam search algorithm has made an error, we can detect it and abort without completing the search. Empirical results on natural language parsing problems have demonstrated that this approach typically produces faster convergence and better results [37, 43]. Furthermore, it removes the need for an external model to fulfill the role of the $GEN(x)$ function, or the need for potentially expensive dynamic programming algorithms such as the Viterbi algorithm to perform inference. While a beam search is usually only performed at test or inference time, recent work has used ideas from imitation learning (see Section 3.4.8) to allow beam search to be performed at training time [151].

3.4.6 Reranking

In the last section, I briefly described how the structured perceptron modifies the regular perceptron to solve structured learning problems by reranking candidate solutions. Reranking is a general approach to solving structured prediction problems that uses a variety of different approaches in addition to the structured perceptron [43]. The rationale behind this reranking approach is that often we have an approach for solving a problem but are unable to use the full set of features we would like to use, or are unable to directly optimize the loss function we want. Reranking allows a simple model to be used to produce a “top-n” list of candidate solutions from the complex output space. A separate model can then be used to rerank this “top-n” list. Since the size of this list is fixed, a much richer set of features can be used over the input and output spaces that would otherwise render the problem intractable. Furthermore, the reranking process can often be optimized using a loss function closer to the desired one [43]. As a result, reranking has been applied to a variety of NLP problems, including parsing [37, 22], machine translation [252, 24], and semantic role labeling [222]. More recently, these techniques have been applied to neural network parsing models. In 2017, Fried et al [72] applied generative reranking techniques to neural network parsing models, and in 2018 Zhang et al [256] developed neural ranking models for dependency parsing.

3.4.7 Large Margin Methods

A number of structured learning algorithms have adapted large margin methods to the domain of structured prediction. Large margin, or maximum margin methods learn a separating hyperplane between two target classes that maximizes the margin of separation of the two classes. The perceptron algorithm discussed earlier is a form of online margin classifier and has been adapted to the large margin case, as described in [71].

In [224], Tsochantaridis developed the SVM$^{\text{STRUCT}}$ formalism to adapt the support vector machine algorithm to the task of structured prediction by developing a plane cutting algorithm that iteratively adds in constraints to the optimization problem on an “as needed” basis [43]. Similarly, Taskar et al. [219] developed the maximum margin markov network (M^3N) to adapt the CRF model to maximize the margin of separation between the classes. The key difference in these two approaches is that the M^3N scales the size of the margin by the loss function, whereas the $\text{SVM}^{\text{STRUCT}}$ approach scales the margin by the size of the training error [43]. In a comparison by Nguyen and Guo [153], the $\text{SVM}^{\text{STRUCT}}$ showed superior generalization performance to the CRF algorithm for some sequence labeling problems.

3.4.8 Imitation Learning

Imitation learning is a supervised learning approach for solving sequential decision making problems using reinforcement learning. Reinforcement learning is a form of machine learning where an agent learns to optimize its utility given its environment. The agent has a set of actions by which it can interact with the environment and alter its state. By performing actions and receiving feedback in the form of rewards and punishments, over time the agent learns a policy which determines the optimal set of actions to make from each possible state in order to maximize its utility function. When reinforcement learning is used in a supervised learning setting, it is referred to as 'imitation learning'.

Most approaches to structured prediction treat learning and search (inference) as separate parts of the algorithm, normally utilizing different methods for each. For example, the incremental perceptron of Collins [37] uses a perceptron for learning and beam search for inference, while the CRF algorithm [126] uses a dynamic programming algorithm called the Viterbi algorithm for inference. One interesting alternative idea is to embrace search and treat structured prediction entirely as a search problem - the learning component then learns to search rather than to predict by re-framing the problem as a reinforcement learning problem [46].

The first algorithm to take this approach was the SEARN algorithm (for 'search-learn') [43]. SEARN decomposes the structured prediction problem into the problem of making a sequence of cost-sensitive classification decisions which result in the construction of the structured output. SEARN works by maintaining a current policy, and training a classifier using this policy to generate new training data, which are then used to learn a new policy. The SEARN algorithm starts with an initial optimal policy which is constructed such that the set of actions taken result in optimal performance on the training data. However, as training proceeds, it slowly deviates from this optimal policy to the learned policy. This mitigates the issues discussed in section 3.4.1 where structured learning algorithms make errors when labeling new data points because they were not trained on their own previous noisy decisions. When constructing a classifier for each training iteration, SEARN trains a separate cost-sensitive classifier for each decision to be made. One advantage of the SEARN algorithm is that any cost-sensitive classification algorithm can be used to construct the classifier. Furthermore, it can also optimize any arbitrary loss function as the classification decisions are cost-sensitive, and the cost is determined by the chosen loss function.

The performance of SEARN was compared to the structured perceptron, CRF, SVM$^{\text{STRUCT}}$, and M^{3}N models on several structured learning tasks, including handwriting recognition, Spanish named entity recognition, syntactic chunking and a joint chunking and tagging task [43]. Different SEARN models were trained using a perceptron, logistic regression and an SVM trained with a linear and a quadratic kernel. The authors found that the SEARN algorithm had the highest accuracy on all but the syntactic chunking task, where the CRF model proved to be the most accurate. The SVM algorithm had the lowest classification error of all the base-classifiers used by the SEARN algorithm on every task except the syntactic chunking task. SEARN has also been shown to be effective at other NLP tasks, including entity detection and tracking, and document summarization [43, 44]. Recent advancements in imitation learning involve learning better search policies by a process of introspection, allowing a model to evaluate its own search strategies in order to improve them [211]. In addition, some authors have applied ideas from imitation learning to allow models to perform a beam-search during training time, not just test time [151], extending the learning to search paradigm with a beam search.

One criticism of imitation learning approaches is that because the algorithm's prediction affects its future inputs, this violates the assumption that the individual training examples are drawn independently, and are identically distributed (IID) [187]. This IID assumption is made by most statistical machine learning approaches. However, this is also true of any structured prediction approach that conditions its future predictions on its past actions, such as the CRF and MEMM algorithms. Furthermore, SEARN can leverage any classification algorithm, and is not restricted to only algorithms for which the IID assumption holds.

3.5 Deep Learning for Natural Language Processing

Deep learning is the study of so-called 'deep' methods of building machine learning models. Dealing primarily but not exclusively with the recent advances in training deeper neural network models with many hidden layers, this subfield of machine learning studies the construction and application of machine learning models that are capable, through exploiting multiple processing layers, of learning multiple levels of feature representation and abstraction [130]. Deep learning has proven most successful at solving problems that use unstructured data, such as in computer vision, speech recognition and NLP. In this section, I will focus on the recent advances in deep learning models for NLP tasks, where deep models have been developed that achieve state-of-the-art accuracy on a number of core tasks, including sentiment analysis [206], speech recognition [42], and information retrieval [202]. Deep neural networks present a very flexible and powerful set of machine learning techniques that can be adapted to solve a wide variety of more complex problems, including structured learning problems, by adapting the learning algorithms and network topology to fit the problem. As a consequence, deep neural network models have also achieved state-of-the-art results in various NLP structured prediction tasks, including language modeling [147], natural language parsing [207], and machine translation [74, 218]. In this section I will describe the main types of deep learning algorithm that can be used to solve complex NLP structured learning problems, such as the some of research problems discussed in this thesis. However, I will start by discussing how neural word embeddings have become an integral part of most deep learning NLP models, and how these embeddings are derived using neural language models.

3.5.1 Neural Word Embeddings

Representing words using discrete representations is insufficient for many natural language processing tasks because it fails to capture the semantic and syntactic similarities between words [131]. Due to large vocabulary sizes and the Zipfian nature of word frequency distributions, where a large proportion of words appear very infrequently, models built using discrete representations tend to scale poorly and do not generalize well to new data points. Consequently, a number of vector-space models of semantics have been proposed over the years based on the distributional hypothesis proposed by Harris in 1954 [92], which states that the meaning of a word is determined by the various contexts in which it occurs. These models represent each word as a sparse high-dimensional vector, where each element represents a measure of the association of the word with some context, such as a document, a sentence or a sliding window of words [228]. NLP applications built using these types of distributed representation tend to generalize better than discrete representations. This is because similar words tend to appear in similar contexts and so have similar representations under this model.

More recently, it has become popular to use neural network models to learn dense vector representations of words, an idea that dates back to the work of Rumelhart et al. in 1986 [190]. The first word embeddings were developed through a series of papers published in the early 2000's describing 'Neural Probabilistic Language Models'. In their seminal 2003 paper, Bengio et al learned a distributed vector representation of words, called an embedding, by training a neural-network language model to predict which words came next in a sentence [5]. This approach was shown to out-perform previous language models based on n-gram probability models. Then in later work in 2011 by Collobert and Weston [38, 39], neural word embeddings derived from pre-trained language models were shown to perform well across a variety of different tasks including part-of-speech tagging, chunking, named-entity recognition, and semantic-role-labelling. One useful property of embedding models is that the pre-trained embeddings can be fine-tuned for specific tasks by incorporating them as part of a deep neural network model, and updating the word vectors using back-propagation. When embeddings are used in this way, this demonstrates a form of transfer learning; semantics about a word's use are transferred via its embedding vector to aid in performing these separate but related tasks.

The main drawback with these early embedding models was scaling the algorithm to large corpora containing very large vocabularies. To address this issue, Mikolov et al. [147] developed the *word2vec* algorithm, which uses a hierarchical softmax output layer combined with a simpler skip-gram language model to create a more efficient model that could scale to large corpora, while still attaining state-of-the-art performance in language modelling tasks. Furthermore, they showed that the embeddings learned by this method perform well on word analogy tasks, demonstrating that the learned representations are capable of representing a wide range of different relationships between words. In more recent work, a number of authors have trained deep-language models [52, 171, 249] capable of learning *contextualized word embeddings* - word embeddings that take into account the context in which a word is currently being used. Incorporating these language models into deep neural networks that are then trained to perform other tasks, such as sentiment analysis, question answering and coreference resolution, has produced gains in accuracy in these tasks, in many cases advancing the current state-of-the-art performance in these tasks [171].

Neural word embeddings have become the standard way of representing words in most deep neural network models. In recent years, a number of different types of neural network have been developed which have advanced the state-of-the-art in a number of different problem domains.

3.5.2 Beyond Bag of Words Models

One of the challenges in building machine learning models to solve NLP problems is the varying length of the inputs. Most machine learning algorithms expect a fixed-sized input representation for the rows in a dataset, yet phrases, sentences and documents all have varying lengths. The traditional solution to this problem is to use a 'bag of words' representation, where each word or phrase in a document is represented by a separate feature in the dataset which indicates its presence or absence in the document. This representation has been successfully applied to a wide range of different NLP problems, such as document classification and information retrieval. However, the order of the words and the relationships between them are often very important for determining the meaning of a sentence, for instance the sentences 'the dog chased the cat' and 'the cat chased the dog' have very different meanings but the same bag of words representation. While this approach provides a fixed-length representation of a section of text, it ignores important relationships between the words that are present, such as syntactic and grammatical relationships. Consequently, there are NLP tasks where understanding the relationships between the words in some text is crucial for solving the problem at hand, for example when extracting causal relations from text, and in sentiment analysis. In the following sections, I will cover three types of deep neural network model that are capable of learning long-distance relationships between words in a piece of text: recursive neural networks, recurrent neural networks, and convolutional neural networks.

3.5.3 Recursive Neural Networks

Sentiment analysis is the problem of determining the general attitude of the writer with respect to the topic they are writing about, for instance it is used to determine if online product reviews on e-commerce sites are positive, negative or neutral. To address the limitations of the bag of words model, Socher et al. [206] developed the *Recursive Auto-Encoder* (RAE). The RAE is a recursive neural network that learns a compact, fixed-length vector representation of a piece of text by learning to parse it recursively. It uses word vectors, word embeddings pre-trained on a language modelling task, to represent words, and parses a sentence into a single vector by recursively combining pairs of adjacent vectors into a single vector. The algorithm has a dual objective function, which minimizes the classification error on the training dataset while simultaneously learning a compact representation of a sentence using an unsupervised learning algorithm called an *auto-encoder*. Socher et al. showed that their model outperformed a baseline bag of words sentiment classifier, as well as the state-of-the-art-sentiment analysis algorithm at the time [206]. The architecture of the RAE is illustrated in Figure 3.1 below:

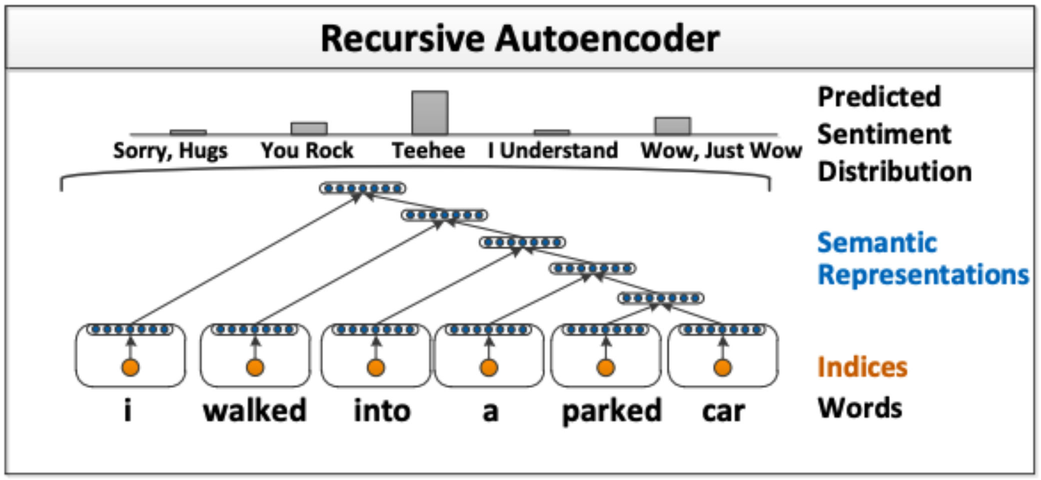


Figure 3.1: Recursive Autoencoder. Image from Socher et al 2011, "Figure 1", page 151 of 'Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing' [206]

A number of other recursive neural network models have since been developed for sentiment analysis, for example [209, 166], further improving upon the accuracy of the RAE. Recursive neural networks have also been adapted to solve complex structured learning tasks such as natural language parsing [208], and question answering [108]. In each of these approaches, sentences are parsed into a tree structure, and the model learns to compress this tree structure into a constant length vector, which is then used to build a supervised classification model.

3.5.4 Recurrent Neural Networks

Recurrent neural networks (RNNs) were developed for supervised learning problems involving sequential prediction, such as time-series prediction, part-of-speech tagging and language modelling. In a recurrent neural network, the connections between the layers form a directed cycle that creates an internal state within the network that allows it to learn to detect patterns in sequential data. So called 'simple recurrent networks', originally developed by Jordan in 1986 [110] and further developed by Elman in 1990 [63], contain context units that are connected to the hidden layer. When the hidden layer neurons fire, they activate the context units which act to echo back each of the hidden neurons' activation states on the next step in the input sequence. This allows the network to maintain a form of short-term memory, producing a non-linear dynamical system that can learn complex temporal patterns. RNN models have been used to produce state-of-the-art language models [146, 248], and have been successfully applied to a variety of sequence labelling tasks [78, 86], including word tagging problems [172, 237].

One of the difficulties with RNN's is that they are hard to train because they suffer from the vanishing gradient problem. This is where the errors diminish as they are back-propagated through the entire input sequence [150]. One solution to this problem is to use a Long Short-Term Memory model (LSTM) [100]. This is a more complex form of RNN which contains a number of gated units that determine which inputs are significant enough to remember, and when to clear its internal state. This model has been shown to be very effective at learning long distance relationships over sequences, for instance it achieved the best known results in unsegmented connected handwriting recognition [87], and has been used as part of a former state-of-the-art speech recognition system [85]. More recently, the Gated Recurrent Unit (GRU) was proposed by Cho et al. in 2014 [28]. The GRU is a variant of the LSTM RNN but without the separate memory cells, which form one of the 3 types of gated unit in an LSTM. GRU's have been shown to have comparable performances to LSTM's on a number of tasks [30], including sequence labelling, while being more computationally efficient. An illustration of a RNN applied to a sequence labelling task is shown in Figure 3.2 below:

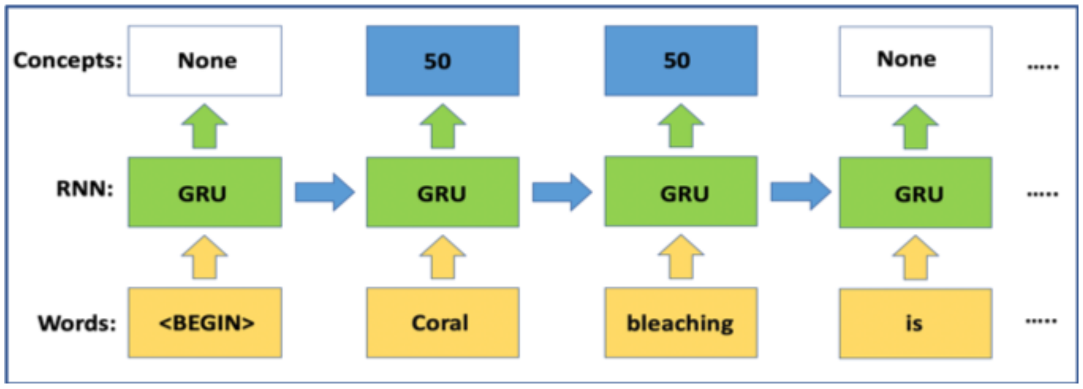


Figure 3.2: Illustration of an RNN applied to a word labeling problem. The recurrent cell in this example is a Gated-Recurrent Unit (GRU)

One limitation of RNN's is that they cannot incorporate information from future inputs when making a sequential prediction. For instance, in a part-of-speech tagging problem, information from words to the right as well as to the left of the current word can influence its part-of-speech. To address this problem, Schuster et al [197] created the bidirectional RNN, which connects two hidden layers of opposite directions to the same output layer. One hidden layer is processed in the normal forward direction, while the second hidden layer is processed via a backward pass through the input sequence. The authors showed that this bidirectional structure improved classification performance on a number of tasks, including phoneme detection. Bidirectional RNNs have been successfully applied to many different tasks, such as dependency parsing and sequence labeling [152, 223].

3.5.5 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a form of feed-forward neural networks that are designed to mimic the structure of the animal visual system. In animals, the retina consists of cells that are sensitive to small sub-regions of the retina, and are tiled across the whole visual field [104]. CNNs mimic this behavior by using small groups of artificial neurons to learn locally receptive feature detectors that are tiled across the image so that they overlap. These feature detectors can learn to detect low-level local features in images, allowing them to perform tasks such as edge detection. The process of tiling or 'convolving' these feature detectors across the image allows the network to exhibit a form of 'translational invariance' where it can detect the same features in an image regardless of where they occur [150]. These 'convolutional' layers are are usually fed into 'subsampling' layers that aggregate the local features detected by the layer below by computing an average or a max operation over the outputs, providing a small amount of shift invariance. A typical CNN has several alternating convolutional and pooling layers, one on top of the other, which allows the network to learn hierarchies of features, with the upper layers having larger receptive fields than the layers below. In 2010, Krizhevsky and Hinton used a CNN to achieve the lowest classification error on the CIFAR-10 image recognition dataset [122]. The architecture of this network is illustrated below, in Figure 3.3. Since then, CNNs have come to dominate the field of computer vision, achieving state-of-the-art results in multiple image recognition and classification tasks on a number of different datasets, including ImageNet [97].

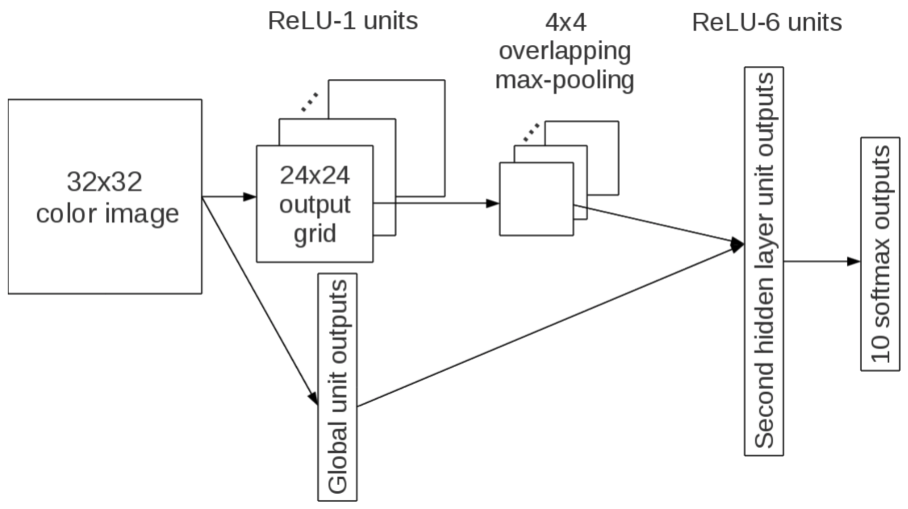


Figure 3.3: Convolutional Deep Belief Network architecture. Image from Krizhevsky and Hinton 2010, "Figure 4: The architecture of our convolutional net", page 5 [122]

However, although CNNs were originally designed to solve problems in the field of computer vision, they have more recently been successfully applied to other types of unstructured data, including text. In a pair of landmark papers from Collobert and Weston [38, 39] the pair showed how neural word embeddings trained on a language modelling task could be used to represent words as features in a 'time-delay neural network', a form of deep convolutional network. The convolutional layer formed a 1-dimensional convolution over the sequence of word vectors, the result of which was fed into a sub-sampling layer that computed a max operation over the entire sequence. Finally, the outputs from this layer were fed into a sequence of densely connected layers. This network was shown to achieve strong performance on a number of different tasks including part-of-speech tagging, chunking, named-entity recognition and semantic-role-labelling. More recently, Kalchbrenner et al. [115] built on this approach and developed a dynamic convolutional neural network for classifying sentences. This network used a novel dynamic k-Max pooling operation to sub-sample the varying length input sequences, and achieved competive results on a number of sentiment analysis tasks, and a question classification task.

One of the challenges with training deep NLP models using word embeddings is training a model that can process the large vocabulary sizes of most real-life datasets. For each unique word present in the training data, the model needs to store and compute a word embedding. This can result in very large models with billions of parameters that require large amounts of memory, take a long time to train, and are more likely to over-fit the training data. To address this issue, Zhang and LeCun [255] built a temporal convolutional network using characters instead of word embeddings for the input features. The authors used a one-of-m encoding scheme to compute m length vectors for each character. They then truncated each document to a fixed character length l, and concatenated the l character vectors together to form an $m \times l$ matrix. These matrices were then fed into a convolutional temporal network that performed a one-dimensional convolution over the sequence of character vectors. By using characters as features, the model ignored any explicit knowledge of words, phrases or sentences. Nevertheless, it was able to achieve impressive performance on three NLP tasks: sentiment analysis, ontology classification, and text classification.

Chapter 4

A Word Tagging Model for Essay Concepts

Research Question 1 asks:

> "Which of the following model types is the most effective machine learning model to automatically label words with their associated essay concepts from a pre-defined causal model?"
>
> A Window-Based Word Tagging Model
>
> B Conditional Random Field
>
> C Hidden Markov Model
>
> D Structured Perceptron
>
> E Recurrent Neural Network

To address this question, the problem was treated as a multi-label word tagging problem: given a sequence of words, predict zero, one or more tags (concept codes) for each word. This is illustrated in Figure 4.1 below, which shows an example coral bleaching essay where the words have been labeled with their associated concept codes. The concept codes in the figure refer to the codes defined in the coral bleaching causal model illustrated in Figure 2.1.

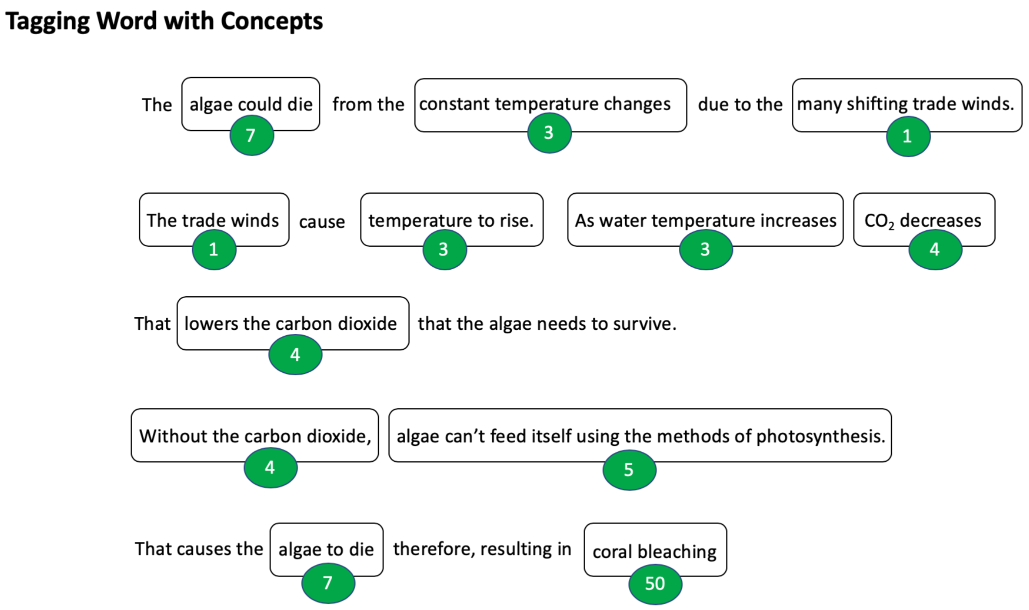

Tagging Word with Concepts
The
algae could die
7
from the
constant temperature changes
3
due to the
many shifting trade winds.
1
The trade winds
1
cause
temperature to rise.
3
As water temperature increases
3
CO₂ decreases
4
That
lowers the carbon dioxide
4
that the algae needs to survive.
Without the carbon dioxide,
4
algae can't feed itself using the methods of photosynthesis.
5
That causes the
algae to die
7
therefore, resulting in
coral bleaching
50

Figure 4.1: Labeling words with concept codes. Each box represents a separate concept, with the code denoted by the green circle.

In this chapter, I will discuss how classification performance on this task was measured and evaluated, what features were used to solve this problem and how the final set of features were chosen and evaluated, and I will compare the efficacy of the five different model types above at solving this problem.

4.1 Evaluation Metrics

The choice of evaluation metric is critical to solving any machine learning problem. The goal of a machine learning algorithm is to minimize the error on the training data according to the preferred evaluation metric, so the choice of metric influences the types of machine learning models that can be applied to solve a particular problem.

In a classification problem, the relative distribution of classes observed in the training and test data should be taken into account when selecting an evaluation metric. In our set of scientific explanatory essays, most of the labels in the dataset are quite rare, with the most common label (concept code 50) being assigned to 8.86% of words in the coral bleaching dataset, and assigned to 6.36% of words in the skin cancer dataset. Other concept codes occur even less frequently. See Tables 2.7 and 2.8 in chapter 2 for the percentages of words assigned to each concept code in the two datasets. Accuracy is considered a poor evaluation metric when the classes are imbalanced because a classifier can achieve a high level of accuracy by always predicting the most common class, without learning anything from the data. Metrics such as precision, recall, and the F_1 measure are therefore more commonly applied to datasets with imbalanced classes. These metrics are calculated using the number of correctly identified examples of the target class, the 'true positives' (tp), the number of correctly identified examples not belonging to that class, the 'true negatives' (tn), the number of examples wrongly attributed to that class, the 'false positives' (fp) and the number of examples belonging to that class that were falsely labeled, the 'false negatives' (fn). When evaluating multi-class or multi-label classification problems using these metrics, the classification problem is framed as multiple binary classification problems, one per unique class label.

4.1.1 Precision, Recall and the F_1 score

Precision measures the proportion of data points that were predicted as belonging to a particular class that were correct. It is defined as [140]:

$$precision = \frac{tp}{tp + fp} \tag{4.1}$$

Recall measures the 'false alarm rate', also known as the 'type I error rate'. Recall measures the proportion of the positive examples from the entire dataset that the algorithm correctly predicted for a particular class, and is defined as [140]:

$$recall = \frac{tp}{tp + fn} \tag{4.2}$$

While precision only measures the accuracy of the labels that were predicted to belong to a particular class, recall measures the coverage of the algorithm - how accurately does it cover the entire dataset, without focusing only on the data points the model is most confident at predicting? Modifying an algorithm to improve recall often has a negative impact on precision, and vice versa. In certain problem areas where we care more about either precision or recall, for instance when training a cost-sensitive classifier, that metric would be preferred over the other. However, if we care equally about reducing false positives and false negatives, then we want a classifier that optimizes both precision and recall. The F-score metric is typically preferred in this scenario, and is calculated as the harmonic mean of precision and recall. The general formula for the F-score is described in equation 4.3 below [140]:

$$F_\beta = (1+\beta^2)\frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall} \tag{4.3}$$

The β coefficient in the F_β calculation controls how much emphasis is placed on recall over precision in the metric. The F_1 score, which has a β value of 1, is commonly used as it puts equal emphasis on recall and precision, and is the variant I will use throughout this thesis. Recall, precision and F_1 score all range from 0 to 1.0, with higher values indicating a more accurate model. An F_1 score of 1.0 means all of the data points were classified correctly.

Precision, recall and the F_1 score are commonly used for evaluating ranking systems, such as information retrieval and recommender systems. In this context, it is common to compute precision or recall at a cutoff point k. For instance, 'precision at k' measures the precision of the first k documents. For a large number of these ranking problems, the ground truth labels are not known for every data point due to the size of the dataset; it is therefore more practical to evaluate the performance of these systems by evaluating the quality of the highest ranked documents returned. These metrics are also commonly used in text classification tasks where there is no result ranking, and a prediction is made for each data point in the entire dataset. This can roughly be thought of as using a cutoff of 1, although the ranking and classification tasks are quite distinct. It is the classification usage of precision, recall and F_1 score that I am referring to in this thesis.

4.1.2 AUC

An alternative to F_1 score for evaluating binary classification performance is the AUC metric (Area Under the Curve). This measures a single point on the 'Receiver Operating Characteristic' (ROC) curve [210], and is defined as:

$$AUC = \frac{1}{2} \cdot (\frac{tp}{tp+fn} + \frac{tn}{tn+fp}) \tag{4.4}$$

ROC curves show how the number of positively classified examples varies with the number of negatively classified examples, and AUC is an average of the true positive rate (recall) and the true negative rate. However, ROC curves can present an overly optimistic view of classification performance when there is a large skew in the class distribution because it treats both positive and negative examples equally [47].

4.1.3 Micro and Macro Averaging

The goal of solving Research Question 1 is to produce the most accurate model for labeling words with their associated concept codes. I measure the performance of each approach at predicting each individual concept code by calculating the F_1 score for each code over all of the words in the validation dataset using 5-fold cross-validation. To compare the overall efficacy of different approaches at the tagging problem, an average F_1 score is computed over all of the different classes. There are two main approaches for averaging F_1 scores over multiple classes - the 'macro-average method' and the 'micro-average method' [210].

The macro-F_1 score is calculated from the macro-averaged precision and recall. For l labels, the macro-average precision and recall are calculated by computing the mean precision and recall over all labels:

$$precision_M = \frac{1}{l} \cdot \sum_{i=1}^{l} precision_i \tag{4.5}$$

$$recall_M = \frac{1}{l} \cdot \sum_{i=1}^{l} recall_i \tag{4.6}$$

The macro-average F_1 score is calculated as the harmonic mean of the macro-average precision and recall [210]:

$$F1_M = \frac{2 \cdot (precision_M \times recall_M)}{(precision_M + recall_M)} \tag{4.7}$$

The micro-F_1 score is computed by totaling all of the true positives, false negatives and false positives across all class labels, and using those totals to first compute micro-average precision and recall:

$$precision_\mu = \frac{\sum\limits_{i=1}^{l} tp_i}{\sum\limits_{i=1}^{l} tp_i + \sum\limits_{i=1}^{l} fp_i} \tag{4.8}$$

$$recall_\mu = \frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l} tp_i + \sum_{i=1}^{l} fn_i} \tag{4.9}$$

The micro-F_1 score is then computed from harmonic mean of the micro-average precision and recall [210]:

$$F1_\mu = \frac{2 \cdot (precision_\mu \times recall_\mu)}{(precision_\mu + recall_\mu)} \tag{4.10}$$

While macro-averaging treats all classes equally, micro-averaging favors larger classes, weighting each class' contribution to the average based on its relative frequency in the dataset. This is therefore the evaluation metric chosen for this research because it provides a more accurate view of the performance of the algorithm on the task as a whole. In scenarios where you want all classes to be treated equally in the average, the macro-F_1 would be a more appropriate metric.

4.1.4 Optimizing the Micro-F_1 Metric

Like a number of other classification metrics arising from the field of information retrieval [17], the F_1 metric is not continuous, not differentiable and thus is not appropriate for gradient-based training [165]. Direct optimization is therefore difficult because it does not decompose over all examples [61]. Nevertheless, F_1 score is commonly used as an evaluation metric in class imbalanced problems for the reasons discussed in the previous section. Instead, machine learning models are typically trained to maximize classification accuracy in the belief that doing so will also optimize the F_1 score [61]. Because micro-F_1 score extends the F_1 score to measure an algorithm's combined classification performance across multiple labels, these same limitations extend to the micro-F_1 score also.

4.2 Experimental Design

When evaluating the performance of any machine learning approach, it is critical to ensure that the final solution has been evaluated on some separate held out dataset, usually called the *test set* [150, 148]. This allows us to measure the accuracy of the model at making predictions on previously unseen data points. To ensure the validity of this approach, there can not be interactions between the data within the test set and the construction of the final machine learning model, ensuring that no decisions influencing the final model were made based on this dataset. This includes any hyper-parameter tuning or feature selection approaches.

The essays in the dataset were split into 80% training data and 20% test data, as recommended in [150]. As discussed in Section 2.4, there is a large class imbalance in this dataset across both the concept codes and the causal relations, and so it is important to ensure that the training and test datasets have similar distributions of labels. The standard approach for partitioning datasets with a large class imbalance is to use *Stratified Random Sampling*, which is a sampling approach that attempts to ensure the same distribution of labeled data is present in each data partition [201].

To address this problem in the two datasets studied in this thesis, a form of Stratified Random Sampling called *Proportionate Allocation* [101] was performed to ensure that the distribution and relative proportion of labels in the dataset were roughly even between the training and tests datasets. Given the end goal of this research was to detect causal relations within the essays, the splits were chosen across essays so as to minimize the Kullbach-Liebler (KL) Divergence [124] between the distribution of causal relations across the words within the essays in each split. The split was performed at the essay instead of the sentence or word level, so that the same test data could be used for all 4 research questions, which are defined at the word, sentence and essay level (depending on the research question). Because a word can be part of multiple causal relations, and the dataset needed to be split across essays and not words, the splits were chosen using a random search rather than more analytical methods. 50,000 different partitions were chosen at random over the essays, and the partition with the lowest KL-Divergence between the training and test dataset was selected as the final partition. Because the KL divergence is undefined for probabilities of 0, causal relations occurring in less than 2 essays were not included in this test, and any splits where one or more of the remaining causal relations had zero examples were rejected. The combined KL-Divergence for all of the causal relations was computed as the sum of the KL-Divergence computed per relation. The split with the minimum total Kl-Divergence was then selected as the final dataset partition.

For discrete probability distributions P and Q, the KL-Divergence between P and Q is defined in Equation 4.11 below:

$$D_{KL}(P\|Q) = -\sum_{x \in X} P(x)\log(\frac{Q(x)}{P(x)}) \tag{4.11}$$

The training dataset was used to perform both feature selection and hyper-parameter tuning using 5-fold cross validation (CV). When performing CV, the accuracy of the model is listed on both the training folds and the separate held out folds, which are reported as the training data (CV) and validation data (CV) respectively, to clarify that these were derived using cross-validation. Figure 4.2 below illustrates the data-partitioning strategy.

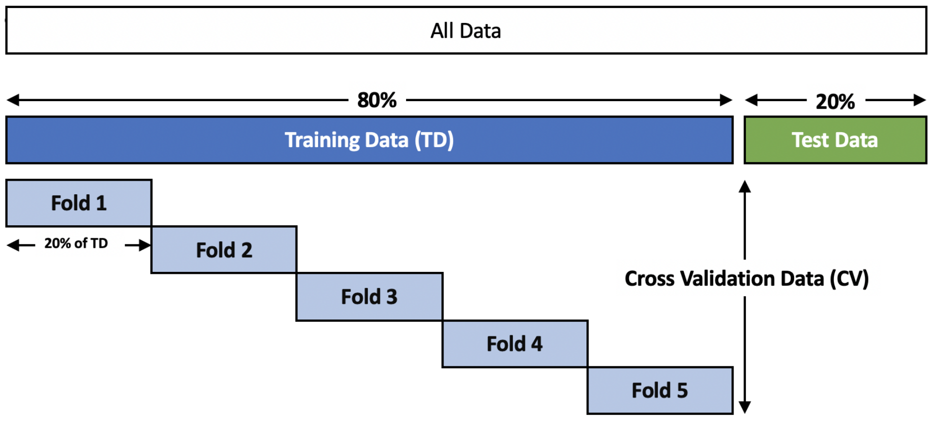


Figure 4.2: The Different Dataset Partitions. For Each Fold, the Validation Data (CV) is Shown in Light Blue. The Remaining Data within each Fold was Used for the Training Data (CV) for that Fold.

The CV folds were created by partitioning the data on the essays, and not at the word or sentence level, thus preserving entire essays within each CV fold. This prevents the algorithm learning style and grammar cues for an individual author from sentences in the training data and making use of that information to classify sentences in the validation or test data, which would potentially lead to unreliable validation and test data metrics.

Following feature selection and hyper-parameter tuning, the performance of the different models was then evaluated by their performance on the separate test dataset (the 20% partition) following training each model on the entire training dataset, this time without cross-validation. However, the model that performed best in cross-validation on the training data is the model that was chosen to be used in solving subsequent research problems to ensure there is no feedback between the final model and the test dataset. If instead the optimal model was selected based on its test set performance and this model was used to construct other machine learning models, that could result in overly optimistic test set error metrics.

The relative sizes of the training and test datasets are described in Table 4.1 below. 'Vocab' indicates the number of unique words.

Table 4.1: Dataset Partition Sizes

	Coral Bleaching				**Skin Cancer**			
Split	**Essays**	**Sents.**	**Words**	**Vocab**	**Essays**	**Sents.**	**Words**	**Vocab**
Train	901	8,280	136,957	4,279	870	8,573	145,486	4,201
Test	226	1,918	30,699	2,088	218	2,097	35,413	2,046
Both	1,127	10,198	167,656	4,770	1,088	10,670	180,899	4,702

4.3 Statistical Testing

When comparing the performance of different machine learning algorithms and models, using comparative techniques such as k-fold cross validation alone can provide misleading results because it is hard to determine if the differences in performances are real or due to chance. As such, some researchers utilize statistical techniques to estimate the likelihood that differences in the performance of multiple models are due to chance or not. In statistical testing, a null hypothesis is formulated stating that the observed differences between two populations are due to chance. A *p-value*, produced by some statistical test, provides an estimate of the probability of observing results at least as extreme as those observed, assuming the null hypothesis is correct [240]. Commonly a p-value of 0.05 is used as the threshold for 'significance' (called the α value). p-values lower than 0.05 are thus assumed to be 'significant', implying the observed results were very unlikely to have been observed due to chance, thus rejecting the null hypothesis [193].

There are however a number of issues with applying traditional statistical tests to compare machine learning models because they were not designed with computational methods in mind [193]. It is often impossible to verify that the assumptions of these tests hold, and it is always possible to show that there is a significant difference between two treatments provided enough data is used [109]. Due to the number of problems with applying statistical testing to this area, some authors [58, 50] have even argued we should drop the process of statistical testing entirely. However, rather than abandoning the process, it is important to determine when a statistical test is warranted, and if so, the appropriate technique or techniques to apply for a particular experiment [109].

Statistical tests come in two forms, parametric and non-parametric. Parametric tests make strong assumptions about the distribution of the underlying data, while non-parametric tests make weaker assumptions and are thus less powerful (less capable of rejecting the null hypothesis) [109]. The most commonly applied parametric statistical test is the *Student's t-test*, which tests whether two samples are drawn from the same population using differences in the observed means and standard deviation. When comparing classification performance on some task, one approach is to run a t-test on the classification metrics from both classifiers, computed using k-fold cross-validation, where k is sufficiently large, such as 10 or 30 folds [193, 109]. However, this is widely viewed as a poor choice of statistical test for comparing model performance as most of its assumptions are violated. In particular, the t-test assumes both samples are drawn independently from the same population. Re-sampling from the training data using cross-validation violates this assumption, leading to a high probability of Type I error i.e., a high likelihood that the null hypothesis will be incorrectly rejected [193, 109, 176, 53]. The t-test also assumes that the samples come from a normally distributed population, and have the same variance, which is not always true when measuring classification performance using cross-validation [109].

In spite of their weaker statistical power, non-parametric techniques are usually preferred [75] over parametric methods when comparing machine learning models because they make fewer assumptions about the underlying data. Consequently, many authors recommend McNemar's test for comparing the performance of two classifiers [193, 109, 176, 53, 41] as it has a lower probability of Type I error (as it makes fewer assumptions). McNemar's test is a non-parametric equivalent of a t-test for nominal data [144], and can be used to compare the predictions of two different classifiers on the same test dataset. Given the predictions of two classifiers, classifier A and B, and the ground truth labels, a 2x2 contingency table is computed, which looks at the number of instances of the following:

- Both classifiers were correct
- Both classifiers were incorrect
- A was correct and B was incorrect
- B was correct and A was incorrect

A χ^2 test is then calculated based on this contingency table to compare how many times each classifier was correct when compared to the other [109]. This allows us to estimate the probability that A wins over B at least as many times as observed in the experiment [193]. However, McNemar's test is a pairwise test — it is designed for comparing two populations, or in this case the predictions from 2 classifiers. If more than 2 classifiers are compared, a generalization of McNemar's test called *Cochran's Q test* [32] can be used to compare more than two sets of nominal data. Several authors propose using Cochran's Q test to evaluate ipiple classifiers to determine if there is a significant difference in their predictions [176, 125, 134].

To compare the performance of multiple machine-learning classifiers for the different Research Questions in this thesis, the following null hypothesis was formulated:

All classifiers are equally accurate at the classification task.

I applied Cochran's Q test to test the null hypothesis, using the predictions from each algorithm across all labels in the dataset. A significance threshold of 0.05 was used for every test. For the experiments where the null hypothesis was successfully rejected, I also wanted to determine if any classifier was significantly better than the other classifiers. Following the recommendation in [176], I used McNemar's test to compare the performance of each pair of classifiers, under the same null hypothesis (for the pair). For McNemar's test, the Binomial distribution was used in place of the χ^2 distribution to compute the exact p-value. However, when performing multiple comparisons in a statistical test, there is a higher chance of a Type I error due to the *multiplicity effect* [193]. This means that if a statistical test is run x times, there are x chances of observing 'significance'. This is well understood in the statistical community, and is typically addressed via a *Bonferroni correction* [176]. The Bonferroni correction divides the required significance value, the α value, by the number of comparisons. For example, if the α value is 0.05 and 5 tests are performed, then a p-value of $0.05/5 = 0.01$ or lower is required for significance. To reduce the number of pairwise comparisons, I only compared the performance of the most accurate classifier (across both datasets) with that of the other classifiers, rather than performing all pairwise comparisons. This allowed me to determine if any of the classifiers are significantly better than the best performing classifier (according to the micro-F_1 measure). Furthermore I used a Bonferroni correction to adjust the α value, dividing it by the number of pairwise comparisons to ensure an accurate estimate of significance. To compute the test statistics and p-values, the *mlxtend* python library was used [176].

4.4 Pre-Processing

Prior to performing feature selection, some initial pre-processing was performed on the dataset. Words occurring in only one sentence were replaced with a special *INFREQUENT* token. The distribution of word frequencies in a corpus of text tend to follow a Zipfian distribution [139] meaning that most unique words occur only once or a small number of times. Thus by removing such infrequent words, this dramatically reduces the size of the feature space over which the model has to learn, reducing the the risk of the model overfitting the dataset, and helps it to generalize better to new data points. While these rare words could be removed entirely, this can negatively impact word tagging accuracy by changing the relative positions of the words in the sentence. Additionally, all numbers in the text were replaced with a sequence of 0's of the same length as the original number (for example 1980 becomes 0000, while 52 becomes 00) as described in [137]. This allows the model to better generalize over numbers by collapsing numbers with the same numbers of digits into the same token.

Words were also converted to lower case, and punctuation characters and stop words were not removed as they aid classification accuracy in word labeling tasks such as part-of-speech tagging. Punctuation characters were treated as separate tokens when tokenizing sentences, and the start and ends of each sentence were padded with special start and stop tokens to ensure all word windows were of the same length and to allow the model to detect when a word was near the start or end of a sentence. In addition, I used a spelling corrector based on [160] to correct typographic errors, and noisy sentences with three or fewer words were removed from the dataset.

4.5 Feature Extraction and Selection

The word tagging problem involves utilizing two principle types of information from the sentence when tagging a word, in addition to the word itself - 1) the words surrounding the target word providing context, 2) the tags or labels assigned to the surrounding words. Each of the algorithms evaluated for solving this problem differ in how they make use of information from the word labels, so the goal of feature selection was to determine the set of features extracted from the surrounding words that were the most useful at predicting the target word's tag. This set of features could then inform the choice of features used to train each of the different models that are capable of handling arbitrary features. Feature selection is an important technique for machine learning problems with large numbers of features, such as NLP problems which tend to have high dimensionality. High dimensional datasets present a challenge to machine learning algorithms due to the so-called *curse of dimensionality*, a problem where machine-learning algorithms struggle to generalize well on datasets with a large number of dimensions. This term was originally coined by Bellman in 1957 [4] to describe the difficulty in achieving statistically significant results from high-dimensional datasets due to the sparse distribution of data in high-dimensions. High dimensional data makes it challenging for machine learning algorithms to determine which features are relevant and predictive. By performing feature selection, we can select the optimum set of features that maximize the model's classification accuracy on out-of-sample data, and lower the classification error.

A wrapper method of feature selection was used to determine the optimal feature set. This is where the same model is trained on various subsets of features to assess their utility [91]. In order to evaluate a large number of different feature sets, it is important to use an approach that is computationally efficient. The window-based classifier model (see section 3.4.2.3) was selected for this task because it is computationally efficient when implemented using a logistic regression classifier. This model also is not a structured learning algorithm and cannot make use of the previously predicted concept codes or those assigned in the training data. Because of this fact, it can be used to evaluate the quality of the features extracted solely from the surrounding words, and not their labels. Using a logistic regression classifier trained using the binary relevance problem transformation method has also been shown to work well for multi-label classification problems [106, 135]. The binary relevance transformation method is also more computationally efficient than other transformation methods such as label powerset and classifier chains. Furthermore, as is typical of NLP problems this dataset is considered 'sparse' because it contains a larger number of rare features. A logistic regression classifier can be trained efficiently on this data because it can make use of the sparsity of the data.

4.5.1 Features Evaluated

In this thesis I use the term 'feature set' to describe a set of features extracted using the same approach and the same algorithm. For instance, POS tags would be considered one feature set, and trigram features a different feature set. For this problem, I treat feature selection as the problem of determining the optimal combination of feature sets. Due to the large number of individual features extracted from each dataset, determining the optimal combination of individual features would be too computationally intensive to be practical.

Different feature sets were extracted from the words present within the context window and then used to train a logistic regression model to evaluate the performance of each feature set in isolation and in combination with other feature sets. The different feature sets included unigrams, bigrams, trigrams, stemmed unigrams and stemmed bigrams and trigrams, part-of-speech (POS) tags, dependency relations and finally Brown clustering labels. The *NLTK* software package [6] was used to compute POS tags for each word, while the *SpaCy* python natural language parsing package [103] was used to compute dependency relations and Brown cluster labels. Brown clustering is a hard hierarchical agglomerative clustering algorithm that groups words into clusters based on the contexts in which they appear [16].

Dependency parsing is a form of natural language parsing where binary relationships between words are detected and constructed into a dependency parse tree. The dependency relations were extracted by a dependency parser from the *SpaCy* python package. *SpaCy* uses the *CLEAR Style* set of dependency labels [29], which contains 52 different dependency relations, including the ROOT relation. To use the dependency relations as features, each unique dependency pair from the parse tree was extracted and used as a binary feature in the model only if that binary relation involved the word to be tagged. Each dependency relation feature encoded the word pair within the relation, the type of dependency relation and the role of each word within that dependency. For example, in the sentence "rising global temperatures caused coral bleaching", the word '*bleaching*' has the relation '*adjectival modifier*' with its head word '*coral*'. This would then be encoded as a feature as BLEACHING-AMOD-CORAL, where AMOD represents '*adjectival modifier*'. If instead, '*bleaching*' was the head word, and the relation was the same, then a different feature would be created and represented as CORAL-AMOD-BLEACHING.

For each of the ngram feature types and the POS tags, two different feature sets were computed - the first using a bag-of-word (BOW) style encoding that ignored the ngram's position within the window, and a second 'positional' variant that combined the ngram with its ordinal window position. This 'positional' variant ensured that the same ngram produced a different unique feature for each possible window position it was observed in. To achieve this, the ordinal position of each ngram within the word window was appended to the ngram to create a positional feature. For example, if the window size was 5, and the window contained the following unigrams 'coral bleaching was caused by', 5 positional unigram features would be generated: CORAL-1, BLEACHING-2, WAS-3, CAUSED-4 and BY-5. A lot of features only appeared once in the dataset, and therefore did not help the model to generalize to new datapoints. To address this, a minimum feature frequency threshold of 5 was applied, removing any features that appeared in fewer than 5 word windows for each experiment.

4.5.2 Forward Selection

Once the data was pre-processed, the final combination of feature sets was determined by finding the optimal combination of feature sets for each size of word window using a technique called *forward selection.* In forward selection, the model is initially trained once using each unique set of features to be evaluated, and the single set of features are chosen that produced the highest classification accuracy on the training dataset. A greedy search is then conducted, incrementally adding in the next set of features that produced the biggest increase in accuracy until either no further improvement is achieved [150], or the maximum number of desired features have been added. Forward selection was performed once for each odd-sized word window, ranging from 1 to 15 words. Only odd-sized word windows were chosen to ensure there were the same number of words on either side of the target word to be tagged. For this experiment, a maximum limit of 6 different combined feature sets was chosen to ensure the feature selection process remained tractable. The optimal set of features was then selected by computing the micro-F_1 score using 5-fold cross validation. For each round of feature selection, if none of the remaining feature sets added in that round improved the micro-F_1 score on the validation folds, or if the maximum of 6 feature sets had been evaluated, then feature selection was halted.

4.5.3 Feature Selection Results

The optimal size of word window was found to be 9 words (4 words each side of the target word) for both datasets, achieving a micro-F_1 score of 0.825 for the coral bleaching dataset, and 0.801 for the skin cancer dataset. Above this window size, the micro-F_1 score started to decline on the validation dataset while still increasing on the training data, indicating the model was starting to over-fit. This is illustrated by figures 4.3 and 4.4 below, which show the maximum micro-F_1 score across all feature sets for that window size. The training and validation micro-F_1 scores were computed using 5-fold cross-validation on the training dataset, averaging the scores across all of the folds.

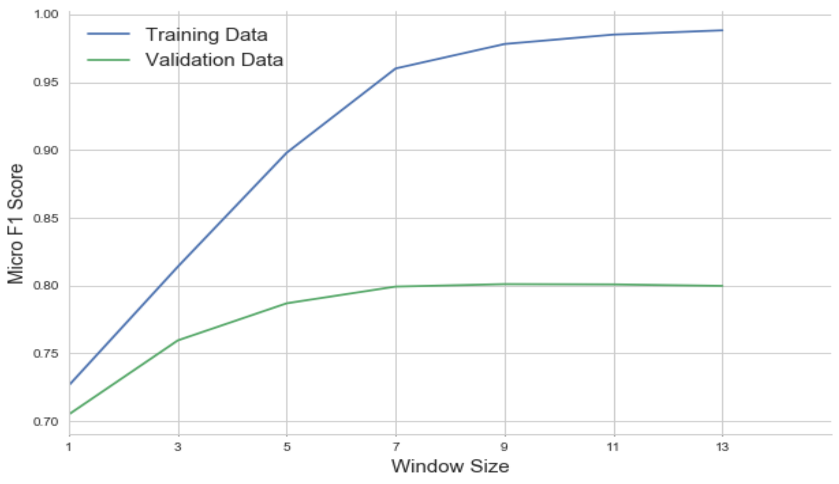


Figure 4.4: Maximum Micro-F_1 score By Window Size For Skin Cancer

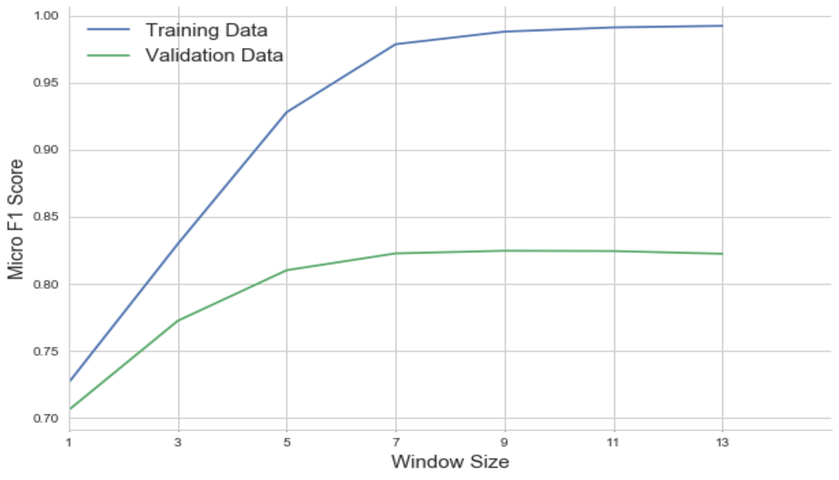


Figure 4.3: Maximum Micro-F_1 score By Window Size For Coral Bleaching

The number of features in each feature set for the optimal word window size of 9 are listed in Table 4.2. In general, the greater the ngram size, the larger the number of features, as can be seen with the BOW ngrams. However, for the positional variants, the rarity of the individual trigrams when combined with their window positions meant that many of those features were below the minimum frequency threshold of 5, resulting in fewer positional trigrams features than positional bigrams.

Table 4.2: Average Number of Features in Each Feature Set Across All Cross-Validation Runs (Window Size - 9 Words).

Feature Set	**Coral Bleaching**	**Skin Cancer**
BOW Unigrams	1,617.8	1,560.6
BOW Bigrams	20,957.4	19,481.4
BOW Trigrams	46,318.2	43,600.4
Positional Unigrams	8,305.0	7,878.8
Positional Bigrams	23,740.4	23,424.8
Positional Trigrams	18,619.8	20,119.2
Positional Stemmed Unigrams	6,486.6	6,363.4
Positional Stemmed Bigrams	23,263.6	23,344.4
Positional Stemmed Trigrams	18,941.6	20,337.0
BOW POS	40.6	41.6
Positional POS	321.8	323.0
Dependency Parsed Relations	4,848.0	4,711.2
Brown Cluster Labels	512.8	493.2
Total	173,973.6	171,679.0

The relative importance of each set of features can be evaluated by looking at the model's performance when trained on each set of features in isolation. Table 4.3 ranks the individual feature sets by micro-F_1 score.

Table 4.3: Each Feature Set Ranked by Micro-F_1 Score When Used in Isolation (Window Size - 9 Words).

	Coral Bleaching		**Skin Cancer**	
	Feature Set	F_1	**Feature Set**	F_1
1	Positional Stemmed Unigrams	0.813	Positional Stemmed Unigrams	0.791
2	Positional Unigrams	0.806	Positional Unigrams	0.788
3	Positional Stemmed Bigrams	0.770	Positional Stemmed Bigrams	0.762
4	Positional Bigrams	0.753	Positional Bigrams	0.754
5	Positional Stemmed Trigrams	0.702	Positional Stemmed Trigrams	0.697
6	Positional Trigrams	0.678	Positional Trigrams	0.684
7	BOW Bigrams	0.611	BOW Bigrams	0.602
8	BOW Unigrams	0.600	BOW Trigrams	0.577
9	BOW Trigrams	0.574	Dependency Parsed Relations	0.572
10	Dependency Parsed Relations	0.558	BOW Unigrams	0.567
11	Brown Cluster Labels	0.360	Positional POS	0.356
12	Positional POS	0.219	Brown Cluster Labels	0.324
13	BOW POS	0.060	BOW POS	0.102

The top 6 out of 13 feature sets for both datasets are positional variants of unigrams and ngrams, indicating that word position within the word window is very important for predicting the target word's tag, while the POS tags and Brown Cluster labels were amongst the least predictive feature sets when used in isolation. The stemmed unigram and bigrams also out-perform the unstemmed variants in both datasets. If we look at the micro-precision and micro-recall scores for the stemmed and the unstemmed positional unigram and bigram feature sets, we see that the gains in micro-F_1 score come from improvements in the micro-recall, with a minor change in micro-precision (see Tables 4.4 and 4.5 below). This suggests that the process of stemming allows the algorithm to generalize better by collapsing different words with the same stemmed form into the same tokens.

Table 4.4: Impact of Stemming on Classification Accuracy on the Coral Bleaching Validation Dataset

	Micro-F_1	**Micro-Recall**	**Micro-Precision**
Positional Unigrams	0.806	0.734	0.893
Positional Stemmed Unigrams	0.813	0.747	0.891
Percentage Improvement	0.9	1.8	-0.2
Positional Bigrams	0.753	0.643	0.908
Positional Stemmed Bigrams	0.770	0.666	0.911
Percentage Improvement	2.3	3.6	0.3

Table 4.5: Impact of Stemming on Classification Accuracy on the Skin Cancer Validation Dataset

	Micro-F_1	**Micro-Recall**	**Micro-Precision**
Positional Unigrams	0.788	0.733	0.853
Positional Stemmed Unigrams	0.791	0.740	0.850
Percentage Improvement	0.4	0.1	-0.4
Positional Bigrams	0.754	0.667	0.868
Positional Stemmed Bigrams	0.762	0.679	0.868
Percentage Improvement	1.1	1.8	0.0

However the purpose of using model-based feature selection is to evaluate the accuracy of the model when used with combinations of feature sets. While some feature sets may not be very useful for building a classifier on their own, they may improve the performance of the model more noticeably when used in combination with other feature sets. Table 4.6 shows how the micro-F_1 score increases as new feature sets are added by forward-selection. Each row shows the new feature set that was added in that iteration, and the resulting micro F_1 score. The second most predictive feature sets are positional stemmed bigrams for both datasets. When the feature sets were evaluated in isolation, the positional unigrams were the second most predictive feature set, as seen in Table 4.3. The stemmed and un-stemmed positional unigrams are highly correlated, and so once the stemmed unigrams are added to the model, the un-stemmed version adds little new discriminative information, and so the bigrams are chosen instead by forward selection.

Table 4.6: Micro F_1 Score as Additional Feature Sets Are Added by Forward Selection (Window Size - 9 Words). Pos. = Positional, Dep. = Dependency

	Coral Bleaching		**Skin Cancer**	
# Feats.	**Added Feature Set**	F_1	**Added Feature Set**	F_1
1	Pos. Stemmed Unigrams	0.8127	Pos. Stemmed Unigrams	0.7911
2	Pos. Stemmed Bigrams	0.8191	Pos. Stemmed Bigrams	0.7991
3	BOW Unigrams	0.8227	Brown Cluster Labels	0.8009
4	Brown Cluster Labels	0.8236	BOW Unigrams	**0.8011**
5	Pos. Stemmed Trigrams	0.8242	Dep. Parsed Relations	0.8009
6	Dep. Parsed Relations	**0.8247**	Pos. Unigrams	0.8003

To reduce computational complexity, the maximum combination of feature sets evaluated was 6. For the coral bleaching essays, the model's validation accuracy continued to increase by smaller amounts as more feature sets were added, but in the skin cancer dataset the algorithm started to overfit once more than 4 feature sets were added, and no trigram feature sets were selected. This can be seen in figure 4.5, which shows how the maximum F_1 score changes as more feature sets are added. The Brown cluster labels were selected in both datasets despite producing very low micro-F_1 scores when used in isolation. This would indicate that these feature sets differed sufficiently from the other feature sets to add some predictive power. However the improvements in F_1 scores in each dataset after the initial positional stemmed unigram feature sets were added was minimal, suggesting that the relative positions of the individual words within the context of the target word was most important in determining its associated concept code or codes.

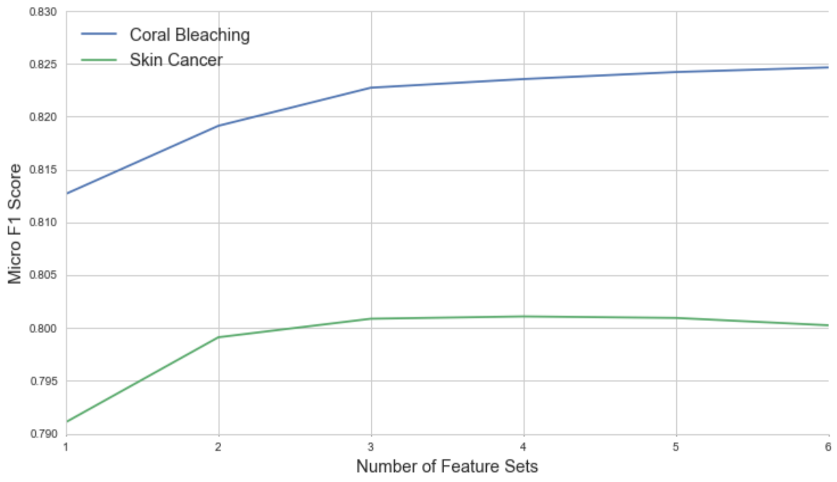


Figure 4.5: Maximum Micro-F_1 Score by Number of Feature Sets (Window Size - 9 Words)

Table 4.7 below shows the reduction in the total number of features as a result of feature selection.

Table 4.7: The Reduction in Number of Features for the Window-Based Tagging Model as a Result of Feature Selection. Window Size = 9 Words

Dataset	**# All Features**	**# Optimal Features**	**% Reduction**
Coral Bleaching	173,974	55,670	68%
Skin Cancer	171,679	44,352	74%

It should be noted that only 3 of the 5 model types evaluated used this set of features. The Window-Based Word Tagging Model, the CRF model and the Structured Perceptron all make use of a word window around the target word to be tagged, and are able to utilize this set of pre-determined features. However, the Hidden Markov Model is a probabilistic graphical model and is not designed to use arbitrary features computed over the word sequence. Furthermore, the Recurrent Neural Network is able to derive its own features over the entire word sequence it is trained on, and is not restricted to making use of a word window in order to take into account the target word's context.

Different features were therefore used for the Recurrent Neural Network model, as discussed in section 4.7.5.

4.6 Problem Transformation Methods

In both sets of essays, multiple concept codes can be associated with individual words. This is an example of a multi-label classification problem (MLC), as outlined in section 3.3. For the word labeling task, problem transformation methods were used to address the MLC problem because they allow existing machine learning models to be adapted to solve the MLC problem. Binary relevance (BR) is a simple problem transformation method that is computationally efficient and has been shown by multiple authors to be effective at solving multi-label classification problems (for example [106], [135]). The main limitation of BR is that the method is unable to take advantage of dependencies between labels [136], so its performance was compared to to that of the label powerset method (LP), which does not suffer from this limitation. The principle difference between these two approaches is that for BR, a separate classifier is trained for each concept code to be predicted, while in the LP approach, a separate classifier is trained for each unique combination of tags that were observed in the training data. At prediction time for the LP approach, each word is assigned the set of tags taken as the union of all predicted tag sets for that word. For more details on these two different problem transformation techniques, please refer to section 3.3.1.

To determine which problem transformation method was the most effective at creating a word tagging model, the performance of both the BR and LP approaches were evaluated using the optimal set of features and window-sizes determined by the feature selection process. However, as described in section 2.4, very few words in each dataset have multiple tags, so a problem transformation method is not going to have a large impact on the accuracy of the model. To evaluate the relative effectiveness of problem transformation methods on this problem, a third method was used, which I will call 'most common tag'. In the most common tag approach, the training data was modified such that all words assigned multiple tags were assigned only the most common tag, based on tag frequencies computed from the training dataset. The trained model was then evaluated on the 20% test set partition described earlier (see section 4.2), which retained any instances of multiple tags. The results are shown in Table 4.8 below:

Table 4.8: Problem Transformation Method Accuracy (Micro F_1 score)

Method	**Coral Bleaching F_1**	**Skin Cancer F_1**
Most Common Tag	0.8317	0.8087
Label Powerset	0.8316	0.8087
Binary Relevance	0.8247	0.8011

The most common tag approach slightly out-performed the LP method on the coral bleaching dataset, and achieved the same micro-F_1 score on the skin cancer dataset. It appears that their were too few occurrences of each label powerset with multiple labels for the model to make use of dependencies between different labels to improve the classification accuracy. Because the most common tag approach is a simpler approach that does not require a modification to the underlying machine learning algorithm, this method was chosen to train the five different machine learning models.

4.7 Model Evaluation

Five different machine learning algorithms were evaluated on the word tagging problem to determine the most accurate algorithm for this task:

- Window-Based Word Tagging Model
- Conditional Random Field
- Hidden Markov Model
- Structured Perceptron
- Recurrent Neural Network

The choice of feature sets used in each model was informed by the results of the feature selection process, and each model was trained using the most common tag method described in section 4.6. The micro-F_1 score of each model was computed using 5-fold cross validation, and computed on both the training and tests folds for comparison. Hyper-parameter tuning was conducted on each model and the hyper-parameter settings were chosen which maximized the micro-F_1 score on the test folds. Each model was then re-trained on the entire training dataset using the optimal settings to evaluate its performance on the separate test dataset.

4.7.1 Window-Based Word Tagging Model

The window based word tagging model uses a sliding window of words around the target word to train a machine learning classification algorithm to classify each word based on its context, as described in section 3.4.2.3. A window size of 9 words was used to train the window-based classifier using the optimal feature set determined by the feature selection exercise described in section 4.5, and the most common tag approach, detailed in section 4.6. The logistic regression classifier implementation from the *LibLinear* package [65] was used as the classification algorithm.

When training a machine learning model on a classification or regression problem, most algorithms provide one or more hyper-parameters designed to control the complexity of the resulting model. This concept is referred to as 'regularization' and is used to help prevent overfitting by encouraging the discovery of simpler models over more complex ones. The assumption here is that simpler models are more likely to better explain the data, based on the principles of Occam's razor. Three different methods of regularization are available for the logistic regression algorithm - L1, L2 and dual mode, which combines both constraints (see [65] for further details). Each of these methods constrains the size of the coefficients learned by the model to reduce the variance in the model. For a model with m coefficient values, w_i, L1 regularization penalizes the sum of the absolute values of the coefficients (see Equation 4.12), while L2 regularization penalizes the sum of the squared coefficients (see Equation 4.13) [7]:

$$\lambda \|w\|_1 = \lambda \sum_{i=1}^{m} |w_i| \tag{4.12}$$

$$\frac{\lambda}{2} \|w\|_2 = \frac{\lambda}{2} \sum_{i=1}^{m} w_i^2 \tag{4.13}$$

In Equations 4.12 and 4.13, the constant term λ is a hyper-parameter controlling the degree of regularization. In the *LibLinear* package, the degree of regularization is controlled by the C parameter which is described as the 'inverse of regularization strength', i.e. $C = 1/\lambda$ [167], and thus has a similar purpose to the C parameter used to regularize Support Vector Machines. Low C values lead to stronger regularization by encouraging smaller coefficient values. A grid search was performed to determine the optimum regularization method and corresponding C value. C values of 0.1, 0.5, 1, 10, and 100 were evaluated, and dual mode with a C value of 0.5 produced the best micro-F_1 score on the validation data in cross validation on both datasets. The micro F_1 scores attained on the training, validation and test datasets using these settings are presented in in Tables 4.9 and 4.10.

Table 4.9: Window-Based Tagging Model Classification Accuracy on the Coral Bleaching Dataset

Data Partition	**Micro-F_1**	**Micro-Recall**	**Micro-Precision**
Training Data (CV)	0.940	0.918	0.963
Validation Data (CV)	0.832	0.781	0.889
Test Data	0.842	0.802	0.885

Table 4.10: Window-Based Tagging Model Classification Accuracy on the Skin Cancer Dataset

Data Partition	**Micro-F_1**	**Micro-Recall**	**Micro-Precision**
Training Data (CV)	0.901	0.878	0.924
Validation Data (CV)	0.810	0.773	0.850
Test Data	0.814	0.779	0.853

In both datasets, the micro-F_1 scores are higher on the test dataset than the validation dataset. This reflects the fact that the entire training dataset was used to train the model that was evaluated on the test dataset, whereas only 80% of the training data was used to train the models in each fold during 5-fold cross validation. The micro-precision scores are higher than the micro-recall scores for each dataset and each partition. One of the biggest challenges in solving any NLP machine learning problem is handling out-of-vocabulary words (OOV) - words seen in the test or validation data that were not present in the training data. In the coral bleaching dataset, on average 2.1% of the words in the validation data and 3.9% of the words in the test dataset were not found in the training data set, while in the skin cancer dataset 1.9% of the validation data and 4.6% of the test data set words were OOV. This is therefore one possible explanation for the higher precision scores. This problem can be addressed in part by stemming, which collapses multiple morphological forms of a word to the same stem, but stemming will not recognize words that have the same or similar meanings but different stems.

The window-based tagging model attains a higher micro-F_1 score on the coral bleaching dataset than the skin cancer dataset. Tables 2.2 and 2.3 lists a number of key statistics describing the two different datasets. On average, the skin cancer essays were longer, contained longer sentences and more unique words, and thus represent a more complex set of documents for classification, which may explain this difference in classification performance.

4.7.2 Conditional Random Field

The Conditional Random Field (CRF) algorithm is a linear chain probabilistic graphical model which expresses the relationships between random variables using a graph structure that forms a simple linear chain. Unlike the window-based tagging model, the CRF algorithm is able to learn from both the current observations and the tags assigned to the previous words in the sentence. For more details about the CRF algorithm, please see section 3.4.2.1. The CRF algorithm was trained using the same feature sets as the window-based tagger, using a 9 word window and the optimal feature set from the feature selection exercise. The most common tag method was used to address the MLC problem. The CRF algorithm implementation from the *CRFSuite* package [162] was used, which is a first-order Markov CRF, and therefore looks only at one previous word's tag when making predictions. The CRF algorithm uses the *Viterbi* algorithm [235] as the 'decoder', to determine the most probable tagging sequence given a sequence of observations. The algorithm is trained with gradient descent using the L-BFGS method.

The CRF model was regularized using L2 regularization, and the following values were evaluated for the C parameter - 0.1, 0.5, 1, 10, and 100. A C coefficient value of 0.1 achieved the highest micro-F_1 score on the coral bleaching validation data, and a value of 0.9 on the skin cancer dataset. For each C value, the algorithm's performance was also evaluated with the *feature possible transitions* setting 'on' and 'off'. When set to 'on', this generates additional negative transition features that do not occur in the training data to associate all possible label pairs. Setting this to 'on' slightly improved the micro-F_1 score on the skin cancer dataset, but decreased this metric very slightly on the coral bleaching dataset. This indicates that the CRF model tended to over-fit the additional features generated by this setting on the coral bleaching but not the skin cancer dataset. The micro-average F_1 score, precision and recall of this model on the different data partitions are listed in Tables 4.11 and 4.12.

Table 4.11: CRF Classification Accuracy on the Coral Bleaching Dataset

Data Partition	**Micro-F_1**	**Micro-Recall**	**Micro-Precision**
Training Data (CV)	0.994	0.994	0.994
Validation Data (CV)	0.824	0.772	0.883
Test Data	0.835	0.797	0.878

Table 4.12: CRF Classification Accuracy on the Skin Cancer Dataset

Data Partition	**Micro-F_1**	**Micro-Recall**	**Micro-Precision**
Training Data (CV)	0.971	0.966	0.977
Validation Data (CV)	0.799	0.758	0.846
Test Data	0.804	0.759	0.855

In comparison to the window-based tagging model, for both datasets the CRF model achieves a higher micro-F_1 score on the training data, and a lower score on both the validation and test datasets. This suggests the model was more prone to overfitting the data, and was not able to use the previous word tags to improve classification performance. As with the window-based tagging model, precision was higher than recall for each dataset, and the model again performed slightly better on the test dataset than the validation dataset.

4.7.3 Hidden Markov Model

The Hidden Markov Model (HMM) is also a linear chain probabilistic graphical model which predicts the joint probability of observing each tag with the current word. The HMM is a probabilistic graphical model which creates a generative rather than a discriminative model of the data. Generative models attempt to directly model the data generation process itself, rather than predicting the most probable class given a set of arbitrary features. Causal inference is then performed by interrogating the generative model. Because the HMM is a generative and not a discriminative model, it is only able to make use of information available to the graphical model, and cannot make use of arbitrary features of the data. It is therefore unable to make use of information about words that occurred earlier or later in the sentence [54], and can only use the current word, in addition to the previous tag, to make a prediction. Consequently, the HMM model was unable to use the same feature sets as the previous two models, and instead was evaluated using unstemmed and stemmed words as the observations. In both datasets, the HMM achieved a higher micro-F_1 score when using the stemmed words.

Maximum Entropy Markov Models (MEMM's) extend the HMM to make use of additional features over the inputs. However, MEMMs suffer from the label-bias problem (see section 3.4.2.1) and are typically out-performed by the CRF algorithm [54, 129]. For more information about HMMs, please refer to section 3.4.2.1.

The HMM implementation found in the *NLTK* [6] software package was used for this task. This implementation also uses the *Viterbi* algorithm [235] to determine the most probable tagging sequence given a sequence of observations. The principle hyper-parameter in an HMM model are the number of hidden states in the model [150], and the method of smoothing the probabilities to better handle low frequency or previously unseen words. Because the hidden states in the HMM for this task correspond to the concept codes, the number of states is fixed and cannot be optimized. In addition, because infrequently occurring words and previously unseen words were replaced with the special *INFREQUENT* token, as described in section 4.4, then smoothing was unnecessary.

The performance of the HMM model using the stemmed words across the different data partitions and data sets is listed below in Tables 4.13 and 4.14.

Table 4.13: HMM Classification Accuracy on the Coral Bleaching Dataset

Data Partition	**Micro-F_1**	**Micro-Recall**	**Micro-Precision**
Training Data (CV)	0.799	0.850	0.754
Validation Data (CV)	0.758	0.789	0.728
Test Data	0.747	0.799	0.702

Table 4.14: HMM Classification Accuracy on the Skin Cancer Dataset

Data Partition	**Micro-F_1**	**Micro-Recall**	**Micro-Precision**
Training Data (CV)	0.696	0.769	0.645
Validation Data (CV)	0.674	0.733	0.625
Test Data	0.675	0.731	0.628

The HMM under-performed the other models on both datasets as it was not able to make use of the other contextual information from the word window. Unlike the previous two models, the HMM achieved higher recall than precision for each dataset and each partition. Because the model was only using stemmed words as the features from the sentence and could not make use of rarer, more specific features (such as the positional unigrams and ngrams) this likely resulted in the HMM making broader generalizations over the training data. This would lower the false negative rate but increase the false positive rate, therefore improving recall at the cost of precision. The difference in the training and test accuracies for the HMM was also much less than the other model types, indicating this model was less prone to overfitting, likely as a result of the simpler set of features it was able to learn from.

4.7.4 Structured Perceptron

A key problem in building structured prediction models that rely on their previous predictions is deciding how to choose the previous labels used to train the system (see section 3.4.1). One approach is to use the 'ground truth' - the actual labels assigned to the training data to condition future tagging decisions on, which is how the HMM and CRF models are trained. However, it is usually better to train a model using its own noisier predictions, because this is how it will make predictions on new datapoints that it hasn't seen before [45]. To do so requires an online machine learning algorithm that can learn iteratively, such as the structured perceptron.

The structured perceptron is a modification of Rosenblatt's Perceptron algorithm for use in structured prediction problems [35]. In contrast to the CRF and HMM models, a greedy decoding algorithm was used to train the structured perceptron. Whereas the *Viterbi* algorithm uses dynamic programming to search for the most probable sequence of tags for a sentence, a greedy algorithm simply chooses the most probable tag to assign to each word in the sentence. The Viterbi algorithm is more likely, but not guaranteed, to find a more optimal solution as a deeper search is conducted. However, the computational complexity of the *Viterbi* algorithm can result in very long training times for an online algorithm with so many features and a large number of different tags, making it impractical for this problem. For more information on the structured perceptron, please refer to sub-section 3.4.5.

The greedy averaged perceptron implementation from the *SpaCy* software package [103] was used with modifications to support the optimal feature set from section 4.5. The averaged perceptron is a modification of the perceptron where the model's weights are replaced with their average values, computed over the course of training the model [35]. Averaging the model's weights following training has been shown to improve the model's accuracy on the test dataset on a number of different tagging problems [35]. For each dataset, the model was trained with and without weight averaging, and with and without using the previous predicted label as an additional feature. To help prevent overfitting, 'early-stopping' was used to ensure that training was halted once the algorithm's performance on a separate held-out dataset started to decline [150]. To achieve this, during 5-fold cross-validation, each training data fold created in cross-validation was further split into two partitions - 80% was used to train the perceptron, and the remaining 20% of the fold's data was used to determine when to stop training the algorithm. The perceptron was then trained iteratively until the performance on the held out dataset started to decline, at which point the number of training iterations was noted. The algorithm was then trained from scratch on the entire training data fold for the number of iterations that resulted in the optimal classification accuracy on the held out dataset in the initial run.

In both datasets, the algorithm had the highest micro F_1 score when weight averaging was applied, and the previous predicted label was included in the features. The model's accuracy across the 2 datasets and the different data partitions are listed below in Tables 4.15 and 4.16.

Table 4.15: Structured Perceptron Classification Accuracy on the Coral Bleaching Dataset

Data Partition	**Micro-F_1**	**Micro-Recall**	**Micro-Precision**
Training Data (CV)	0.948	0.930	0.966
Validation Data (CV)	0.829	0.778	0.887
Test Data	0.837	0.794	0.884

Table 4.16: Structured Perceptron Classification Accuracy on the Skin Cancer Dataset

Data Partition	**Micro-F_1**	**Micro-Recall**	**Micro-Precision**
Training Data (CV)	0.913	0.892	0.935
Validation Data (CV)	0.809	0.767	0.845
Test Data	0.814	0.773	0.860

While the structured perceptron was less accurate than the window-based tagging model on the coral bleaching dataset, it achieved the same micro-F_1 score on the skin cancer dataset, and it out-performed the CRF and HMM models on both datasets.

4.7.5 Recurrent Neural Network

A bidirectional Long Short-Term Memory (LSTM) is a form of RNN that is able to learn long-distance relationships from sequential data by maintaining an internal state across multiple time steps, and by processing sequential data in both a forward and backward direction, as described in section 3.5.4. Bidirectional LSTM's have been shown by a number of authors to attain at or near state-of-the-art performance on a number of word tagging tasks, including part-of-speech tagging [172, 237], chunking and named entity recognition [237]. A more recent variant of the LSTM, called the Gated Recurrent Unit (GRU) was proposed by Cho et al. in 2014 [28] and has been shown to have comparable performance on sequence labeling tasks, while being more computationally efficient [30]. Inspired by this work, a bidirectional GRU RNN was trained on the word labeling task.

A bidirectional RNN can automatically utilize information from the context of the word to be tagged due to the recurrent connections within the hidden layer. Consequently, no explicit word-window was used to train the recurrent neural network, and the RNN was fed a sentence at a time, with its state reset between sentences. The RNN was trained using a word embedding layer (see sub-section 3.5.1 for more information), which enables the network to learn a vector representation for each word by transferring knowledge from the language model used to train it. The use of pre-trained word embeddings in place of discrete representations of words has been shown to improve the performance of neural networks on a number of NLP tasks [38, 39]. 100-dimensional GloVe vectors were used for the pre-trained embeddings, and produced by the *GloVe* algorithm [168] trained on 1.6 billion tokens from a 2014 copy of the English Wikipedia. These vectors can be downloaded from [88]. Amongst the many different topics covered, Wikipedia contains scientific content and as such should be relevant to the essay topics studied.

The general architecture of the RNN consisted of a word-embedding layer, followed by one or two recurrent layers and finally a softmax layer that computed a probability distribution over all possible tags for each individual word to be tagged. Cross-entropy was used as the loss function, as is typical for classification problems, and the pre-trained word embeddings were updated by back-propagation as the network was trained (i.e. they were not fixed). The *Adam* algorithm for stochastic optimization was used to perform gradient descent because it has been shown to converge faster and result in lower classification errors on a number of different datasets [118]. To prevent overfitting, 10% of the training data was kept aside as a development dataset and was used to perform early-stopping [150]. This thus acts as a form of regularization, halting training once the network appears to overfit on the development dataset. The performance of the network was evaluated on the development dataset after every epoch (each pass through the training data) and training was halted once 3 successive epochs were completed without any further improvement in the maximum micro-F_1 score attained so far. The network's weights were then reset to the weights that achieved the best micro-F_1 score on the development data. Because the development dataset was taken from the training dataset, the dataset partitions did not change and remained the same as those used in the other experiments. The training data metrics were calculated using the original training dataset, which included the development data.

Dropout [215] has become a very popular form of regularization for training deep neural networks, and involves randomly setting a proportion (usually 50%) of the activations of each network layer to 0. However, applying *Dropout* when training the RNN's used in this task caused problems with convergence in the initial experiments, and was not used in the final set of experimental runs. I suspect this is due to the small size of the dataset and the infrequency of the class labels. As some classes only appear with a small number of words, randomly ignoring those words while training makes it hard for the neural network to learn on these datasets.

To determine the best network configuration, 5-fold cross validation was performed with a grid search to evaluate a number of different configurations. The model was trained both with and without the pre-trained embeddings. When the pre-trained embeddings weren't used, the 100 dimensional word embedding vectors were randomly initialized instead using a uniform distribution. A one-directional GRU RNN was also evaluated in addition to the bidirectional model, and the model was trained with both one and two RNN layers. Finally, three different sizes of RNN hidden layer were evaluated with 64, 128 and 256 neurons. Training the neural network with more hidden nodes or more than 2 layers proved too computationally intensive given the available resources. One of the most important hyper-parameters to tune when training a neural-network is the learning rate because this can have a significant impact on how well the final model converges, and how likely it is to get stuck in local minima. However, one of the main advantages of the *Adam* optimization algorithm is that it dynamically adjusts the learning rate for each of the network's trainable parameters during training to optimize performance [118]. It is therefore unnecessary to tune the learning-rate when this optimization algorithm is used.

The optimal configuration on the validation data set used pre-trained word embeddings, 2 bidirectional GRU RNN layers with 256 hidden units. The micro-average precision, recall and F_1 scores of the RNN with the optimal configuration on the different data partitions are listed in Tables 4.17 and 4.18 below.

Table 4.17: Recurrent Neural Network Classification Accuracy on the Coral Bleaching Dataset

Data Partition	**Micro-F_1**	**Micro-Recall**	**Micro-Precision**
Training Data (CV)	0.927	0.925	0.928
Validation Data (CV)	0.837	0.822	0.853
Test Data	0.842	0.830	0.855

Table 4.18: Recurrent Neural Network Classification Accuracy on the Skin Cancer Dataset

Data Partition	**Micro-F_1**	**Micro-Recall**	**Micro-Precision**
Training Data (CV)	0.918	0.925	0.911
Validation Data (CV)	0.821	0.821	0.822
Test Data	0.837	0.807	0.869

The bidirectional RNN attained the same micro-F_1 score as the window-based classifier on the coral bleaching dataset, and out-performed all of the other algorithms on the skin cancer dataset. Whereas the RNN achieved the best recall, precision and micro-F_1 score on the skin cancer dataset, the higher micro-F_1 score on the coral bleaching dataset was as a result of a higher recall, because the precision was lower than that of the window-based classifier. The pre-trained word embeddings allow the model to incorporate information from a different and much larger corpus by learning a distributed representation for the meaning of each word. This can be considered a form of transfer learning, where information is transferred between the language model, used to train the embeddings, and the RNN model utilizing those embeddings. If we compare the model's best performance both with and without the word embeddings, we see that most of the improvement in the micro-F_1 score comes from the improvement in the micro-recall and not the micro-precision, as can be seen in Tables 4.19 and 4.20. There is around a 3% improvement in recall on the test datasets when the embeddings are used, but only around a 1% improvement in precision. This suggests that word embeddings allow the model to generalize better to cover more unique words by providing a distributed representation of the meaning of a word, where similar words possess similar representations.

Table 4.19: Impact of Pre-Trained Word Embeddings on RNN Classification Accuracy on the Coral Bleaching Validation Dataset

	Micro-F_1	**Micro-Recall**	**Micro-Precision**
Without embeddings	0.818	0.795	0.844
With embeddings	0.837	0.822	0.853
Percentage Improvement	2.3	3.4	1.1

Table 4.20: Impact of Pre-Trained Word Embeddings on RNN Classification Accuracy on the Skin Cancer Validation Dataset

	Micro-F_1	**Micro-Recall**	**Micro-Precision**
Without embeddings	0.806	0.798	0.814
With embeddings	0.821	0.821	0.822
Percentage Improvement	1.9	2.9	1.0

Without using the pre-trained word-embeddings, the bidirectional RNN would have only attained the 4th best micro-F_1 score on the coral bleaching dataset and the 3rd best micro-F_1 score on the skin cancer dataset, when comparing the algorithm's cross validation performance on the validation data folds. The main advantage the RNN has on this dataset therefore appears to be its ability to make use of pre-trained word-embeddings, and fine tune them to the specific classification task. Deep neural networks typically require large amounts of data to perform well on most tasks, which may explain the relatively poor performance of this model on this task without the pre-trained word embeddings.

The bidirectional RNN also performed much better than the regular RNN on this task, as shown Tables 4.21 and 4.22. This suggests that the context to the right of the word is also important in determining its concept code, and had an even greater impact than the use of pre-trained word embeddings.

Table 4.21: Impact of Using a Bidirectional RNN on Classification Accuracy on the Coral Bleaching Validation Dataset

	Micro-F_1	**Micro-Recall**	**Micro-Precision**
Unidirectional	0.760	0.735	0.787
Bidirectional	0.837	0.822	0.853
Percentage Improvement	10.1	11.8	8.4

Table 4.22: Impact of Using a Bidirectional RNN on Classification Accuracy on the Skin Cancer Validation Dataset

	Micro-F_1	**Micro-Recall**	**Micro-Precision**
Unidirectional	0.762	0.747	0.778
Bidirectional	0.821	0.821	0.822
Percentage Improvement	7.7	9.9	5.7

4.8 Summary of Results and Discussion

The performance of the 5 different approaches on the datasets can be analyzed in a number of different ways to throw light on the differences in the individual algorithms. In addition to comparing their performance using the micro-F_1 score, I will also compare the algorithms' performance using the macro-F_1

score, as well as showing how each model performs on each individual concept code. This will illuminate how sensitive each model is to the relative frequency of each code, and whether some codes are inherently more complex to learn than others. Finally, I will run a set of statistical tests to see if there is a statistically significant difference in performance between the 5 models, and then discuss how extensible these results are to other texts and other domains.

4.8.1 Micro-F_1 Performance

The classification accuracy of each algorithm on the test data can be seen in summary Tables 4.23 and 4.24 below.

Table 4.23: Test Data Accuracy by Algorithm on the Coral Bleaching Dataset. All metrics listed are micro metrics.

Algorithm	F_1	**Recall**	**Precision**	**Features**
Window-Based Tagger	**0.842**	0.802	**0.885**	Word Window Feats.
CRF	0.835	0.797	0.878	Word Window Feats.
HMM	0.747	0.799	0.702	Stemmed Unigrams
Structured Perceptron	0.837	0.794	0.884	Word Window Feats.
Bidirectional RNN	**0.842**	**0.830**	0.855	Word Embeddings

Table 4.24: Test Data Accuracy by Algorithm on the Skin Cancer Dataset. All metrics listed are micro metrics.

Algorithm	F_1	**Recall**	**Precision**	**Features**
Window-Based Tagger	0.814	0.779	0.853	Word Window Feats.
CRF	0.804	0.759	0.855	Word Window Feats.
HMM	0.675	0.731	0.628	Stemmed Unigrams
Structured Perceptron	0.814	0.773	0.860	Word Window Feats.
Bidirectional RNN	**0.837**	**0.807**	**0.869**	Word Embeddings

Overall, the bidirectional RNN and window-based tagging model both achieved the optimal micro-F_1 score on the coral bleaching dataset (when rounded to 3 decimal places), while the RNN achieved the best micro-F_1 score out of all of the models on the skin cancer dataset, with a micro-F_1 score

2.8% higher than the perceptron and window-based tagger on that dataset. The RNN model had the strongest overall performance across both dataset, as expected. All models, with the exception of the structured perceptron, achieved a higher micro-F_1 score on the coral bleaching dataset, suggesting this dataset was slightly easier to learn from. Similarly, all of the models except the HMM model achieved higher precision than recall on both datasets. This suggests one of the challenges with this problem is making predictions on rare words and words not seen in the training data.

The strong performance of the window-based tagging model on both datasets compared to the CRF, HMM and structured perceptron models suggests that the surrounding words, and not their concept codes, are the most important factor in predicting a word's concept code. This is because the window-based tagging model did not make use of previously predicted concept codes, which the other models were able to do. Furthermore, the HMM model was the only model not able to use information about the surrounding words, and achieved the lowest classification accuracy out of all of the different models. However, the window-based tagging model was used to perform the feature selection, and so may have gained some advantage over the other three models that used the same set of features chosen by that process. This may also explain it's superior performance when compared to those other three models. The superior performance of the bidirectional RNN over the regular RNN also suggests that the words present on the right side of the target word are important in predicting the word's concept code, not just the words to the left of the target word.

The results of the feature selection exercise also indicated that the relative position of the surrounding words in the word window was also very important, as the positional ngrams out-performed the bag-of-word ngrams by a significant amount (see Table 4.3 and Table 4.6). The main advantage of using a bidirectional RNN for this task is that it can use the context of the entire sentence in order to predict the word tags, while also taking into account the relative order of those words. The window-based tagging approach, however, can only use words found within the word window and is unable to make use of words found elsewhere in the sentence. This capability, combined with the ability to make use of the pre-trained word embeddings could explain the superior performance of the bidirectional RNN at this task. The bidirectional RNN comfortably out-performed the other approaches on the skin cancer dataset, indicating that these aspects of the algorithm were particularly useful on that dataset.

In the feature selection exercise the stemmed ngrams out-performed the unstemmed ngrams due to improvements in recall. Similarly, using pre-trained word embeddings in the bidirectional RNN improved the micro-F_1 score primarily by improving the recall of the model. By improving the recall, these features allowed the model to generalize better to cover more positive examples of each code. Also, the poor performance of the part-of-speech tags and dependency parsed relations as features indicates that the meaning of the words in the sentence mattered much more than the grammatical structure of the sentence when predicting the concept codes present.

When trying to solve a structured prediction problem, one important question to ask is 'Is it more effective to use the ground-truth labels or the algorithm's own predictions to condition future predictions on?'. Both the structured perceptron model and the CRF algorithm were trained using the same set of features, and both algorithms are able to make use of the previous word's label when predicting the target word's label. One important difference between these two algorithms was the way in which they were trained. The structured perceptron was trained using its own predicted labels, whereas as the CRF model was trained using the ground-truth labels from the training data. The structured perceptron out-performed the CRF model on both datasets, which suggests that training an algorithm on its own predictions is a more effective approach for this type of problem.

Each of the five algorithms achieved higher micro-F_1 scores on the coral bleaching dataset. Both datasets have quite different characteristics, as described in sections 2.4. There are fewer concept codes in the skin cancer dataset, 9 compared to 13 codes, the total word count for the entire dataset is smaller for the skin cancer corpus, but the skin cancer essays are longer and use more unique words per essay. The skin cancer essays also have more concept codes per essay, on average, than the coral bleaching essays, while a greater percentage of words and sentences are assigned concept codes in the coral bleaching dataset. The longer essays, the greater variation in word usage, and the lower density of concept codes could explain why higher micro-F_1 scores were attained on the coral bleaching dataset.

The goal of solving Research Question 1 was to build an accurate word tagging model to detect the concepts that could then be used to construct an accurate causal model of each essay. The micro-F_1 metric was chosen as it takes into account the relative frequencies of each concept code when computing an F_1 score over all of the concept codes; the more common codes have a larger impact on the micro-F_1 score than the less frequent codes. This is necessary to optimize the accuracy of the final causal model construction. However, if we are more interested in analyzing the performance of the different algorithms at solving the tagging problem, it is important to understand how well the different approaches perform on rarer classes. The macro-F_1 score computes an unweighted average over all classes, and is better suited to answering this problem by treating each class equally.

4.8.2 Macro F_1 Performance

Tables 4.25 and 4.26 below shows the macro-F_1 score of each algorithm on the test.

Table 4.25: Macro Test Data Metrics by Algorithm on the Coral Bleaching Dataset

Algorithm	**Macro-F_1**	**Macro-Recall**	**Macro-Precision**
Window-Based Tagger	0.740	0.689	**0.800**
CRF	0.725	0.676	0.781
HMM	0.657	0.725	0.602
Structured Perceptron	0.737	0.691	0.789
Bidirectional RNN	**0.769**	**0.756**	0.783

Table 4.26: Macro Test Data Metrics by Algorithm on the Skin Cancer Dataset

Algorithm	**Macro-F_1**	**Macro-Recall**	**Macro-Precision**
Window-Based Tagger	0.761	0.693	0.843
CRF	0.756	0.685	0.843
HMM	0.644	0.678	0.613
Structured Perceptron	0.757	0.690	0.840
Bidirectional RNN	**0.779**	**0.711**	**0.862**

Based on the macro-F_1 score, the RNN model is the most accurate model on both datasets, which is in agreement with the the micro-F_1 metric results. The window-based tagging model again had the highest precision on the coral bleaching data, but the recall was much lower than the RNN resulting in a lower macro-F_1 score overall. For both the micro and macro-F_1 metrics, when combining results from both datasets, the RNN is the most accurate model, followed by the window-based tagger and the structured perceptron, with the CRF and the HMM models having the weakest performance on the word labeling task.

4.8.3 Performance By Individual Concept Code

Tables 4.27 and 4.28 show the individual F_1 scores for each algorithm across all concept codes for each dataset. Appendix E provides a more detailed breakdown of the accuracy metrics by code for each of the 5 algorithms across both datasets.

Table 4.27: Test Data F_1 Score by Algorithm by Code on the Coral Bleaching Dataset

Code	**Percentage of-Words**	**Window-Based Tagger**	**CRF**	**HMM**	**Structured Perceptron**	**Bidirectional RNN**
1	3.32%	0.826	0.836	0.784	**0.839**	0.819
2	0.85%	0.740	0.762	0.441	**0.783**	0.712
3	5.36%	**0.827**	0.820	0.733	0.808	**0.827**
4	1.89%	0.832	0.793	0.708	0.827	**0.845**
5	1.45%	0.449	0.319	0.152	0.446	**0.587**
5b	1.45%	0.030	0.000	0.073	0.031	**0.308**
6	0.88%	**0.836**	0.834	0.821	**0.836**	0.833
7	3.03%	0.838	0.826	0.731	0.820	**0.840**
11	0.63%	0.899	**0.927**	0.869	0.893	0.909
12	0.51%	0.863	0.863	0.698	0.882	**0.939**
13	1.34%	0.734	0.729	0.702	0.717	**0.753**
14	1.47%	**0.748**	0.700	0.715	0.731	0.687
50	8.86%	0.904	0.900	0.871	0.901	**0.908**

Table 4.28: Test Data F_1 Score by Algorithm by Code on the Skin Cancer Dataset

Code	Percentage of-Words	Window-Based Tagger	CRF	HMM	Structured Perceptron	Bidirectional RNN
1	3.38%	0.826	0.801	0.712	0.813	**0.853**
2	3.33%	0.852	0.835	0.745	0.846	**0.853**
3	2.30%	0.835	0.792	0.725	0.828	**0.836**
4	1.86%	0.733	0.731	0.529	0.730	**0.750**
5	2.82%	0.852	0.851	0.802	0.854	**0.862**
6	2.88%	0.679	0.706	0.580	0.673	**0.751**
11	0.33%	0.621	0.660	0.607	0.604	**0.667**
12	0.52%	0.529	0.529	0.403	**0.557**	0.472
50	6.36%	0.836	0.831	0.641	0.845	**0.870**

The RNN more consistently out-performs the other models on the skin cancer dataset compared with the coral bleaching dataset, attaining the highest F_1 score for every concept code except code 12. In contrast, it only has the highest F_1 score on 7 out of 13 concept codes in the coral bleaching dataset. If we examine the highest F_1 scores for each concept code, we see a clear correlation between F_1 score and code frequency in the skin cancer dataset, but this is much less pronounced in the coral bleaching dataset. The Pearson correlation coefficient measures the linear correlation between two variables, or populations, resulting in a number between 1.0 and -1.0, values close to 1.0 have a strong positive correlation, values close to -1.0 have a strong negative correlation and values close to 0.0 are uncorrelated [191]. If we examine the correlation between F_1 score and concept code frequency, the skin cancer codes have a strong positive correlation of 0.77 while the coral bleaching codes have a weak positive correlation of only 0.22.

While is it not clear exactly why the RNN's performance was much more dominant on the skin cancer corpus, it appears that this dataset was more consistent across labels. In general, the more data points a machine learning algorithm has access to, the better it will generalize to new data points. The lower correlation between code frequency and F_1 score across all algorithms on the coral bleaching corpus implies much greater variability between concept codes, and some infrequent codes were much easier to classify than some of the more frequent codes. This could explain the higher macro-F_1 scores on the skin cancer dataset, which implies more consistency in labeling accuracy across different labels.

Recurrent neural networks have been shown to excel at some NLP transfer learning tasks, provided the different tasks are semantically related [251, 150]. The results described in section 4.6 indicated that treating the tagging problem as a multi-class classification problem, rather than a set of distinct binary classification problems was the better approach. Training the RNN to label all concept codes rather than each code individually allowed it to transfer knowledge learned from labeling one code to other codes. The RNN is the only algorithm evaluated that is able to share parameters from the model between different codes. In contrast, the other algorithms perform multi-class classification by training multiple binary classifiers, and selecting the class with the highest probability. It is possible that this transfer learning capability makes the RNN more effective on the skin cancer dataset, where there were fewer concept codes, and more consistency between the codes and the relative difficulty in labeling them.

4.8.4 Statistical Analysis

Although the results show that some algorithms appear to perform better than others at this task, an important question to ask is whether the differences in relative performance are statistically significant, i.e. could they have occurred simply due to chance? Following the approach outlined earlier this chapter in Section 4.3, I performed Cochran's Q Test [32] to test the null hypothesis, which states that there is no difference in the accuracy of each of the 5 algorithms studied. When the p-values are very small (e.g. below 0.001), it is customary to simply report the value as $p < 0.001$, and omit the remaining significant digits [69] as the result is obviously significant when the values are that small. For the Coral Bleaching dataset, a χ^2 value of 2002.8 was attained, while for the Skin Cancer dataset, a χ^2 value of 3915.2 was calculated. In each case, the p-value was 0.0, rejecting the null hypothesis which assumed there was no difference between classifiers. Since the null hypothesis was rejected, McNemar's test [144] was performed, comparing the performance of the Bidirectional RNN to the other four models, to determine if there was a significant difference between its accuracy on each dataset and that of the other models. Treating each dataset as independent, four comparisons where performed, meaning the significance threshold for each individual pairwise test is adjusted to 0.0125 (0.05/4) using the Bonferroni correction. The results of McNemar's test are shown below in Tables 4.29 and 4.30:

Table 4.29: Comparing the Performance of the Different Algorithms on the Coral Bleaching Dataset with the RNN Model Using McNemar's Test

Algorithm	F_1	**p-value vs RNN**
Window-Based Tagger	0.842	0.039
CRF	0.835	0.362
HMM	0.747	< 0.001
Structured Perceptron	0.837	0.093
Bidirectional RNN	0.842	-

Table 4.30: Comparing the Performance of the Different Algorithms on the Skin Cancer Dataset with the RNN Model Using McNemar's Test

Algorithm	F_1	**p-value vs RNN**
Window-Based Tagger	0.814	< 0.001
CRF	0.804	< 0.001
HMM	0.675	0
Structured Perceptron	0.814	< 0.001
Bidirectional RNN	0.837	-

The results show that on the Coral Bleaching dataset, there was a statistically significant difference between the RNN model and the HMM model, but not between the RNN and other models. Thus on the Coral Bleaching dataset, while the Cochran's Q test indicated that there was a significant difference between the performance of all of the models, the RNN model was not significantly better than the other models. However in the Skin Cancer dataset, the differences between the RNN model and the other models was below the corrected α value of 0.0125, indicating the RNN model was the superior model on that dataset. Overall this makes sense, as the gap between the RNN model's micro-F_1 score and the other models was much greater on the Skin Cancer dataset.

The Structured Perceptron, CRF model and Window-Based classifier all utilized the same set of features computed from a window of words surrounding the target word to be tagged. It is not surprising then that they have very similar performance on the word labeling task in both datasets. However, the RNN model was capable of deriving its own features from the text due to its recurrent nature, and its use of a pre-trained embedding layer. The writing in the skin cancer essays was more complex; on average the sentences were longer and had more unique words. It appears that RNN model with the GloVe embeddings was better able to handle this complexity than the other techniques, and extract additional information that was useful for that dataset. On the coral bleaching dataset, however, its predictions were not significantly different than the other strong models, and it likely learned analogous features over the input text. Due to the differences in the results across different datasets, to determine which model is superior at this task requires repeating this test on a number of other analogous datasets.

4.8.5 Extensibility to Other Texts and Domains

An important question for any NLP research problem is how extensible the results are to other texts and to other domains. While the same models perform at a high level of accuracy across the two datasets tested, there are some differences in the performance of the models across the two datasets, which has implications for how well these results extend to other domains. The accuracy of the RNN model is correlated to the frequency of each code, although that correlation is much higher on the skin cancer dataset, where the writing was more complex. This implies that the RNN model would generalize well to other scientific essays with a similar level of writing complexity, provided a similar number of data points were provided, and the writing was no more complex.

It may however be possible to achieve similar performance on a much smaller dataset. In Hastings et al [96], the authors used the same set of coral bleaching essays to train the window-based tagger, and were able to attain a micro-F_1 score of 0.77 using only 20% of the essays when selected using active learning. Active learning is a technique where a model is initially trained on a classification task, and is then used to select the best data points to label next to improve the model's performance on that task [199]. This implies that the window-based classifier could perform well on a smaller dataset, especially if active learning were used to select the data-points to label.

As we will see in the following chapter, the larger the number of labels, the harder it is to perform well on a multi-class classification task. Thus if these techniques were applied to another set of scientific essays, or another domain entirely where there were a lot more unique concepts to learn, the performance may be much worse. The quality of the writing, the grammar and the spelling was quite poor in these essays, so assuming the same pre-processing was applied, it is fair to assume that these models could also perform well in other domains with similar quality writing. Furthermore, if these models were trained in domains with better quality writing and spelling, it is likely that fewer data points would be needed to attain similar levels of accuracy because the text would have less variation. Key to the performance of the RNN model was the availability of the pre-trained word embeddings, trained on a semantically related text. For the RNN model to perform at a similar level on a different domain, it is fair to assume that it would also require word embeddings trained on a semantically related dataset (to that domain).

Overall, I can only speculate as to how well these techniques will generalize to other texts as this is very much an empirical question. Evaluating these techniques on a broader range of domains and types of problem, not just learning scientific concepts from student essay writing, would provide a better understanding of the overall utility of these models at performing this task, as well as in determining how well these approaches generalize beyond the texts studied in this work.

4.9 Conclusions

Deep neural networks have been shown to excel at some NLP transfer learning tasks, provided the different tasks are semantically related [251, 150] This usually involves using a pre-trained model (normally a language model) to initialize the lower layers of a deep neural network, such as a word embedding layer. The neural network is then trained to perform a different NLP classification task. In this chapter, I demonstrated how important the pre-trained word embeddings were for the performance of the bidirectional RNN model. Without the embeddings, the RNN would not have achieved the highest micro-F_1 score on either dataset. Deep neural networks typically require large datasets to perform well on NLP tasks, however the two datasets studied here consist of only around 10,000 sentences and less than 200,000 words. Most deep learning NLP models are trained on datasets that are many orders of magnitude larger. One of the most popular datasets, the Penn Treebank, contains 4.5 million words and is commonly used to train POS tagging and Named-Entity Resolution models [141]. However, by using embeddings trained on a large semantically related dataset (1.6 billion words from Wikipedia), a deep RNN was able to generalize effectively on a small dataset. This has broader implications for other word labelling tasks, where the dataset is small but embeddings exist that have been trained on a dataset with semantically similar content.

In addition, the bidirectional nature of the RNN model was also key to its higher classification accuracy. The other word tagging models were able to learn from either the context to the left of the word to be labelled, a window surrounding the target word, or both (CRF and perceptron). However, the bidirectional RNN model was the only method capable of learning from all of the words to right as well as the left of each target word. Training a one-directional RNN on the same task produced lower classification accuracy, performing worse than all of the other models except the HMM model on both datasets. These results imply that for word labelling problems *in general*, the entire context of the word within the sentence is important for accurately predicting some labels, not just the surrounding context, or preceding words.

In the next chapter, I will discuss the different approaches to detecting causal relations between the concept codes identified by the models described in this chapter.

Chapter 5

Causal Relation Extraction

Research Question 2 asks:

> "What is the most effective approach for determining causal links between concepts from the algorithms below?"
>
> A Extending the optimal tagging model from Research Question 1 to detect causal relations
>
> B Creating a Stacked Model using the predictions from the optimal word tagging model from Research Question 1
>
> C Transition-Based Parsing Model
>
> D Recurrent Neural Network

To address Research Question 2, this was treated as a multi-label classification problem - given a sentence, predict which causal relation or relations are present. To help illustrate the nature of this problem, please refer to Figure 5.1 below which extends Figure 4.1 from the previous chapter. This illustrates how causal relations, shown as blue arcs, are formed between the pre-identified concepts within the essay.

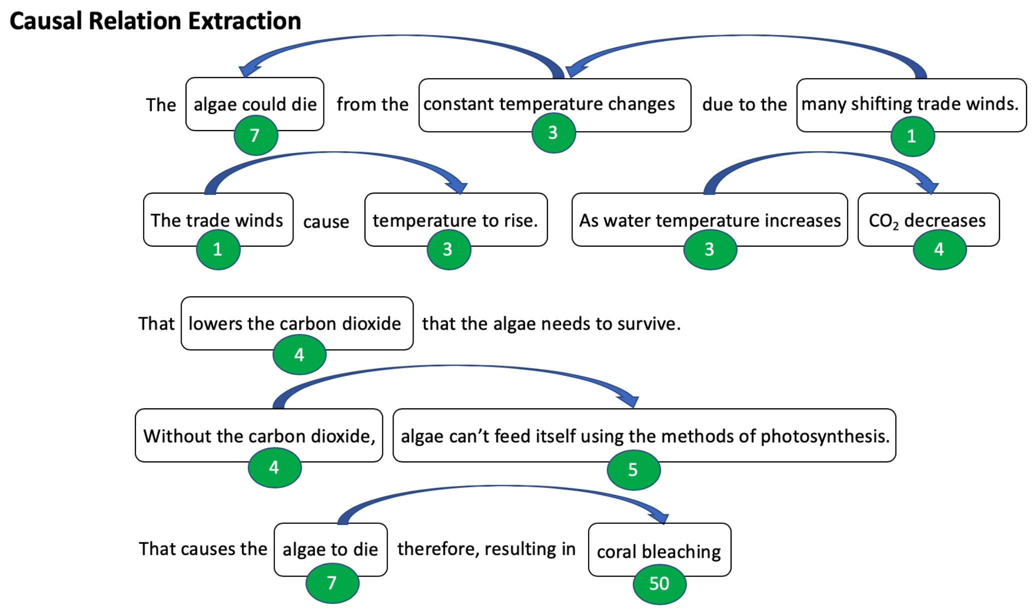

Causal Relation Extraction
The
algae could die
7
from the
constant temperature changes
3
due to the
many shifting trade winds.
1
The trade winds
1
cause
temperature to rise.
3
As water temperature increases
3
CO₂ decreases
4
That
lowers the carbon dioxide
4
that the algae needs to survive.
Without the carbon dioxide,
4
algae can't feed itself using the methods of photosynthesis.
5
That causes the
algae to die
7
therefore, resulting in
coral bleaching
50

Figure 5.1: Extracting causal relations between concepts. Each box represents a separate concept, with the code denoted by the green circle. The cause-effect relations are indicated by the blue arrows, each arrow going from causer to effect code.

In part A of Research Question 2, the best word tagging model from Research Question 1 was adapted to tag causal relations. This allows us to determine how effective the same technique is at extracting causal relations, where the tags span a much longer sequence of words. Causal relations can span the length of a whole sentence, and are often influenced by the presence of the other codes and causal relations that occur within the same sentence because they may form part of a larger causal chain. To attempt to utilize this information, for part B, a stacking model was built using the predictions from the best word tagging model. 'Stacking' or 'Stacked Generalization' is a model ensembling technique where the predictions from a set of base-classifiers are used as training data to train a second set of meta-classifiers to solve a prediction task [246]. By providing a stacked model with information about which concept codes are in a sentence, and their predicted probabilities, it should be able to more effectively determine which concepts form a causal relation.

The main drawback to these two initial approaches is that they treat each unique causal relation as a separate label to be predicted and may perform poorly on causal relations that contain few examples in the training data. A different approach that attempts to solve this problem is to treat the problem as a binary classification problem - given a pair of predicted concept codes, does there exist a causal relation between them? One analogous problem from the domain of NLP is dependency parsing, a type of natural language parsing that detects a set of binary grammatical relations between words in a sentence. For part C, a transition-based dependency parsing model will be trained to parse causal relations instead of binary dependency relations. A transition-based dependency parser makes a sequence of parsing decisions, reading words from an input stream and combines them iteratively to form a dependency parse-tree. However for this problem, the relations exist between concepts and not words, and not all concepts connect together to form a tree structure. Subsequently a novel dependency parsing algorithm was developed using a custom transition system and features designed specifically for this task. At the heart of any parsing problem is a complex search problem that determines which is the most likely parse tree given the many different ways a sentence can be parsed. The SEARN algorithm has been shown to outperform a number of structured learning algorithms on a number of complex NLP tasks [43, 44]. SEARN treats a stuctured learning problem as a sequence of search decisions, training a separate machine learning model to make each decision, and was be used to train this parsing model. Core to solving any supervised learning problem is optimizing the predictions based on the chosen evaluation metric. The flexibility of the SEARN algorithm is that it is able to optimize any evaluation metric, even if it is not differentiable. I will show how a custom cost function can be used to improve the parser's accuracy by optimizing the micro-F_1 metric.

In order to determine whether a sentence contains a causal relation, it is important to understand not just what concept codes are present in a sentence, but the context in which they are used, and the relationships between the words in the sentence. RNN's are effective at learning these types of long distance relationships between the words in a sentence because they maintain an internal state through their recurrent connections as they process the words sequentially. In addition, bidirectional RNN's are able to make use of the context on either side of a word because they process a sentence in both directions. Bidirectional RNN's have recently achieved state-of-the-art results in a number of NLP tasks, including language modeling [146, 248] and sequence labeling [78, 86], [172, 237], and is the final model that was used in part D to detect causal relations. All of these techniques are described in more detail in Chapter 3, and section 1.3.2 discusses the motivations for this choice of algorithms in more detail.

In this chapter, I will discuss how classification performance on this task was measured and evaluated, and how each algorithm was adapted to solve this problem, and I will then compare the efficacy of each of the selected algorithms at this task.

5.1 Evaluation Metrics

The same evaluation metrics described in section 4.1 were used to evaluate the classification accuracy of the different approaches to this problem, namely the micro-precision, micro-recall and micro-F_1 score metrics. However, causal relations span multiple words in a sentence and cannot be associated with a single word. Consequently, classification accuracy was calculated at the sentence and not the word level for this task, resulting in this becoming a multi-label classification (MLC) problem because multiple causal relations can potentially be assigned to each sentence. For each approach, the model's accuracy at detecting the individual causal relations in the sentence was evaluated, where a causal relation links exactly two concept codes from the causal model. For example, the concept STORMS / RAINFALL from the coral bleaching causal model can cause an INCREASE IN FRESH WATER (see Figure 2.1).

For the purpose of calculating the evaluation metrics, each unique causal relation observed in the entire dataset was treated as a separate label to be predicted, and the micro-average metrics were computed over all of those labels. Some causal relations were very rare and were only present in either the test dataset or the training dataset but not both (often these had only one or two occurrences, and were present only in one or two essays). To ensure that the training and test metrics were comparable and shared the same labels, the models were trained and evaluated using all of the labels that exist in the entire dataset, not just the labels that exist in the training dataset only.

Not all causal relations annotated in the dataset consisted of valid inferences present in the causal model. However, these inferences are still important for representing the causal arguments presented by the students in their essays, and are also important inputs to any tutoring system built using this research, and so are considered valid labels for this task. For example, the following two causal relations would be treated as different target causal relations, even though the second relation is an invalid inference:

- 'INCREASE IN WATER TEMPERATURES' $\rightarrow$'DECREASE IN PHOTOSYNTHESIS'

- ‘Decrease in Photosynthesis’ →‘Increase in Water Temperatures’

The first inference is valid - that increasing water temperatures causes a drop in photosynthesis, and this can be inferred from the source text, and forms part of the causal model. However, in some essays the authors reverse the causality and make the second claim - that a decrease in photosynthesis causes an increase in water temperature. This is an invalid inference based on the source text, but is still included in the annotations, and serves as an additional separate causal inference for the model to learn to label.

It should also be noted that I only consider causal relations that exist between two concept codes from the causal model. Any other causal inferences that are present in the student essays are not considered part of this task, and were not annotated in the dataset.

5.2 Experimental Design

Because some of the approaches evaluated in this chapter use predictions obtained from the RNN word tagging model from Chapter 4, the same training and validation datasets were used to evaluate the performance of the algorithms on this task and in the previous research question. Please refer to Section 4.2 and Figure 4.2 for more details about how the different data partitions were defined.

5.3 Pre-Processing

The same pre-processing was carried out on the dataset for each approach, as described in Section 4.4. For both the stacked model and the transition based-parser, the training and test datasets were also augmented with the concept codes predicted by the bidirectional RNN model used to solve Research Question 1. Predictions from the RNN model were used because that model had the highest micro-F_1 scores on the validation data for both datasets. To obtain the predicted concept codes for the training dataset, the predictions were obtained by running 5 fold cross validation and taking the predictions from the held out fold on each run. For the test dataset, the predictions were obtained using a model that was first trained on the entire training dataset. In both cases, this ensured that the tagging model's predictions always came from data unseen by the model making the predictions, and thus represent the algorithm's performance on new data points.

5.4 Model Evaluation

As described in Section 5.1, for each different approach the micro-F_1 score was computed over all causal relations at the sentence level using 5-fold cross validation. For Research Question 1, the same set of features was used for 3 of the 5 different models because those 3 models all made use of lexical information computed over a word-window surrounding the word to be tagged. However, each of the models used to address Research Question 2 in this chapter use very different approaches to solving the problem, and consequently different approaches were used for each model. Once feature selection was complete, hyper-parameter tuning was performed on each model as before. In each step, the optimal set of features and the optimal hyper-parameters were chosen based on the highest observed micro-F_1 score on the validation dataset. These features and hyper-parameter settings were then used to re-train each model on the training data and evaluate its performance on the test dataset.

5.4.1 RNN Word Tagging Model

The first candidate approach to causal relation extraction is to use the optimal word tagging model from Research Question 1, the bidirectional RNN model, to solve the causal relation extraction task. To achieve this, the causal relation extraction problem is modelled as a word tagging problem, where every word that constitutes part of the causal relation, or is tagged with one of the two concept codes that make up the relation, is then labelled with the causal relation tag. Because I am measuring the classification accuracy at the sentence level and not at the word level, the model's predictions had to be rolled up to the sentence level. The predicted sentence level causal relations formed the union of all of the causal relations predicted for one or more words in the sentence. Because the algorithm chosen for part D of this problem is also a Recurrent Neural Network, evaluating an RNN's performance on this task addresses both part A and part D of Research Question 2.

To train the model, each causal relation was defined as a separate tag, and the algorithm was trained to predict the tag assigned to each word. Causal relations span much longer sequences of words than concept codes, and so a lot more words have multiple overlapping causal relations (see Tables 2.4 and 2.9 in Section 2.4). Unlike most traditional machine learning models, neural-networks are capable of predicting multiple output labels in multi-label classification problems. Rather than using a softmax layer to predict a probability distribution over different classes, a neural network can be trained to predict a binary vector representing a one-hot encoding of the different labels present. Using all the different network configurations described in Section 5.4.1.1, different RNN models were trained to predict a binary output vector per word using the binary cross-entropy cost function. However, none of the RNNs converged on the training data; each model predicted there were no causal relation tags for any word. Because there were so many target labels - 86 for the coral bleaching data and 49 for the skin cancer dataset, it appears that the labels are too sparse and infrequent to learn an effective multi-label classifier with a neural network using this approach.

Instead of predicting a binary vector, the most common causal relation was assigned to a word if the word was labelled with multiple relations, mirroring the design of the RNN word tagging model from the previous chapter (see Section 4.6). Words with no causal relations were assigned a special 'EMPTY' tag. The same RNN model architecture was used as described in Section 4.7.5, namely a word-embedding layer using 100-dimensional pre-trained GloVe vectors [168], a 2 layer bidirectional RNN (using a Gated Recurrent Unit) and a softmax layer. Cross-entropy was used as the loss function and the *Adam* optimizer [118] was used to train the deep neural-network. Early stopping using a hold out set was used to prevent overfitting [150].

5.4.1.1 Hyper-Parameter Tuning

One of the key advantages of using a deep-neural network model is that no feature selection is needed; the model is sophisticated enough to learn its own set of representations over the data. In place of feature selection, the performance of a neural-network is very dependent on its network configuration, and this formed the focus of the hyper-parameter tuning as before. In the previous chapter, I showed that the performance of a RNN at tagging concept codes for this domain was dependent on the recurrent network being bidirectional, and using pre-trained rather than randomly initialized word embeddings. Without either of these aspects of the model, the RNN under performed most of the other model types, so they appear to be critical for its performance on word labelling tasks on these 2 datasets. For this task I assumed these 2 features were also critical for attaining good classification accuracy, and only the size of the RNN was optimized using grid search. Bidirectional RNN's comprised of one or two layers of GRU's were evaluated using 3 different sizes of hidden layer - 64, 128 and 256 neurons. Because the *Adam* optimizer was used to train the network (see Section 4.7.5), it was unnecessary to tune the learning rate.

The validation data micro-F_1 scores on both datasets across different hyper-parameter settings are listed in Tables 5.1 and 5.2 below. These tables show the causal relation Micro-F_1 scores across different hyper-parameter values for the RNN model. In each table, the rows are sorted by Micro-F_1 score, and *Num. Parameters* represents the total number of connections, or trainable parameters, in each neural network.

Table 5.1: RNN causal relation Micro-F_1 scores on the Coral Bleaching validation data across different hyper-parameter values.

Micro F_1	**Num. Layers**	**Hidden Layer Size**	**Num. Parameters**
0.680	2	256	1,526,799
0.673	2	128	552,571
0.650	1	256	738,811
0.636	1	128	355,195
0.635	1	64	237,115
0.634	2	64	286,651

Table 5.2: RNN causal relation Micro-F_1 scores on the Skin Cancer validation data across different hyper-parameter values.

Micro F_1	**Num. Layers**	**Hidden Layer Size**	**Num. Parameters**
0.769	2	256	1,511,870
0.756	2	128	542,398
0.754	1	256	723,902
0.752	2	64	278,846
0.742	1	128	345,022
0.730	1	64	229,310

The results show that in general, the greater the number of trainable parameters (connections) the higher the micro-F_1 score. The Pearson correlation can be used to quantify the strength of a linear correlation [191], and ranges from -1 for a strong negative linear correlation to +1 for a strong positive linear correlation, with values close to 0 meaning no correlation. The Pearson correlation between the micro-F_1 score and the number of connections in the network is 0.817 on the coral bleaching dataset, and 0.824 on the skin cancer dataset, indicating a strong positive correlation between these numbers in both cases. The number of connections represents the overall capacity and complexity of the neural network; larger networks can learn more complex functions over the training data, which appears to help with this task. For both datasets, the highest micro-F_1 score was achieved with a 2-layer RNN with 256 hidden units. This is the same configuration that achieved the highest micro-F_1 score on the word labeling task for Research Question 1, and also represents the network configuration with the highest number of connections.

5.4.1.2 Results

The classification accuracy of the RNN model using the optimal network configuration on the training, validation and test datasets are summarized in Tables 5.3 and 5.4 below.

Table 5.3: Bidirectional RNN Causal Relation Classification Accuracy on the Coral Bleaching Dataset

Data Partition	**Micro-F_1**	**Micro-Recall**	**Micro-Precision**
Training Data (CV)	0.869	0.871	0.866
Validation Data (CV)	0.680	0.695	0.665
Test Data	0.676	0.656	0.698

Table 5.4: Bidirectional RNN Causal Relation Classification Accuracy on the Skin Cancer Dataset

Data Partition	**Micro-F_1**	**Micro-Recall**	**Micro-Precision**
Training Data (CV)	0.872	0.855	0.889
Validation Data (CV)	0.769	0.761	0.777
Test Data	0.792	0.798	0.786

The RNN model was much more accurate on the skin cancer data than the coral-bleaching data at this task (17% higher micro-F_1 score). On the previous research question, the RNN as well as all the other models were slightly more accurate on the coral-bleaching dataset, indicating that the causal relation labeling task was easier on the skin cancer data.

The performance of the model on the skin cancer dataset was higher for the test data than the validation data, which matches our observations on the previous research question. Again, this suggests the larger training dataset helped the model to generalize better on the skin cancer data. However, the model performed slightly worse on the test data for the coral bleaching dataset indicating that the model over-fit the data slightly.

One limitation with this approach is that the model is trained to only predict the most common causal relation tag for a word where multiple causal relations are present. Unlike the concept codes where very few words contained multiple codes for either dataset, 2.9% of words in the coral bleaching data and 4.3% of words in the skin cancer data were assigned more than one causal relation. This represents 13.8% of words with causal relations in the coral bleaching data and 13.5% of words with causal relations in the skin cancer data. An examination of the accuracy of the model (at the sentence level) at predicting causal relations that overlap with other relations on at least one word shows that the model is much less accurate at predicting overlapping causal relations (see Table 5.5 below). To compute the prediction accuracy on overlapping relations, the model's predictions were filtered to examine the prediction accuracy for just the overlapping relations compared to the non-overlapping relations. This calculation measures the micro-recall metric because the data was filtered based on the actual labels. If the data was instead filtered by the model's predictions, this would compute the micro-precision metric instead. However, the micro-precision would not be useful measure of overlap classification performance, because it would show how well the model predicts that causal relations overlap, and not how well the model actually performs on overlapping relations.

Table 5.5: RNN Micro-Recall on Overlapping vs Non-Overlapping Causal Relations

Dataset	**Overlapping Relations**	**Non-Overlapping Relations**
Coral Bleaching	0.563	0.754
Skin Cancer	0.691	0.831

However, it could be that these relations are particularly hard to predict given the number of overlapping words. Comparing the performance on this model on overlapping causal relations with the other models will better illustrate if this is a major limitation of this approach.

5.4.2 Stacked Model

Stacking is a model ensembling technique where predictions from one or more base models are used as features to train a separate meta-classifier on a supervised learning task. Empirically, stacking has been shown to be particularly effective for solving multi-label classification problems [145] by using the predictions from multiple binary classifiers to more accurately predict the final set of labels. For this specific problem, a separate binary classifier was trained for each unique causal relation, and their predictions were used as features to train a separate meta-classifier. For more information on stacking as a machine learning technique, see Section 3.3.1.5.

In order to train a stacked model, I needed to use predictions on data that were not part of the base-classifier's training data because most classifiers attain much higher classification accuracy on the training data than on unseen data points. Overfitting can occur if training the stacked classifier on training data predictions as they don't reflect the noise present in predictions on unseen data points. The stacked model was therefore trained on the predictions on the validation data obtained by performing 5-fold cross validation on the training data with the base classifiers. These predictions then formed the training dataset for the stacked model. In order to evaluate the model's performance on the test dataset, the meta-classifier was first trained on this training dataset of cross validation predictions. Then a second set of base-classifiers were trained on the entire training dataset without cross validation, and their predictions on the test data were used as inputs to the stacked model. A logistic regression classifier was chosen as the meta-classifier because it is robust to overfitting and can learn from arbitrary features computed over the inputs. The best performing word tagging model was used as the base-classifier, which was the RNN for both datasets.

5.4.2.1 Feature Selection

To make sentence level causal relation predictions using predictions from the word tagging model, features were extracted using the word level predictions from the entire sentence. The use of confidence estimates from the base-classifiers as features has been shown to improve the accuracy of the stacked model's predictions on a number of tasks [221, 198]. Consequently, the minimum, maximum and average probability estimates for each concept code were included in the evaluated feature sets. In addition, a set of binary features were computed indicating whether each concept code was predicted at least once (with a probability of at least 0.5) and a separate set of binary features were created for each pair of codes that were predicted to be present somewhere in the sentence. Because the number of possible features for this model was relatively small, a grid-search was performed over every combination of feature sets to determine the optimal set on the validation data using 5-fold cross validation. The optimal feature set for the coral bleaching dataset were the combination of binary concept code features and the binary code pair features, while for the skin cancer dataset the maximum and average predicted probabilities and the binary code pair features achieved the best performance.

The reduction in the number of features used by the stacked model as a result of feature selection are shown in Table 5.6 below.

Table 5.6: The Reduction in Number of Features for the Stacked Model as a Result of Feature Selection

Dataset	**# All Features**	**# Optimal Features**	**% Reduction**
Coral Bleaching	147	105	28.6 %
Skin Cancer	85	65	23.5 %

5.4.2.2 Hyper-Parameter Tuning

Once the optimal feature sets were determined, the stacked model's hyper-parameters were then optimized using these feature sets. A grid search was again performed to determine the optimum regularization method and corresponding C value for the logistic regression model. C values of 0.1, 0.5, 1, 10, and 100 were evaluated, and an L1 penalty with a C value of 1.0 produced the best micro-F_1 score on the coral bleaching dataset, while dual mode (L1 and L2 regularization) with a C value of 1.0 produced the best micro-F_1 score on the skin cancer dataset.

5.4.2.3 Results

The performance of the stacked model on all dataset partitions and both data sets using the optimal feature sets and hyper-parameters can be seen in Tables 5.7 and 5.8 below:

Table 5.7: Stacked Model Causal Relation Classification Accuracy on the Coral Bleaching Dataset

Data Partition	**Micro-F_1**	**Micro-Recall**	**Micro-Precision**
Training Data (CV)	0.741	0.711	0.774
Validation Data (CV)	0.695	0.656	0.738
Test Data	0.704	0.674	0.736

Table 5.8: Stacked Model Causal Relation Classification Accuracy on the Skin Cancer Dataset

Data Partition	**Micro-F_1**	**Micro-Recall**	**Micro-Precision**
Training Data (CV)	0.801	0.757	0.850
Validation Data (CV)	0.763	0.711	0.823
Test Data	0.765	0.719	0.816

Compared to the RNN model, the stacked model achieves a higher overall micro-F_1 score, with a higher micro-precision but a lower micro-recall on the coral bleaching data, on both the test and validation datasets. However, on the skin cancer dataset, the stacked model performs worse than the RNN model in both the validation and test data, with a lower micro-F_1 score, micro-precision and micro-recall on the validation data, and a slightly higher recall but lower micro-F_1 score and micro precision on the test data. On both datasets, the model again achieves a slightly higher micro-F_1 score on the test data because it was trained on a larger dataset.

5.4.3 Shift-Reduce Dependency Parser

The final model evaluated for this task adapts a shift-reduce dependency parser approach to detect causal relations in the essay sentences. Dependency parsing is a form of natural language parsing that identifies the binary grammatical relations between words in a sentence (see Section 3.4.3). A transition based shift-reduce dependency parser is a greedy approach to dependency parsing that approximates a globally optimum solution by making a series of locally optimal choices, guided by a discriminative classifier trained on some golden dataset of parsed sentences [158], resulting in the construction of a dependency parse tree. A dependency parse tree consists of a set of binary dependency relations, where each word in a sentence has exactly one directed asymmetric dependency relation with one other word in the sentence, forming a tree structure. Figure 5.2 below shows one example dependency parse structure learned by the causal relation parser from one sentence in the coral bleaching dataset:

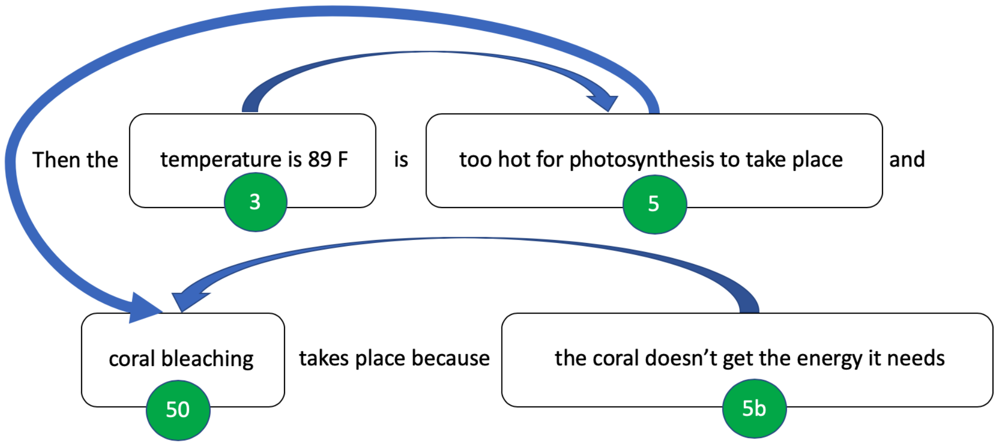

Then the
temperature is 89 F
3
is
too hot for photosynthesis to take place
5
and
coral bleaching
50
takes place because
the coral doesn't get the energy it needs
5b

Figure 5.2: Example coral bleaching dependency parse structure. Each box represent a concept, with the code denoted by the green circle. The cause-effect relations are indicated by the blue arrows, each arrow going from causer to effect code

How the parser constructs this parse structure will be explained briefly in this chapter, and in more detail in Appendix G. A shift-reduce parser is a type of conditional-history based parsing model which makes sequential parsing decisions conditioned in part on its parse decisions earlier in the parse. Conditional-history parsers, data-driven natural language parsing and dependency grammars are discussed in more detail in Section 3.4.3.

5.4.3.1 Shift-Reduce Transition-Based Dependency Parsing

A shift-reduce parser is an efficient approach to table-driven bottom-up parsing that is used to parse formal languages such as programming languages, in addition to its use in natural language parsing. Parsers operate on an input stream of tokens, transforming them into a parse tree. In the case of natural-language parsers, the input stream usually consists of a sequence of words forming a sentence. Shift-reduce parsers construct the parse tree incrementally, bottom-up and left to right without backtracking, by making a combination of *shift* and *reduce* steps, while manipulating a stack. The stack represents the current state of the parser at any step in the current parse, and can hold one or more partially complete parse trees that are eventually combined into one final parse tree. The *shift* step advances the input stream by one token, and creates a new single-node parse tree out of the shifted token and pushes it onto the top of the stack. A *reduce* step then applies some grammar rule to some of the parse trees in the stack, joining them together into a single tree with a new root node. To parse a sentence, the parser applies shift and reduce steps as needed until all of the input sequence has been consumed and all partial parse trees on the stack have been combined into a single parse tree.

Natural language dependency parsers are typically *transition-based parsers*. This means that they are defined in terms of a transition system - an abstract state machine that consists of a set of rules governing how the system transitions from one state to another. A number of different transition systems have been developed to adapt shift-reduce parsing to the task of natural-language dependency parsing, including Nivre's *arc standard* and *arc-eager* systems [154, 155]. Unlike traditional dependency parsers which operate on the sequence of words in a sentence, the causal relation parser instead operates on the concept codes predicted by the RNN word tagging model. The parser uses a modified version of Nivre's *arc-eager* transition system to allow it to ignore orphan concept codes that don't form part of any causal relation. Please refer to Appendix G for more details on the transition system.

One of the challenges in training a parsing model using machine learning is enabling the system to learn to recover from its own errors, and make the optimal parsing decision even when it is in a state not observed in the training data. In 2012 Goldberg and Nivre presented the idea of a *dynamic oracle* [84], which is capable of determining the optimal parse decision provided any parser configuration, including those that deviate from the golden parse tree (the optimal parse tree from the training data). If the parser deviates from the golden parse during training, the oracle examines all possible parses that can be reached from the current parser state, and determines the parsing decision which produces the fewest errors in the final parse tree.

An approach called imitation learning was used to enable the parser to learn from the training data. Imitation learning adapts reinforcement learning to solve supervised learning problems. Training a parser involves learning to make a sequence of classification decisions before a final prediction is made. Usually the loss associated with a particular decision is not known, only the loss associated with the final prediction. It is not clear which of these decisions contributed most to the success or failure of the final prediction, resulting in a credit-assignment problem. This is the exact problem reinforcement learning algorithms attempt to solve. The SEARN algorithm is an imitation learning algorithm that solves structured prediction problems by decomposing each problem into a sequence of cost-sensitive classification decisions, and then trains a set of cost-sensitive classifiers to handle each of these decisions [43]. A key advantage of this approach is that any arbitrary loss function can be optimized using the SEARN algorithm by adapting the cost-function based on the desired loss function. Subsequently, a custom loss function was designed that optimizes the micro-F_1 score during training. Appendix G describes the shift-reduce parser algorithm in more detail, and explains how the SEARN algorithm was used to train the parser on the training data to optimize the micro-F_1 score. For specific details on the custom cost function, please refer to Appendix G Section G.5.

Because the causal relation parser is a modified dependency parser, the features used to train the machine learning models need to be appropriate for training a parsing model. In the next section, I will discuss the different types of features used in training this model.

5.4.3.2 Features Evaluated

The features evaluated for the dependency parser were based on those used by Zhang and Nivre [258] to train a dependency parser. Zhang and Nivre used a combination of local and non-local features, and achieved the state-of-the-art attachment score on the Penn Treebank at the time of publication [258]. As discussed in the previous sections, instead of parsing individual words, the parser operates on concept codes, each of which can span a sequence of several words. The concept codes are predicted by the recurrent neural network word tagging model, discussed in the previous chapter in Section 4.7.5. The parser configuration can be represented by the triple $\langle S, I, A\rangle$, where S is the stack, I is the list of remaining input tokens, and A is the current set of arc relations for the dependency tree. The different feature templates can be seen in Tables 5.9 and 5.10, which attempts to capture all of the salient information about the current parse state necessary for making the next parsing decision, and groups them into feature sets such as *Single Concept Codes*, *Valency* and *Unigrams*.

I will use a slightly different notation here to clarify which tokens come from the stack and which come from the input stream, their relative locations within each, and also to illustrate that these tokens are concept codes and not words, as in the previous example. I denote the top of stack token S_0 and the front items of the input stream with I_0, I_1 and I_2, individual words associated with a concept code are denoted as w while the codes or tags are denoted with a t. Often multiple words are associated with each concept code, in which case each word forms a separate feature. It is common when training dependency parsers to use features describing the left and right modifiers of S_0 and the left modifiers of I_0 in combination with the POS tag and the words in the sentence. To translate that to the causal relation parser, in place of modifiers I use any codes that form a parsed causal relation in A with S_0 or I_0. Any codes that act as a causer in any causal relation with S_0 are prefixed S_{0hc} and those forming an effect relation are prefixed S_{0he}. Similarly, any concept codes that comprise S_{0h} that appeared to the left of S_0 in the sentence are denoted S_{0lc} for causer codes and S_{0le} for effect codes, and those appearing on the right are denoted S_{0rc} and S_{0re}. These types of features are based on the baseline features used by Zhang and Nivre in [258] and are listed in Table 5.9 as *Single Concept Codes*, *Two Concept Codes*, *Three Concept Codes*.

Additionally, one other set of baseline features was added that is unique to this piece of work. This set encodes the set of words that appear in the sentence between the S_0 and I_0 codes. If both of these codes are adjacent (with no words in between), a special empty set symbol was used to encode this state. The words in between these codes are denoted b, and are combined with the concept codes for S_0 and I_0 to form the three templates called *Between Words*. This set of features was included to allow the model to learn words and phrases that link two concepts that can imply causality, such as "because of" or "causes".

Table 5.9: Baseline Dependency Parser Features

Single Concept Codes
S_0wt; S_0w; S_0t; I_0wt; I_0w; I_0t; I_1wt; I_1w; I_1t; I_2wt; I_2w; I_2t;
Two Concept Codes
S_0wtI_0wt; S_0wtI_0w; S_0wI_0wt; S_0wtI_0t; S_0tI_0wt; S_0wI_0w; S_0tI_0t; I_0tI_1t;
Three Concept Codes
$I_0tI_1tI_2t$; $S_0tI_0tI_1t$; $S_{0hc}tS_0tI_0t$; $S_{0he}tS_0tI_0t$; $S_0tS_{0lc}tI_0t$; $S_0tS_{0le}tI_0t$; $S_0tS_{0rc}tI_0t$; $S_0tS_{0re}tI_0t$; $S_0tI_0tI_{0lc}t$; $S_0tI_0tI_{0le}t$;
Between Words
S_0tb; I_0tb; S_0tI_0tb;

The next set of features templates refer to higher order information about the causal relation graph, and are listed in Table 5.10. The first high-order feature set encodes the cause or effect relations of S_0 and I_0, for example $S_{0hc}w$, which encodes any causer codes forming relations with S_0 in combination with the words tagged with S_0t. These include an additional feature, *lbl*, which encodes the full causal relation label belonging to these relations (including the direction of causality and both codes involved). Next, the *Distance* features relate to the number of words between the S_0 concept code and the I_0 code, and are denoted with a d. For example the template S_0wd comprises the words tagged with the S_0t tag and the value of d (including when $d = 0$). Distance features were used in the easy-first parser of Goldberg and Elhadad in 2010 [83], and then in 2011 by Zhang and Nivre [258]. Zhang and Nivre also describe a set of valency features that count the number of left and right modifiers of a dependency head. I adapt that idea to count the number of cause and effect tags that form parsed relations in A that include S_0 or I_0 and are to the left and right of S_0 or I_0 in the sentence. Left valency features are denoted v_{lc} for cause and v_{le} for effect, and right valency features are denoted v_{rc} and v_{re} respectively.

Higher order context features encoding dependencies of dependencies have been used in a number of different graph-based parsers to improve accuracy [20, 121, 258]. Adapting the templates from Zhang and Nivre [258], I denote S_{0h2c} as the causer of the causer of S_0, S_{0h2e} as the effect of the effect of S_0, thus encoding sequences of parsed causal relations. Similarly, I_{0l2c} denotes the left causers of any causal codes forming any left causal relation with I_0. These higher order context features are called *Third-Order* features in Table 5.10. Additionally, also based on those templates used in [258], I include the set of unique causal relation labels that include S_0 and I_0 and combine these with the word and concept code of S_0 and I_0 to make feature templates. These are again denoted with *lbl*, but include a suffix to denote the subset of relations, e.g. S_0wtlbl_{rc} combines the words and tag of S_0 with the full causal relation labels of any causal relations that include S_0 where the other code is a causer and is on the right of S_0. These are labeled *Label Set* in Table 5.10.

Some sentences consist of multiple causal relations, which may be linked together to form a causal chain, or may enumerate the multiple causes or multiple effects of a single concept code. To attempt to capture this information, I also include a new set of feature templates called *Size Features* that are unique to this piece of work. These templates encode the number of concept codes in the Stack S, in the input stream I, and the number of parsed relations in A. For every concept code forming a relation in A, I also encode the number of times that a code appears in A acting as a causer, denoted A_ct (including that code's tag t) and as an effect, denoted $A_e t$. This last feature template helps to indicate when there are multiple causes or effects forming a relation with a single concept code (usually a code 50).

Table 5.10: Higher Order Dependency Parser Features

Unigrams
$S_{0hc}w$; $S_{0he}w$; $S_{0hc}t$; $S_{0he}t$; S_0lbl; $S_{0lc}w$; $S_{0le}w$; $S_{0lc}t$; $S_{0le}t$; $S_{0lc}lbl$; $S_{0le}lbl$; $S_{0rc}w$; $S_{0re}w$; $S_{0rc}t$; $S_{0re}t$; $S_{0rc}lbl$; $S_{0re}lbl$; $I_{0lc}w$; $I_{0le}w$; $I_{0lc}t$; $I_{0le}t$; $I_{0lc}lbl$; $I_{0le}lbl$;
Distance
S_0wd; S_0td; I_0wd; I_0td; S_0wI_0wd; S_0tI_0td;
Valency
S_0wv_{rc}; S_0wv_{re}; S_0tv_{rc}; S_0tv_{re}; S_0wv_{lc}; S_0wv_{le}; S_0tv_{lc}; S_0tv_{le}; I_0wv_{lc}; I_0wv_{le}; I_0tv_{lc}; I_0tv_{le};
Third-Order
$S_{0h2c}w$; $S_{0h2e}w$; $S_{0h2c}t$; $S_{0h2e}t$; $S_{0hc}lbl$; $S_{0he}lbl$; $S_{0l2c}w$; $S_{0l2e}w$; $S_{0l2c}t$; $S_{0l2e}t$; $S_{0l2c}lbl$; $S_{0l2e}lbl$; $S_{0r2c}w$; $S_{0r2e}w$; $S_{0r2c}t$; $S_{0r2e}t$; $S_{0r2c}lbl$; $S_{0r2e}lbl$; $I_{0l2c}w$; $I_{0l2e}w$; $I_{0l2c}t$; $I_{0l2e}t$; $I_{0l2c}lbl$; $I_{0l2e}lbl$; $S_0tS_{0lc}tS_{0l2c}t$; $S_0tS_{0le}tS_{0l2e}t$; $S_0tS_{0rc}tS_{0r2c}t$; $S_0tS_{0re}tS_{0r2e}t$; $S_0tS_{0hc}tS_{0h2c}t$; $S_0tS_{0he}tS_{0h2e}t$; $I_0tI_{0lc}tI_{0l2c}t$; $I_0tI_{0le}tI_{0l2e}t$;
Label Set
S_0wtlbl_{rc}; S_0wtlbl_{re}; S_0wtlbl_{lc}; S_0wtlbl_{le}; I_0wtlbl_{lc}; I_0wtlbl_{le};
Size Features
$\|S\|$; $\|I\|$; $\|A\|$; $\|A_ct\|$; $\|A_et\|$;

5.4.3.3 Feature Selection

Due to the large number of features, forward selection was used once more to select the best combination of feature sets. A maximum of 6 different combined feature sets were evaluated to ensure the feature selection process remained tractable, and Logistic Regression was used as the model as before. For each dataset, the forward selection process was run twice, once using stemmed words for the word features and a second time without stemming, using 5-

fold cross validation to determine the best F_1 score on the validation data. Default values were used for all of the hyper-parameter values for the Logistic Regression models. The SEARN algorithm itself has 2 key parameters, the β parameter which controls the degree of interpolation between the last 2 trained SEARN models, and the number of iterations, or *maximum epochs*, it is trained over. A value of 0.2 was used for the β parameter of SEARN, and the algorithm was trained for a maximum of 10 epochs, as recommended in [45]. For both datasets, the best F_1 score on the validation data was obtained using stemmed words. Tables 5.11 and 5.12 below compare the accuracy metrics for the best feature sets with and without stemmed words.

Table 5.11: Parsing Model Accuracy for the Best Feature Sets with and Without Stemming for the Coral Bleaching Dataset

	# Feature Sets	**F_1 Score**	**Recall**	**Precision**
Unstemmed	3	0.7140	0.6613	0.7757
Stemmed	6	0.7161	0.6640	0.7777

Table 5.12: Parsing Model Accuracy for the Best Feature Sets with and Without Stemming for the Skin Cancer Dataset

	# Feature Sets	**F_1 Score**	**Recall**	**Precision**
Unstemmed	4	0.7608	0.7055	0.8256
Stemmed	4	0.7634	0.7059	0.8312

For both datasets, the recall is slightly better with the stemmed words. This matches what I saw in the previous chapter; stemming collapses multiple variants of a word into the same form thus allowing the model to generalize better, leading to higher recall. Similarly, for the coral bleaching dataset only, using stemming also allowed the model to make use of more types of feature sets without overfitting the dataset because the optimal number of feature sets was higher. However, the impact of stemming on the overall micro-F_1 score was minimal as only some of the different feature sets used words from the sentence. For the rest of this section, I will focus on the results for the stemmed features only.

In Section 4.5.3 I compared the relative importance of each feature set when used in isolation. However, for this task, the model required at least one set of the baseline features in order to learn to parse each sentence (see Table 5.9 for a list of the baseline feature sets). The higher order features alone did not encode enough about the parser state for the model to learn anything useful, instead they provided value as additional features when used in conjunction with the baseline features. Consequently, Table 5.13 compares only the relative performance of each of the baseline feature sets when used in isolation.

Table 5.13: Each Baseline Feature Set Ranked by Micro-F_1 Score When Used in Isolation (using Stemmed Features).

	Coral Bleaching		**Skin Cancer**	
	Feature Set	F_1	**Feature Set**	F_1
1	Single Concept Codes	0.697	Three Concept Codes	0.741
2	Three Concept Codes	0.692	Single Concept Codes	0.738
3	Two Concept Codes	0.688	Two Concept Codes	0.737
4	Between Words	0.648	Between Words	0.730

From Table 5.13, we can see that for both feature sets, the *Between Words* features were the least useful features, as they focus on the words in between the codes that are being considered, but not on information about the codes themselves that make up the possible causal relation. In the coral bleaching dataset, the features formed from one concept code were more useful than those learned from three codes, while in the skin cancer dataset the opposite is true. It appears the greater specificity of the features created from 3 codes was more useful in the skin cancer dataset than in the coral bleaching dataset. Also, the *From Three Concept Codes* features encoded information from prior parsed causal relations (e.g. $S_{0hc}tS_0tI_0t$), while the *Single Concept Codes* features only use the words and tags associated with S_0 and I_0. In Table 2.9 in Chapter 2, we saw that 12.5% of sentences in the skin cancer dataset have multiple causal relations, while only 6.63% of coral bleaching sentences have multiple relations, and overall a greater proportion of sentences have causal relations in the skin cancer data. It is therefore quite likely that this information is more useful in the skin cancer model because it is much more likely that other relations have been parsed earlier in the sentence.

The true value of many of the feature sets is how much they improve the accuracy when combined with other feature sets. Figure 5.3 shows changes in the maximum micro-F_1 score on the validation data as more feature sets were added:

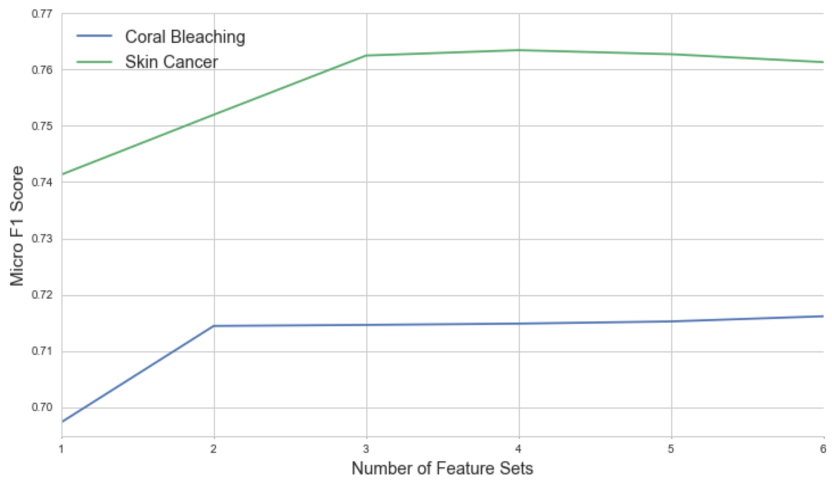


Figure 5.3: Maximum Micro-F_1 Score by Number of Feature Sets (Stemmed Features)

For the coral bleaching essays, the model's validation accuracy continued to increase by smaller amounts as more feature sets were added, but in the skin cancer dataset the algorithm started to overfit once more than 4 feature sets were added. This mirrors the results discussed in Chapter 4 (see Table 4.6) for the previous research question, where the skin cancer model also started to overfit after more than 4 sets of features were added. Once the first 4 feature sets were added, adding *Third Order* and *Valency* features reduced the F_1 score on the skin cancer dataset. This indicates that the skin cancer dataset is more prone to overfitting, especially with large numbers of feature sets. Table 5.14 shows the order in which feature sets were added by forward selection:

Table 5.14: Micro F_1 Score as Additional Feature Sets Are Added by Forward Selection (Stemmed Features).

	Coral Bleaching		**Skin Cancer**	
# Feats.	**Added Feature Set**	F_1	**Added Feature Set**	F_1
1	Single Concept Codes	0.6974	Three Concept Codes	0.7413
2	Between Words	0.7145	Between Words	0.7520
3	Label Set	0.7146	Size Features	0.7625
4	Three Concept Codes	0.7149	Single Concept Codes	**0.7634**
5	Third Order	0.7153	Third Order	0.7627
6	Unigrams	**0.7162**	Valency	0.7613

In both datasets, *Two Concept Codes* were the only baseline feature sets not included in the optimal feature set. This set of features combined information present in other feature sets, and likely did not add sufficient new information to allow the model to generalize well. In contrast, the *Between Words* feature set were the second feature set added in each dataset, following the addition of the top baseline feature set. This implies that the words between the concept codes were useful in determining whether there was a causal relation or not, and provided the most useful new information. *Label Set* and *Unigrams* were only included in the features selected for the coral bleaching dataset, while *Size Features* and *Valency* were only included in the selected skin cancer dataset's features. Because there are more sentences with causal relations and more causal relations per sentence in the skin cancer data, as discussed earlier, both of these features may be more useful to the skin cancer model because they count the numbers of certain types of parsed causal relation. The *Distance* and *Valency* features were not included in either optimal feature set and presumably did not generalize well to new data points.

Forward selection substantially reduced the number of features required to train the parsing model. Table 5.15 below shows the reduction in the total number of features in the parser as a result of forward selection.

Table 5.15: The Reduction in Number of Features for the Shift-Reduce Dependency Parser as a Result of Feature Selection

Dataset	**# All Features**	**# Optimal Features**	**% Reduction**
Coral Bleaching	1,698.8	33.0	98.1%
Skin Cancer	798.8	88.6	88.9%

Once the optimal features were determined using forward selection, the model's hyper-parameters were tuned to optimize accuracy on the validation dataset.

5.4.3.4 Hyper-Parameter Tuning

Because a Logistic Regression model was used as the base classifier, a grid search was performed over the same set of hyper-parameters as in previous experiments - C values of 0.1, 0.5, 1, 10, and 100 were evaluated using L1, L2 and dual mode regularization methods. Two additional hyper-parameters were also tuned that are specific to the implementation of the SEARN algorithm, the β parameter which controls how the model interpolates between the last two trained SEARN models and *max epochs*, which represents the maximum number of training iterations used to train the SEARN model. Values of β of 0.1, 0.2, 0.3, 0.4, 0.5, 0.75 and 1.0 were evaluated, along with *max epoch* values of 1, 2, 3, 5, 10, 15 and 20. It should be noted that for the feature selection process, the recommended default values for these parameters were used, namely a C value of 1.0, dual mode regularization with a β value of 0.2 and a *max epochs* of 10.

For the coral bleaching dataset, a C value of 0.5, using the dual mode of regularization with a β value of 0.5 had the highest micro-F_1 score on the validation dataset with a maximum of 2 epochs. For the skin cancer dataset, a C value of 0.5 using the L1 regularization method with a β value of 0.3 and training for a maximum of 3 epochs were the optimal hyper-parameters. For the first iteration, the SEARN model relies entirely on the oracle's labels to make subsequent predictions. Then, with each new iteration, the model relies more and more on its own past predictions. Having optimal *max epochs* values of 2 and 3 implies that for higher values the model's predictions become too noisy, negatively impacting prediction accuracy. However, after 2 or 3 iterations, the model is starting to make use of its own previous predictions, and so this would appear to be a better approach than relying on the oracle's predictions alone (at iteration 1). With optimal β values being 0.3 and 0.5, interpolating between the previous 2 models when choosing the prior predictions to use works better than picking the predictions from one of these models alone.

5.4.3.5 Results

The performance of the shift-reduce parser model on all dataset partitions and both datasets using the optimal feature sets and hyper-parameters can be seen in Tables 5.16 and 5.17 below:

Table 5.16: Shift Reduce Parser Causal Relation Classification Accuracy on the Coral Bleaching Dataset

Data Partition	Micro-F_1	Micro-Recall	Micro-Precision
Training Data (CV)	0.865	0.766	0.992
Validation Data (CV)	0.722	0.727	0.718
Test Data	0.728	0.766	0.693

Table 5.17: Shift Reduce Parser Causal Relation Classification Accuracy on the Skin Cancer Dataset

Data Partition	Micro-F_1	Micro-Recall	Micro-Precision
Training Data (CV)	0.879	0.790	0.992
Validation Data (CV)	0.768	0.721	0.822
Test Data	0.790	0.760	0.823

As with the other models, the shift-reduce parser has a higher score on the test data than the validation data, due to the larger training set. This also indicates that despite all the feature selection and hyper-parameter tuning, the model did not overfit the training or validation datasets. In the next section, I will discuss how the shift-reduce parsing model compares to the other two approaches.

5.5 Summary of Results and Discussion

To better understand how well these different techniques perform at extracting causal relations from the two different datasets, I will compare their performance using not only their Micro-F_1 performance, but also their Macro-F_1 performance, which is more sensitive to the accuracy of the models at predicting rarer relations. Furthermore, I will analyze how the accuracy at predicting causal relations varies based on the relative frequency of each relation, how the models perform on valid versus invalid casual relations (causal relations not in the model or where the causal direction is incorrect), and I will perform a statistical analysis to validate whether there is a statistically significant difference between the performance of the different models. Finally, I will discuss how well these techniques might generalize to other domains or other texts.

5.5.1 Micro-F_1 Performance

The micro-F_1 score, precision and recall of each of the 3 algorithms at detecting causal relations are listed in Tables 5.18 and 5.19 below:

Table 5.18: Test Data Accuracy by Algorithm on the Coral Bleaching Dataset. All metrics are micro metrics. Probs = Probabilities.

Algorithm	F_1	**Recall**	**Precision**	**Features**
Bidirectional RNN	0.676	0.656	0.698	Word Embeddings
Stacked Model	0.704	0.674	**0.736**	Word Tagging Probs.
Shift-Reduce Parser	**0.728**	**0.766**	0.693	Templated Parsing Feats.

Table 5.19: Test Data Accuracy by Algorithm on the Skin Cancer Dataset. All metrics are micro metrics. Probs = Probabilities

Algorithm	F_1	**Recall**	**Precision**	**Features**
Bidirectional RNN	**0.792**	**0.798**	0.786	Word Embeddings
Stacked Model	0.765	0.719	0.816	Word Tagging Probs.
Shift-Reduce Parser	0.790	0.760	**0.823**	Templated Parsing Feats.

On the test data partitions, the shift-reduce parser model had the highest micro-F_1 score on the coral bleaching dataset, out-performing the stacked model by 3.41%, while the RNN word tagging model slightly out-performed the parser by 0.25% on the skin cancer dataset. This mirrors what I found earlier for each model on both validation sets. On average, the shift-reduce parser had the strongest performance across both test sets as expected, with an average micro-F_1 score of 0.759 compared with 0.734 for the RNN word tagging model and 0.735 for the stacked model.

The shift-reduce parser has two main advantages over the other two models for this task. Firstly, it does not predict each relation as a separate label, instead it learns when two concept codes can be parsed into a causal relation. This allows it to generalize over all causal relations without having to treat each causal relation separately and learn separate features for each. This may explain the much higher recall metric value for the parser model on the coral bleaching dataset when compared to the other models. Also the coral bleaching dataset has a lot more unique causal relations (86 compared to 49), so being able to generalize over all of these relations is much more useful than for the skin cancer dataset, where the RNN has a higher micro-F_1 score. There are 4 causal relations that occurred only in the coral bleaching test set and 2 that occurred only in the skin cancer test set. Of these, the parser model was able to correctly identify 1 of the coral bleaching codes without having seen any previous examples. While this clearly did not make a noticeable difference to the overall micro-F_1 scores, this illustrates the capacity of this model to learn to detect causal relations between arbitrary concept codes.

The second advantage the parsing model has over the other 2 models is that it optimizes the micro-F_1 score. In general, F_1 score is non differentiable and cannot be directly optimized using gradient based approaches (see Section 4.1.4). As the micro-F_1 score extends the F_1 measure to multiple classes, it does not decompose over all examples, making direct optimization even more difficult. Instead, the parsing decisions are weighted using a custom cost function that assigns instance weights proportional to the estimated cost of an incorrect parsing decision on the overall micro-F_1 score. These instance weights are then used when training a set of cost-sensitive classifiers to make each individual parsing decision. Please refer to Appendix G in Section G.5 for more details. We can see the effect this cost function has on the model's performance by comparing the micro-F_1 score of the model trained with the cost function described in Equation G.1 compared to a cost function that returns the same value for all parsing decisions, which I will call the *uniform cost function.* Because the model's hyper parameters were optimized for the custom cost function, the default hyper-parameter values were used to compare the cost functions (using a C value of 1.0, L2 regularization without dual mode, a β value of 0.2 and 10 *max epochs*). The optimal feature sets were used in this comparison as these are assumed to be the same for both cost functions. Table 5.20 shows the micro-F_1 scores on the test data for both datasets with the micro-F_1 cost function and with the *uniform cost function*.

Table 5.20: Micro F_1 Score on the Test Data with Different Cost Functions

Cost Function	**Coral Bleaching Micro-F_1**	**Skin Cancer Micro-F_1**
Micro F_1 Cost	**0.726**	**0.788**
Uniform Cost	0.694	0.786

The cost function makes a big difference on the coral bleaching dataset, but very little difference on the skin cancer dataset where it makes the model only slightly more accurate. Without the uniform cost function, the Stacked Model would have out-performed the parser on the coral bleaching dataset.

In addition to these two advantages, that the parser model has an additional advantage over the RNN model only, which is that it can predict overlapping causal relations, which the RNN model studied in this chapter does not do. Comparing the micro-recall of each model at predicting causal relations in sentences containing one or more overlapping codes (see Tables 5.21 and 5.22 below), shows that the parser model significantly out-performs the other two models on the coral bleaching dataset on the sentences with overlapping codes, and actually performs slightly worse than the RNN tagging model on sentences without overlapping codes. However, on the skin cancer dataset, the RNN tagging model does better on both types of sentence, so this aspect of the model does not seem to help on this dataset. For all three models on both datasets, the micro-recall is lower on the overlapping causal relations, indicating that these are harder to predict than non-overlapping relations. In each case, it seems the different advantages of the parsing model make a large difference on the coral bleaching dataset, but not on the skin cancer dataset.

Table 5.21: Micro-Recall on Sentences with Overlapping vs Non-Overlapping Causal Relations on the Coral Bleaching Test Dataset

Dataset	**Overlapping Relations**	**Non-Overlapping Relations**
RNN Word Tagging Model	0.563	**0.754**
Stacked Model	0.581	0.709
Shift-Reduce Parser	**0.679**	0.752

Table 5.22: Micro-Recall on Sentences with Overlapping vs Non-Overlapping Causal Relations on the Skin Cancer Test Dataset

Dataset	**Overlapping Relations**	**Non-Overlapping Relations**
RNN Word Tagging Model	**0.691**	**0.831**
Stacked Model	0.655	0.753
Shift-Reduce Parser	0.674	0.753

5.5.2 Macro F_1 Performance

To see how well the different models performed on the rarer causal relations, we can compare the macro-average metrics of the 3 models, which can be seen in Tables 5.23 and 5.24 below. The same model in each dataset with the optimal micro-F_1 score also has the optimal macro-F_1 score, but the macro-average scores are much lower. There are many more unique causal relations than concept codes because each causal relation is a combination of 2 concept codes, and as a result each causal relation occurs less frequently than the codes that it consists of. Because a lot of these relations are very rare, averaging across all relations causes the macro-averages to be dominated by the rarer codes. The machine learning models tend to perform worse on relations with few data points as they have fewer examples to learn from, so the rare codes reduce the macro-average scores, making them considerably lower than the micro-average scores. However, the micro-F_1 score was chosen as the main accuracy metric as it reflects the relative frequency of the relations in the entire dataset, and so better reflects the overall performance of the model on the task as a whole.

Table 5.23: Macro Test Data Metrics by Algorithm on the Coral Bleaching Dataset

Algorithm	**Macro-F_1**	**Macro-Recall**	**Macro-Precision**
RNN Word Tagging Model	0.211	0.199	0.226
Stacked Model	0.189	0.189	0.189
Shift-Reduce Parser	**0.306**	**0.338**	**0.280**

Table 5.24: Macro Test Data Metrics by Algorithm on the Skin Cancer Dataset

Algorithm	**Macro-F_1**	**Macro-Recall**	**Macro-Precision**
RNN Word Tagging Model	**0.342**	**0.343**	**0.342**
Stacked Model	0.271	0.250	0.296
Shift-Reduce Parser	0.302	0.286	0.320

On the word tagging task, each model performed slightly better on the coral bleaching dataset than the skin cancer dataset. On the causal relation extraction task however, each model performed significantly better on the skin cancer dataset. This is particularly surprising because 2 of the 3 models evaluated relied on predictions from the word tagging models. There are several possible reasons for these differences in performance. Firstly, there are fewer observed unique causal relations in the skin cancer dataset, 49 different relations compared to 86 in the coral bleaching dataset. However, there are also more words with causal relations (29.71% compared to 21.06%) and more sentences with causal relations (42.02% compared to 27.28%) in the skin cancer dataset. If we compute the average number of examples of each causal relation in each dataset, we find that on average there are 6.99 examples per unique causal relation in the coral bleaching training dataset compared to 20.30 examples per unique causal relation in the skin cancer training dataset (see Table 2.9). With almost three times as many examples to learn from on average for each causal relation, it was much easier for the models to learn from the skin cancer data. Also, the length of the causal relations are slightly shorter in the skin cancer dataset, with on average 10.00 words per causal relation as opposed to 11.17, and learning shorter dependencies is usually easier as discussed in Chapter 3. However, the sentences in the skin cancer dataset are longer and contain more unique words on average, and there are more overlapping causal relations and so the skin cancer essays are more complex in some ways. It appears that the greater number of examples per causal relation more than compensates for any additional complexities in the writing.

5.5.3 Performance By Causal Relation Frequency

To better understand the relationship between causal relation frequency and classification accuracy, Figures 5.4 and 5.5 plot the number of examples for each relation against the F_1 score from the different models. Please refer to Appendix F for the relative frequencies of each causal relation, and Appendix F for the causal relation F_1 scores by model.

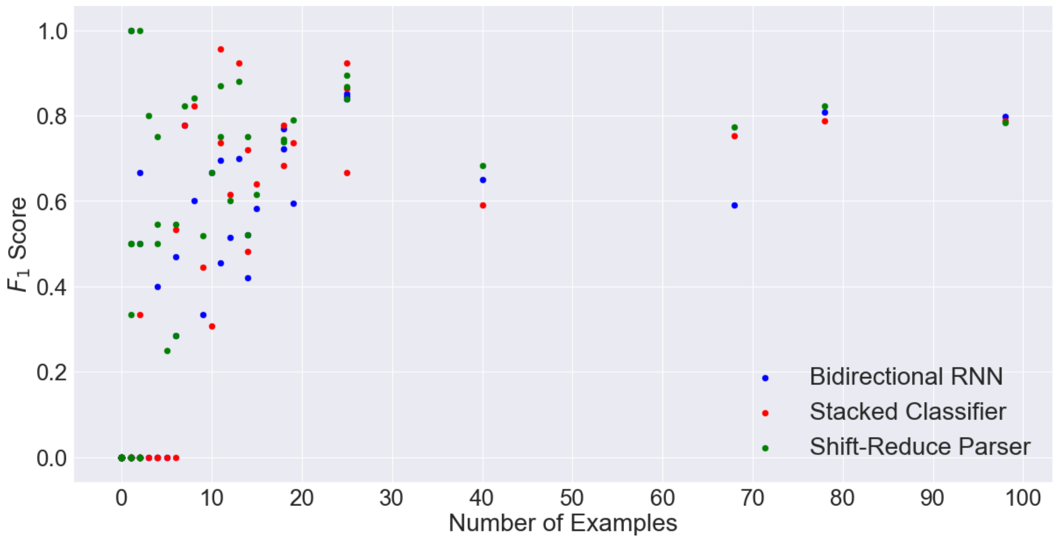


Figure 5.4: Micro F_1 Score By Frequency For the 3 Causal Relation Extraction Algorithms on the Coral Bleaching Data

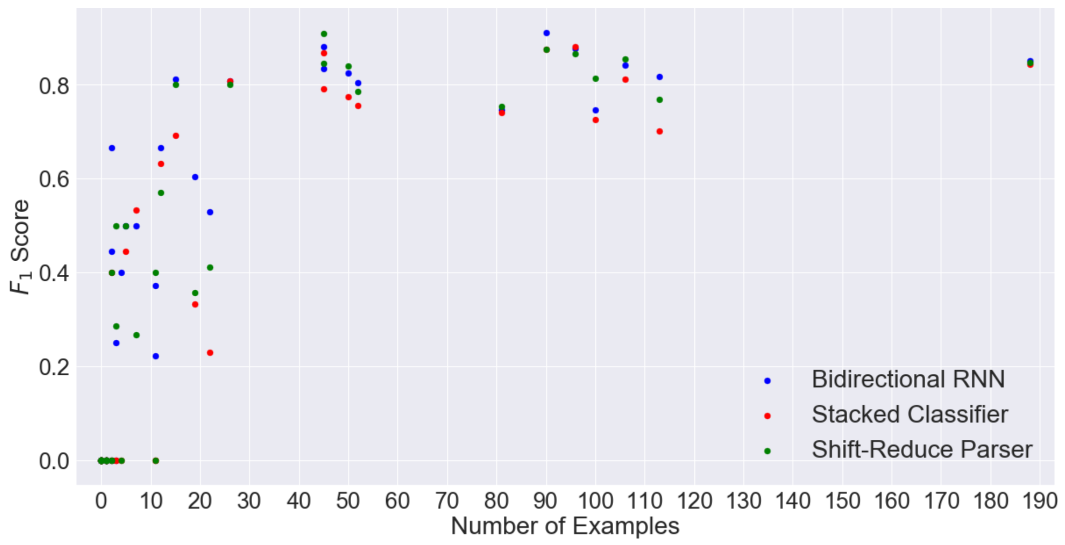


Figure 5.5: Micro F_1 Score By Frequency For the 3 Causal Relation Extraction Algorithms on the Skin Cancer Data

From Figures 5.4 and 5.5, there appears to be a correlation between the frequency of each causal relation and the model's classification accuracy. Once again (see Section 4.8.3), the Pearson correlation coefficient can be used to measure the strength of the linear correlation between two sets of variables [191]. If we examine the correlation between F_1 score and causal relation frequency, the skin cancer codes have a strong positive correlation of 0.76 while the coral bleaching codes also have a relatively strong positive correlation of 0.60. However from the two figures above, the relationship between F_1 score and the number of examples appears to be logarithmic in the number of examples because the F_1 score tapers off as more examples are present. Computing the Pearson correlation between the F_1 score and the natural logarithm of the number of examples gives even stronger correlations of 0.92 and 0.82 for the skin cancer and coral bleaching datasets respectively, indicating that this is more of a logarithmic relationship. As observed in Section 4.8.3 in the previous chapter, there is a strong correlation between each model's accuracy and the frequency of each code. The more examples of a causal relation each model has, the more accurate its predictions will tend to be, as discussed in the previous section.

5.5.4 Performance By Causal Relation Type

Another way to compare the differences in the performance of the different techniques at the causal relation extraction task is to see how their accuracy differs at predicting different categories of causal relation. Four different categories of causal relation were defined as follows:

- **Valid Relations** - Relations that contain a cause and effect that are connected in the reference causal model with the correct direction of causality e.g. $3 \rightarrow 4$. For N concept codes, there are N^2 possible causal relations, but only a subset of these are valid with respect to the reference causal model.
- **Invalid Relations** - Relations that either contain a cause and effect that are not connected in the reference causal model, or where the direction of causality is incorrect e.g. $4 \rightarrow 3$
- **Direct to 50** - Relations that contain a causal concept (with a code less than 50) that connects directly to the final causal outcome, concept code 50 e.g. $4 \rightarrow 50$
- **Adjacent Codes** - Relations where the cause and effect codes are adjacent in the causal model, and the direction of causality is correct e.g. $1 \rightarrow 2$

The sets of *Valid Relations* and *Invalid Relations* are mutually exclusive, and jointly constitute the entire set of relations. *Direct to 50* and *Adjacent Codes* are also subsets of the *Valid Relations*, and contain some overlapping relations that are in both sets. *A priori*, I would expect the models to perform better on the *Valid Relations* because they represent valid inferences and should be more consistently represented in the dataset than the *Invalid Relations*. They should also occur more frequently than the *Invalid Relations* because they are in the reference causal model and thus represent relations that can be inferred from the source text. Also, the model would be expected to perform better on the *Adjacent Codes* because the source texts include statements directly linking those concepts. Although the *Invalid Relations* represent invalid inferences not found in the source text, it is important to be able to classify these accurately to provide feedback for an intelligent tutoring system, or as input for an essay grading system.

Tables 5.25 and 5.26 below show how the differences in the performance of the 3 different algorithms vary by relation type across each dataset. In addition, each table also contains three sets of text statistics which characterize some of the differences between the 4 relation types. *Average Number of Sentences* represents the number of sentences each relation type appears in; *Average Word Span* describes the average number of words per relation of that type, and finally *Average Word Document Freq.* indicates how rare the words forming those relations are. This was computed by calculating the document frequency, i.e. the number of documents each word occurs in, and averaging this over all of the words in each relation type.

Table 5.25: A Comparison of the Micro F_1 Scores of the 3 Algorithms Across the Causal Relation Categories on the Coral Bleaching Data. Some Simple Text Statistics are Included for Each Category. Freq. = Frequency

	Valid Relations	**Invalid Relations**	**Direct to 50**	**Adjacent Relations**
Bidirectional RNN Micro-F_1	0.689	0.148	0.686	0.779
Stacked Classifier Micro-F_1	0.717	0.242	0.725	0.797
Shift-Reduce Parser Micro-F_1	**0.747**	**0.361**	**0.75**	**0.799**
Average Sentences Per Relation	69.1	4.1	152.8	112.7
Average Word Span	11.3	11.1	11.0	11.5
Average Word Document Freq.	544	499	593	526

Table 5.26: A Comparison of the Micro F_1 Scores of the 3 Algorithms Across the Causal Relation Categories on the Skin Cancer Data. Some Simple Text Statistics are Included for Each Category. Freq. = Frequency

	Valid Relations	**Invalid Relations**	**Direct to 50**	**Adjacent Relations**
Bidirectional RNN Micro-F_1	**0.805**	0.640	**0.808**	0.836
Stacked Classifier Micro-F_1	0.775	0.600	0.769	0.806
Shift-Reduce Parser Micro-F_1	0.798	**0.671**	0.799	**0.844**
Average Sentences Per Relation	245.7	17.0	391.6	310.6
Average Word Span	10.4	7.0	10.5	9.9
Average Word Document Freq.	603.2	462.5	637.5	547.6

Of the four different relation types, the *Invalid Relations* were the hardest to classify with the lowest micro-F_1 score in the two datasets, and were much harder to classify than the *Valid Relations.* The *Invalid Relations* have far fewer examples for each causal relation, and are also expressed using rarer words because the *Average Word Document Freq.* is the lowest of the 4 categories in both datasets. In the skin cancer dataset, these relations are also expressed using the fewest words, with a word span of only 7 words on average compared to around 10 words for the other relation types. The *Adjacent Relations* are the easiest category to classify in both datasets, with each model achieving its highest micro-F_1 score on that category. However, the *Direct to 50* category is the most common category, indicating that performance isn't just correlated with the frequency of the relations. The *Adjacent Relations* represent the inferences expected from the reference causal model, whereas in the *Direct to 50* relations, the student is skipping many intervening causal inferences and linking a cause directly to the final causal outcome - 'Coral Bleaching' or 'Skin Cancer'. This problem, resulting from a 'lack of coherence', is a typical problem found in some scientific causal explanations [105]. The word embeddings used in the word tagging model were initially trained on Wikipedia, and it could be that the *Adjacent Relations* contain sections of text that are closest in terms of context and meaning to the Wikipedia text, allowing the word tagging model to perform better at detecting these pairs of concepts.

Comparing the performance of the three different models, the Shift-Reduce Parser has the highest classification accuracy on the hardest category to classify, the *Invalid Relations.* As discussed earlier (see Section 5.5.1, Tables 5.21 and 5.22), this model also had the highest classification accuracy on the overlapping relations, which were much harder to classify than the non-overlapping relations, therefore this model seems to out-perform the other models on the harder types of causal relation. While the parsing model had the highest performance across all 4 categories in the coral bleaching dataset, the RNN model had a higher micro-F_1 score on the *Valid Relations* and *Direct to 50* categories in the skin cancer dataset. These two categories had the highest *Average Word Document Freq.* and thus used the most common words, and the skin cancer dataset also had more examples per causal relation on average than the coral bleaching dataset. Deep learning models typically require a large amount of labels to generalize effectively to new data points, thus the higher word frequency and larger number of labelled examples enabled the RNN model to perform better on these two categories.

5.5.5 Statistical Analysis

While overall the parsing model had the highest micro-F_1 score across both datasets, it is important to determine whether the differences in performance of the different models is statistically significant. Following the approach outlined in Chapter 4, in Section 4.3, Cochran's Q test was applied to the predictions of the three different models. For the Coral Bleaching dataset, a χ^2 value of 23.1 was attained, which has a p-value of < 0.001, which is significant, indicating that there was a significant difference in the accuracy of all 3 models on that dataset. Using McNemar's test to compare the performance of the Shift-Reduce Parser with the other two models produces the results show in Table 5.27 below:

Table 5.27: Comparing the Performance of the Different Algorithms on the Coral Bleaching Dataset with the Shift-Reduce Parser Using McNemar's Test

Algorithm	F_1	**p-value vs SR Parser**
Bidirectional RNN	0.676	< 0.001
Stacked Classifier	0.704	0.693
Shift-Reduce Parser	0.728	-

In this case, the α value for the McNemar test is $0.05/2 = 0.025$, using the Bonferroni correction. While the p-value of the RNN is below this, the Stacked Classifier has a much higher p-value, indicating that the Shift Reduce Parser is not significantly better than the Stacked Classifier, but is better than the RNN model.

For the Skin Cancer dataset, a χ^2 value of 2.6 was calculated for Cochran's Q test, which has a p-value of 0.270. Thus, the null hypothesis could not be rejected for the Skin Cancer dataset, implying that there is no significant difference in the performance of the three algorithms studied. Overall, while the accuracy of the three models was high on both datasets, it does not appear that one model is significantly better than the other. While each algorithm works uses a very different technique to learn causal relations, the strong performance of the stacked classifier indicates that knowing what pairs of concepts are present in a sentence is often sufficient to determine if a causal relation exists in that sentence, at least when it comes to the Skin Cancer dataset. It is possible that all 3 algorithms are primarily learning this fact, and any additional features learned by each model individually add only incremental value to each model's prediction performance, insufficient to make a significant difference in the accuracy of the model's predictions in the Skin Cancer dataset. The added complexity of the parsing models seems useful only for the Coral Bleaching dataset, where far more unique causal relations are present.

5.5.6 Extensibility to Other Texts and Domains

Although these three models achieve a high level of classification accuracy at detecting causal relations in these two datasets, it is important to ask whether these techniques would work well on other related datasets or even different domains entirely. As discussed earlier in this chapter, one of the main challenges in extracting causal relations was the large number of unique causal relations in each dataset, and the relatively low number of examples of each relation. The micro-F_1 score was a lot lower in the coral bleaching dataset than the skin cancer dataset because there were more labels to learn, and each label subsequently had fewer examples on average. Once there are less than around 25 examples for a label, the F_1 score varies greatly by relation (see Figures 5.4 and 5.5), and the accuracy tends to drop quite significantly. For domains where there are fewer than 25 examples per label, or where the complexity of the text was much higher than the text used in this research, I would not expect these techniques to perform as well. Similarly, in texts with a large number of causal relations (such as research papers), if those labels overlap a lot within the text then these techniques would probably not be as accurate because the performance on overlapping relations was much lower than on non-overlapping relations (see Section 5.5.1, Tables 5.21 and 5.22). It is hard to say which techniques would be most impacted, as different models performed very differently on the overlapping relations in the two datasets studied.

The causal relation extraction models discussed in this chapter all require an accurate word tagging model to achieve a high level of accuracy. Thus for these approaches to be effective on text in other domains, the concept code tagging model would also have to work well in those domains. Additionally, there are many forms of argumentation that exist in essay writing that differ from the binary cause and effect relations studied in this work. For example evidence-based arguments, such as those found in social studies essays, take on a very different structure, typically starting with a claim that is then supported by one or more pieces of evidence [189]. These types of argument have a more complex structure than the binary relations studied in this thesis, and it is unlikely the techniques discussed in this chapter would be as effective at detecting these arguments. Furthermore, the models studied in this chapter are only capable of detecting causal relations that exist between concepts defined in the reference causal model. They are not capable of detecting other causal relations within the text. For these techniques to be extended into other domains, causal models would have to be created in those domains, and text annotated according to those models. Nevertheless, I do believe these techniques could be used to detect causal relations and other types of binary relations in other domains, provided sufficient examples exist for each unique relation, and an accurate concept code tagging model could be constructed.

5.6 Conclusions

In this chapter, a novel algorithm for parsing causal relations was introduced that adapts a shift-reduce dependency parser to determine causal relations between concepts. It is hoped that this parsing model could easily be adapted to detect relationships between concepts in other domains, aside from scientific essays. Given a model which can identify the occurrences of concepts within sentences, the shift-reduce parser model could be trained to detect different types of relations between them, irrespective of the subject matter of the text studied, and not limited to causal relations.

The experiments in this chapter suggest that for complex structured learning problems in general (where the output is more complex than a single label) designing an algorithm adapted to utilize the structure of the problem will out-perform more general approaches. The sample complexity of a machine learning algorithm is the number of training examples it requires to effectively learn a target function from data [232, 231]. The parser learns to solve a problem that has lower sample complexity than the problem the other algorithms are learning to solve - determining whether a causal relation exists between two concept codes. There are many more pairs of concept codes within sentences than there are examples of each individual causal relation. The parser is thus able to better generalize to new examples.

Finally, the superior performance of the parsing model on the coral bleaching dataset was dependent on the cost function used (see Table 5.20). A novel cost function was introduced for directly optimizing the micro-F_1 metric across a set of labels, the causal relations. It is well established that for accuracy metrics derived from the field of information retrieval where the metrics are not continuous and thus non differentiable (such as F_1 score), techniques that directly optimize the metric generally out-perform techniques that optimize for classification accuracy [61, 17] (see section 4.1.4). The results in this chapter further support those claims, and provide a new approach for directly optimizing the micro-F_1 score on parsing problems.

In the next chapter I will use pre-trained coreference resolution models to pre-process the training data to replace coreferences with the entities they refer to. I will then re-train the most accurate models from Chapter 4 and from this chapter to determine if this improves the classification accuracy of these approaches.

Chapter 6

Coreference Resolution

Research Question 3 asks:

> "Does the accuracy of the word-tagging and causal relation extraction models improve when coreferents are resolved using either of the following two state-of-the-art coreference resolution systems?"
>
> A The Stanford Coreference Resolution System
>
> B The Berkeley Coreference Resolution System

In computational linguistics and NLP, coreference resolution is the task of determining all words or phrases in a body of text that refer to the same entity or set of entities. For example, consider the following two sentences

"Jane bought a new car. She intends to use it to drive to work."

In the second sentence, the pronoun *she* refers to *Jane* from the first sentence and the pronoun *it* refers to Jane's *new car*. They are said to corefer to the same entity [113]. In order to understand the meaning of a sentence it is critical for any NLP system to be able to determine which entities any coreferences, such as pronouns, refer to. *Anaphora* is the most common type of coreference and refers to expressions that refer to an entity occurring earlier in the document, which is called the *antecedent* [113, 139]. While there are other types of coreference, such as *cataphora* where the entity occurs later in the document, anaphora is the only type of coreference labels that are in the essay dataset, and so will be the focus of the work in this chapter.

Coreference resolution is an active area of research within the academic NLP community, and a number of publicly available coreference resolution systems exist. At the time of writing, the two most accurate publicly available coreference resolution systems are the Stanford Coreference Resolution Parser, and the Berkeley Entity Resolution System, which performs coreference resolution in addition to a number of other entity resolution tasks. The Berkeley coreference resolution system, described in [59], achieved an F_1 score of 61.71 on the CoNLL shared task, which was state-of-the-art when the research was published in 2014. Two years later in 2016, the Stanford neural coreference resolution parser improved on the Berkeley model with an F_1 score of 65.73 on the same dataset and task [31]. Both of these systems have been chosen for this task because the models are publicly available (they can be downloaded from [90] and [89]) and both have achieved high classification accuracies on a number of important coreference resolution benchmarks.

A coreference resolution parser typically takes as input a text document, and returns all the coreference chains the parser found within the document. A coreference chain is a list of spans (a sequence of one or more words) or 'mentions' that the parser has determined refer to the same entity within that document. Coreference resolution parsers do not typically distinguish between anaphora and other types of coreference, and attempt to resolve all coreferents in a document. To address this research question, both parsers will be used to predict the coreference chains present in each essay. The coreference chains will then be used to cross-reference the concept codes that were predicted by the RNN word tagging model, described in Chapter 4, to determine the concept codes assigned to each coreferent in a chain. In other words, when a coreferent of an already coded concept is found, that concept's code will be used for the coreferent. The causal relation parser from Chapter 5 was originally trained on predicted concept codes without resolving coreferents and therefore needs to be re-trained on the new set of predicted concept codes which includes resolved coreferents. This enables the parser to detect causal relations involving coreferents.

Extending Figure 5.1 from the previous chapter, Figure 6.1 below illustrates the coreference resolution problem. In this figure, 2 additional causal relations are added to Figure 5.1, the first between codes 3 and 4 and the second between codes 5 and 7. Both of these relations involve coreferents that refer to concepts mentioned in the previous sentence.

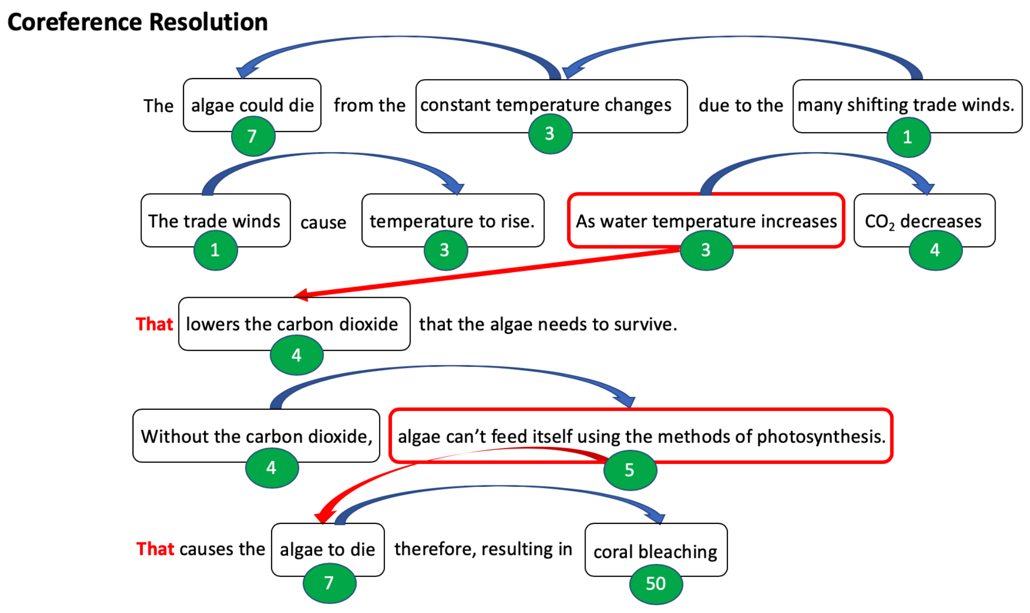

Coreference Resolution
The
algae could die
7
from the
constant temperature changes
3
due to the
many shifting trade winds.
1
The trade winds
1
cause
temperature to rise.
3
As water temperature increases
3
CO_2 decreases
4
That
lowers the carbon dioxide
4
that the algae needs to survive.
Without the carbon dioxide,
4
algae can't feed itself using the methods of photosynthesis.
5
That causes the
algae to die
7
therefore, resulting in
coral bleaching
50

Figure 6.1: Resolving coreferences involving concept codes. Each box represents a separate concept, with the code denoted by the green circle. The cause-effect relations are indicated by the blue arrows, each arrow going from causer to effect code. Causal relations resolved by coreference resolution have red arrows, with the coreferences using bold red text.

6.1 Anaphora Annotations and Coreference Resolution

As part of the annotations process (see Chapter 2 for a more detailed description), words forming anaphoric references were also labelled with an 'anaphora' tag to denote that they formed part of an *anaphoric* reference, along with the associated *antecedent* which was being referenced. Only anaphoric references were labelled by the annotators, other types of coreference were not labelled, and the annotators were instructed to focus only on anaphora that referenced instances of concept codes. As a result, all anaphoric references in both datasets have concept codes associated with their antecedents, and some of these concept codes are also part of causal relations involving those coreferents. Section 2.5 in Chapter 2 describes the relative frequency of anaphoric references, which in general occur rarely within the body of the text, but occur much more frequently within concept codes and causal relations.

Occurrences of concept codes and causal relations including anaphoric references were omitted from the experiments described in Chapters 4 and 5 to simplify the approaches to solving those problems. Incorporating a coreference parser would have prevented an evaluation of the accuracy of each technique at only detecting concept codes or causal relations because the accuracy would also be affected by the performance on the coreference resolution task.

6.2 Experimental Design

The two different coreference parsers discussed above do not differentiate between anaphora and other types of coreference, and generate coreference chains containing all detected occurrences of coreferences in a document. In order to detect anaphoric references, a process was needed to filter the predicted coreference chains to select just the anaphoric references. To solve this problem, a separate word tagging model was trained to predict which words were annotated as anaphora in each dataset, and the predictions from this model were used to filter the coreference chains to only include mentions where one or more words were tagged by this model. In addition to this method, a number of different heuristics were evaluated for filtering the coreference chain to only anaphoric references. This included filtering the words based on their part-of-speech tags, and only including mentions from the chain that contained fewer than a given number of words in length. The risk with this approach is that it involves combining a number of different models and systems, none of which have perfect accuracy, and thus the errors from these different approaches may compound as they build on one another. Therefore a second, simpler approach was also evaluated where the most recently predicted concept code was selected as the antecedent for anaphors classified by the anaphora tagging model.

Once the coreferences were obtained, the predictions from the optimal word tagging model developed in Chapter 4 were used to assign concept codes to the anaphoric references using the concept codes predicted for their antecedents. The shift-reduce parser model from Chapter 5 was then retrained with these additional predicted concept codes to extend the model to detect causal relations that also included anaphoric references. As in the previous two chapters, the same training and test dataset partitions were used in this chapter. Please refer to Section 4.2 and Figure 4.2 for more details about how the different data partitions were defined.

6.3 Evaluation Metrics

To evaluate the accuracy of these techniques at only detecting anaphoric concept codes or causal relations, the micro-F_1 score, precision and recall were computed for only the anaphoric concept codes and causal relations, in addition to computing it for all codes and relations.

6.4 Pre-Processing

The same initial data pre-processing was applied to the essay data as described in the previous 2 chapters. Some additional pre-processing was required to work with the 2 different coreference parsers. The Stanford parser uses its own word tokenizer and sentence parser, and so the coreference output was processed using a custom Python script to match it back to the original essays. For the Berkeley parser, the data was prepared into the CoNLL skeleton format [1], which allows us to pre-tokenize the words and sentences prior to passing the tokens to the coreference resolution system.

6.5 Predicting Anaphors

The two coreference resolution systems used in this chapter resolve coreferents, which need to then be filtered down to only include anaphoric references and exclude other types of coreference. For this task, a machine learning model was trained as a binary classifier to predict which words or phrases, such as pronouns, were labelled as anaphors that refer to a concept code. This model could then be used to filter the coreference chains to just anaphora tags, and does not depend on the coreference resolution systems in any way. A 2-layer bidirectional RNN word tagging model, using GloVe embeddings [88] was selected for this task because it achieved the highest overall classification accuracy across both datasets for the word tagging task (see Chapter 4). The model was trained as described in section 4.7.5, using early stopping to determine the optimal number of epochs to use to train the model without overfitting. The F_1 score, precision and recall for this model on each dataset can be seen in Table 6.1 below. Note that the micro-metrics were not calculated as only one label was predicted (Anaphora or not).

Table 6.1: RNN Classification Accuracy for Detecting Anaphors in the Coral Bleaching and Skin Cancer Datasets

	Coral Bleaching			**Skin Cancer**		
Data Partition	F_1	**Recall**	**Prec.**	F_1	**Recall**	**Prec.**
Training Data (CV)	0.597	0.579	0.616	0.624	0.500	0.830
Validation Data (CV)	0.312	0.294	0.332	0.355	0.258	0.570
Test Data	0.358	0.324	0.400	0.473	0.505	0.445

The accuracy of this model is much lower than the other word tagging models, which achieved micro-F_1 scores of 0.81 or higher for detecting concept codes. This implies this is a more difficult classification task than predicting concept codes, which is not surprising because the model is trying to match against all of the different concept codes while looking for words with little semantic content, such as pronouns.

In addition, the model performed better on the skin cancer dataset. This dataset contained more anaphoric references overall and so the larger number of training data points helped the model to generalize better on that dataset. In each dataset, the performance was also higher on the test dataset than the validation dataset, indicating that the model generalizes better with a larger training dataset.

6.6 Predicting Anaphor Concept Codes

Two different approaches were applied for determining the concept code associated with an anaphor. The first approach uses the two coreference parsers described in the previous sections to determine the antecedent for an identified anaphor. However this is quite a complex approach involving multiple separate models that were trained on very different tasks. Subsequently, a second simpler method was also evaluated which uses the most recently predicted concept code as the antecedent to see if this performs any better than the first method. Both approaches use the anaphor tagging model described in the previous section to first determine the anaphors. These two techniques are described in more detail in the following sections.

6.6.1 Coreference Resolution for Resolving Antecedents

A number of different approaches were evaluated for filtering the coreference chains to select just anaphoric coreferents. Each filtering approach was evaluated in combination with all other approaches, including using the anaphora tagging model described in the previous section, and a grid search was performed on the training data to determine the best combination of filters along with the optimal parameters for each filter. The best combination of filters was then applied to determine the anaphoric references in the test data.

Because anaphors are often pronouns, the part-of-speech (POS) tags of each word were used to create two sets of filters for the coreference chains. One set of POS filters used only coreferents that consisted of either pronouns, determiners or both, while a second set filtered the antecedents, only selecting antecedents that consisted of only noun phrases, verb phrases or both. The POS tags were assigned by the Stanford or Berkeley parser as part of the coreference resolution process. Some of the coreferents identified by the parsers were very long, containing long multi-word phrases, however most anaphoric references are typically very short, consisting of one or two words. Subsequently, another set of filters filtered out sections of the coreference chain based on the length of each section using values of 1, 2, 3, 5, 10 and 20 words, as well as no length filter. One set of these filters filtered the coreferents themselves, ignoring all coreferents with more than the specified number of words, while a second set filtered out the possible antecedents in the same way. The final type of filter used only the previous coreferent in the chain as the antecedent, as opposed to using all coreferents occurring earlier in the chain.

The filters were first applied to the coreference chain to select any anaphors in the sentence. Once a coreferent passed the first filter, its antecedents were selected from the chain, passed through the second set of filters and any remaining antecedents were used to assign concept codes predicted by the word tagging model to the anaphor. Any concept codes that were assigned to at least one word in each antecedent were then assigned to the word or words comprising the anaphor.

A summary of the results achieved on the training data can be seen in Table 6.2 below. This represents the accuracy of each approach at detecting only concept codes that were also labeled as anaphora. ‘Anaphor Filter?’ in the table describes whether or not the RNN Anaphor tagging model described in the previous section was used as one of the coreferent filters. The Stanford parser performed better on the Coral Bleaching dataset whereas the Berkeley coreference system performed better on the Skin Cancer dataset, indicating that the performance of each system is highly dependent on the dataset being evaluated and how similar it is to the dataset the parser was trained on originally. The RNN anaphor tagging model improved classification accuracy in every case except for when the Berkeley parser was used on the Skin Cancer dataset, where the recall was much lower. This implies that in that one case the model excluded too many positive examples by being more selective, improving precision at the cost of recall, but overall seemed to improve the performance of this approach.

Table 6.2: Training Data Classification Accuracy for Detecting Anaphor Concept Codes using the Optimal Set of Filters. All Metrics are Micro Metrics.

Parser	**Anaphor**	**Coral Bleaching**			**Skin Cancer**		
	Filter?	F_1	**Recall**	**Prec.**	F_1	**Recall**	**Prec.**
Stanford	Yes	0.038	0.020	0.292	**0.126**	**0.070**	**0.673**
Stanford	No	0.033	0.035	0.032	0.096	0.059	0.259
Berkeley	Yes	**0.059**	0.032	**0.380**	0.036	0.019	0.300
Berkeley	No	0.035	**0.073**	0.023	0.044	0.047	0.042

The optimal set of filters for both datasets are listed in Table 6.3 below. In the table, 'Nearest Ante.?' describes whether the nearest antecedent from the chain or all previous antecedents were used. A '-' indicates that using no filter was optimal. In every case, using only the nearest antecedent in the coreference chain improved the accuracy of the results. The columns 'Max Anaphor' and 'Max Ante.' in the table describe the maximum number of words allowed in the anaphor or the antecedent for that coreference to be used by the algorithm. Using a length filter on the anaphors did not improve classification accuracy in either dataset. This is because the anaphora tagging model is already restricting the length of the anaphors, meaning additional length filters do not make a difference. However, restricting the antecedents to a maximum of 10 words improved accuracy in the skin cancer dataset only, indicating that some antecedents in that dataset are too long.

The columns 'POS Anaphor' and 'POS Ante.' in Table 6.3 describe which POS filters, if any, were optimal. Using a POS filter to filter either the anaphora or the antecedents did not improve the accuracy in both datasets, although in most cases it also did not reduce the classification accuracy. This indicates that the anaphora and antecedents filtered by the Anaphora tagging model likely already consisted of the same POS types that those filters were filtering for.

Table 6.3: Optimal Set of Filters For Each Training Dataset

Dataset	**Parser**	**Anaphor Filter?**	**Nearest Ante.?**	**Max Anaphor**	**Max Ante.**	**POS Anaphor**	**POS Ante.**
Coral Bleach.	Berkeley	Yes	Yes	-	-	-	-
Skin Cancer	Stanford	Yes	Yes	-	10	-	-

While the micro-precision and micro-recall are both low when not filtering to the predicted anaphor tags, when that filter is applied the precision is much higher than the recall, at or above 0.29 compared to a recall that is 0.073 or lower. Given the much higher recall attained from both the Anaphora tagging model and the concept code classifier, the low recall appears to arise from using the coreference parser to determine the antecedents for each anaphor.

The micro-F_1 score for this task is much lower than was achieved on the concept code word tagging task. Detecting anaphora concept codes is a much harder task that detecting concept codes alone. The antecedents for the anaphora can occur anywhere from a few words to several sentences earlier in the text. Also, the anaphoric references lack the semantic information that can be used to identify the specific concept code they refer to. Coreference resolution is also far from a solved problem in NLP, with the Stanford model achieving an F_1 score of only 0.65 [31] and the Berkeley model a score of 0.62 [59] on the CoNLL 2012 dataset they were both trained on. Because this approach is dependent on the output of each of these systems as well as the output of the concept code word tagging model from Chapter 4, the best accuracy that can be attained from this approach is much lower than the accuracy of each model when used alone. This is because the errors of each model compound when used together to solve a different problem. The two datasets evaluated are also very different from the CoNLL dataset that these parsers were trained on. The CoNLL dataset used annotated corpora from the *OntoNotes* project, which contains coreference annotations from text from transcribed telephone conversations, various news sources, web data and magazine articles [132, 133]. The Berkeley and Stanford coreference parsers were also trained to detect all types of coreference, not just anaphora which is what is being evaluated in this task.

Another aspect that makes this task so challenging is the way in which the data was annotated. The coreferents annotated in each dataset always had a concept code associated with them; the annotators did not annotate any other examples of anaphora that were present. This means that a lot of examples of anaphors are missing anaphora labels in the training data which could be causing a lot of confusion for the model. Solving this problem therefore requires solving two problems at the same time, both detecting anaphora and detecting concept codes. If these two tasks were annotated separately, two separate task-specific models could be trained directly on this dataset and then combined to determine anaphoric concept codes. This approach would likely improve performance on this specific task.

6.6.2 Resolving Antecedents using the Predicted Concept Codes

Anaphors refer to antecedents that occur earlier in the same document, and involves linking words and phrases together, while omitting some terms for brevity. Because the annotators only labeled instances of anaphors that reference a concept code, a reasonable assumption is that the code being referred to is most likely to be the most recently referenced concept code, as that is freshest in the mind of the reader and would make the text more cohesive. A simpler approach to solving this problem is then to use the anaphora tagging model described in section 6.5 to determine which words are anaphors, and then use the most recently predicted code from the word tagging model as the concept code of the antecedent. By reducing the complexity of the solution, this approach could produce more accurate results.

To test this idea, the RNN anaphora word tagging model was used to determine which words were anaphors, and then the most recently predicted concept code was used as the code for that anaphor. The concept codes were predicted by the optimal word tagging model from Chapter 4. This simpler approach was much more accurate than the previous approach, which can be seen from the classification metrics in Table 6.4 below. This table also shows the impact of including this approach on the overall word classification accuracy. ‘Anaphora Codes’ shows the model’s performance on just the Concept Codes that were labeled as anaphors, ’Concept Codes’ shows the model’s performance on concept codes excluding those with anaphora tags, and ‘Both’ shows the performance on all concept codes, including anaphora concept codes.

Table 6.4: Validation Data Classification Accuracy for Detecting Only Anaphor Concept Codes ('Anaphora Codes'), Only Concept Codes and On Both Sets of Codes. All Metrics are Micro Metrics.

	Coral Bleaching			**Skin Cancer**		
Predicted Labels	F_1	**Recall**	**Prec.**	F_1	**Recall**	**Prec.**
Anaphora Codes	0.262	0.241	0.287	0.235	0.167	0.399
Concept Codes	0.837	0.822	0.853	0.821	0.821	0.822
Both	0.832	0.816	0.848	0.815	0.811	0.819

The metrics attained by this approach for the anaphora codes are close to those achieved from the anaphora tagging model alone when applied to the validation or test data (see Table 6.1). This implies that in most cases, the most recent concept code is the one referenced by the anaphor. This approach performs better on the coral bleaching dataset than the skin cancer dataset. This is surprising for two reasons. Firstly, the skin cancer dataset has around 50% more instances of anaphora than the coral bleaching dataset (see Table 2.10) and so you would expect it to be easier to classify. Secondly, the anaphora tagging model is more accurate than the coral bleaching model (see Table 6.1). However, the accuracy of the coreference resolution approaches are dependent on the accuracy of the concept code tagging model as they rely on the predictions from that model to resolve the coreferences. The concept coding model was more accurate on the coral bleaching dataset and is probably the reason that this approach is also more accurate on that dataset. Note that this also applies to any other NLP topics and tasks that also depend on first identifying conceptual information; how well the model performs at identifying concepts will impact the accuracy of other tasks dependent on identifying those concepts.

6.6.3 Results

The performance of the 2 Coreference Resolution systems (using the optimal set of filters) are compared with the heuristic approach in Table 6.5 below. These results mirror the results on the validation data; while the Stanford or Berkeley approaches may achieve the higher recall or precision on each dataset, in each case the overall micro-F_1 score is lower as the combination of recall and precision is much lower than the heuristic approach.

Table 6.5: Test Data Classification Accuracy for Detecting Only Anaphor Concept Codes Using Both Coreference Resolution Models as Well as the Heuristic Approach. All Metrics are Micro Metrics.

	Coral Bleaching			**Skin Cancer**		
Algorithm	F_1	**Recall**	**Prec.**	F_1	**Recall**	**Prec.**
Stanford + Filters	0.038	0.020	0.292	0.126	0.070	**0.673**
Berkeley + Filters	0.059	0.032	**0.379**	0.036	0.019	0.300
RNN Tagger + Heuristic	**0.324**	**0.282**	0.287	**0.235**	**0.167**	0.399

6.6.4 Statistical Analysis

When comparing the performance of different algorithms on solving a machine learning problem, it is important to determine whether differences in accuracy are likely due to chance or not. Statistical testing can help answer this question. Following the approach outlined in Section 4.3 in Chapter 4, I ran Cochran's Q Test to test the null hypothesis, which states there is no significant difference between the performance of each algorithm at the current task (resolving coreferences). A χ^2 value of 2.1, with an associated p-value of 0.354 was attained when comparing model predictions on the coral bleaching dataset. This is above the significance threshold of 0.05, and thus the null hypothesis cannot be rejected for the coral bleaching dataset. According to the results of Cochran's Q test, there is no significant difference between the 3 approaches, despite the relatively large differences in micro-F_1 score. Because this test focuses on disagreements between the models, this implies that most of the time, the 3 approaches make the same mistakes, and get the same data points correct.

Running Cochran's Q test on the skin cancer dataset however does produce a significant result; a χ^2 value of 15.8 which has an associated p-value of < 0.001. This is below 0.05, and the null hypothesis can be rejected. There is a statistically significant difference between the performance of the algorithms on the Skin Cancer dataset. Running McNemar's test to compare the performance of the 2 coreference resolution systems with the heuristic approach produces the results in Table 6.6 below:

Table 6.6: Comparing the Performance of the Coreference Resolution Parsers on the Skin Cancer Dataset with the Heuristic Approach Using McNemar's Test

Algorithm	F_1	**p-value vs RNN + Heuristic**
Stanford + Filters	0.126	0.006
Berkeley + Filters	0.036	0.003
RNN Tagger + Heuristic	0.235	-

In this case there are just 2 comparisons, so the α value is corrected to 0.025. According to McNemar's test, the heuristic approach is significantly better than the two coreference resolution parsers on the skin cancer dataset. It is surprising that there is a statistically significant difference between the heuristic approach and the other approaches on the skin cancer dataset but not the coral bleaching dataset, given that the heuristic approach's overall micro-F_1 score is higher on the coral bleaching dataset. This again reflects what the statistical tests are measuring, that is how often one algorithm is correct when it disagrees with another algorithm, and not the overall accuracy of each approach.

6.6.5 Impact on Overall Word Tagging Performance

Table 6.7 below shows the impact of incorporating the best coreference resolution approach (the heuristic) on the accuracy of the concept code labeling problem when all concept codes are included in the test data.

Table 6.7: Test Data Classification Accuracy for Detecting Only Anaphor Concept Codes ('Anaphora Codes'), Only Concept Codes and On Both Sets of Codes. All Metrics are Micro Metrics.

	Coral Bleaching			**Skin Cancer**		
Predicted Labels	F_1	**Recall**	**Prec.**	F_1	**Recall**	**Prec.**
Anaphora Codes	0.324	0.282	0.379	0.295	0.262	0.337
Concept Codes	0.842	0.830	0.855	0.837	0.807	0.869
Both	0.840	0.827	0.853	0.829	0.799	0.862

In both datasets, the test data micro-F_1 score for the anaphora codes and for both sets of codes was higher than on the validation dataset. The anaphora tagging model performed better on the test dataset because it had a larger training dataset to learn from, as we have seen with the other machine learning models in the previous chapters. This is likely the reason for the higher test performance on this task. As with the validation data, the classification accuracy is higher on the coral bleaching dataset than the skin cancer dataset. If we include all of the instances of each concept code, not just those with anaphoric references (labeled 'Both' in the table), we see that incorporating the anaphora resolution approach lowers the overall concept code classification accuracy on both validation and tests datasets. However, the overall impact on these metrics is small due to the relatively small number of concept codes with anaphora tags.

The impact on the coral bleaching dataset of including the anaphora resolution logic is much smaller than on the skin cancer dataset, resulting in a drop of only 0.02 compared to 0.08 on the test dataset. This is because there are fewer words and concept codes with anaphora tags in the coral bleaching dataset (see Table 2.10). Also, there is a smaller number of anaphora tags in the test dataset partition for the coral bleaching dataset compared to the skin cancer dataset, with only 39 words with anaphora tags compared to 107 in the skin cancer dataset. So the lower accuracy and larger number of test data points resulted in a larger decrease in the overall micro-F_1 score for the skin cancer dataset.

6.7 Predicting Anaphor Causal Relations

In order to predict causal relations that include anaphora, the predictions from the most accurate concept code anaphora resolution approach were used to generate the concept code labels, and then the shift-reduce parser model was re-trained on this new dataset to resolve the additional causal relations that included anaphoric references. Please see section 5.4.3 for a more detailed description of the shift-reduce parser model for causal relation detection.

The micro metrics for the validation datasets (using cross-validation) and the test datasets can be found in Tables 6.8 and 6.9 below. 'Anaphora Only' refers to only the causal relations including anaphors, and 'Causal Relations' refers to the model's accuracy at predicting all causal relations except the anaphor relations. 'Both' refers to the overall accuracy of the model at predicting all causal relations, including the anaphoric causal relations.

Table 6.8: Validation Data Classification Accuracy for Detecting Causal Relations With and Without the Anaphora Causal Relations. All Metrics are Micro Metrics.

	Coral Bleaching			**Skin Cancer**		
Predicted Labels	F_1	**Recall**	**Prec.**	F_1	**Recall**	**Prec.**
Anaphora Only	0.081	0.050	0.214	0.231	0.152	0.479
Causal Relations	0.722	0.727	0.718	0.768	0.721	0.822
Both	0.700	0.682	0.718	0.748	0.688	0.819

Table 6.9: Test Data Classification Accuracy for Detecting Causal Relations With and Without the Anaphora Causal Relations. All Metrics are Micro Metrics.

	Coral Bleaching			**Skin Cancer**		
Predicted Labels	F_1	**Recall**	**Prec.**	F_1	**Recall**	**Prec.**
Anaphora Only	0.051	0.030	0.167	0.151	0.101	0.300
Causal Relations	0.728	0.766	0.693	0.790	0.760	0.823
Both	0.705	0.733	0.679	0.776	0.737	0.819

Despite being much more accurate at detecting the anaphora concept codes on the coral bleaching dataset, the causal relation parser model is more accurate at detecting causal relations including anaphora on the skin cancer dataset. Overall, the parser model had a much higher classification accuracy on the skin cancer dataset, and this higher accuracy seems to translate to the anaphora causal relations also. Please refer to Chapter 5 for a more detailed discussion on the difference in causal relation extraction performance across the two datasets. Similar to the concept code tagging results, the precision on the anaphoric causal relations is much higher than the recall for both datasets on both the validation and test data. The causal relation model is also dependent on the predictions from the concept code model, and the higher precision is likely a result of that model also having higher precision than recall for both datasets. It is common in NLP classification tasks for precision to be higher than recall. The main reason for this is the large variety in the vocabulary used to describe different concepts and entities in textual data. Out-of-vocabulary words that occur in the test data but are missing from the training data can cause a high false negative rate (and thus a low recall) in the test dataset, as the model is unable to correctly classify words or phrases that it has not seen before. The dependency on the concept code tagging model is also the reason the overall classification accuracy for both datasets is low, because any additional errors introduced by the causal relation parser further compound the errors from the concept code labeling task. In addition, very few examples of anaphoric causal relations exist in the dataset with only around 7% of causal relations including anaphora (see Table 2.10), making it hard for the model to generalize well on these examples.

Looking at the classification metrics for all of the causal relations, including the anaphoric causal relations again lowers the overall micro-F_1 score due to the much lower classification accuracy achieved on this task. This approach therefore does not provide the improvement in overall classification accuracy that was expected. The inclusion of these additional causal relations has a lesser impact on the skin cancer F_1 score than on the coral bleaching F_1 score because the F_1 score on the anaphora relations is almost three times higher while the number of causal relations containing anaphora is about the same for both datasets.

6.8 Summary and Discussion

The use of pre-trained coreference resolution models was not effective at resolving concept codes associated with anaphora when combined with the concept code word tagging model, contrary to my expectations when conceiving this approach. This is due to some large differences between the training data used to train the coreference resolution models and the two datasets studied here, and because of differences in how the coreferents were labeled in these two essay datasets. Coreferents in this task did not include other coreferents there were not anaphors, and only included anaphors that were referred to entities that were part of the causal models defined for each task. Subsequently, a simpler approach combining an anaphora labeling model with the concept code labeling model, both trained separately on the essay data, out-performed the coreference resolution parser approach. However even with this approach, a low overall micro-F_1 score was obtained due to the difficulty of this task and the low number of labeled examples in the training data. These results carried over into the causal relation model, which is dependent on the predictions from the concept code tagging model, and which also achieved a low classification accuracy on the anaphoric causal relations. This implies that for any system that is trying to learn relationships between concepts, how well that system identifies concepts is critical to the overall performance of the system.

6.9 Extensibility to Other Texts and Domains

Due to the very low accuracy of the different techniques applied in this chapter, it is very hard to predict how well they would perform in other domains. It is possible that if a similar dataset of essays was collected (and annotated according to a causal model) where all of the anaphors in that dataset were labeled, that the anaphor tagging approach combined with the heuristic could more successfully resolve anaphors. This would in turn improve the accuracy of the parsing model at detecting causal relations. However, it also unlikely that such a simple heuristic of assigning the most recent code to the antecedent would work in other domains, aside from annotating scientific explanatory essays. There are many different ways in which anaphors are used to refer to entities occurring earlier in texts. This heuristic was quite accurate in this case because the students were focused on the narrow tasks of explaining the causality of some scientific phenomena. As they constructed a causal chain concept by concept, it was natural for them to use anaphors that referred to antecedents which where concepts earlier in that chain. For texts involving more complex arguments, such as evidence-based essays or arguments presented in non-scientific text, this assumption may not hold. I therefore believe it is unlikely that the heuristic would work well for domains other than scientific explanatory essays.

As discussed in this chapter, the coreference resolution models did not generalize well to this domain because the essays differed in semantic content from the texts used to train the coreference models. Applying these systems to resolve coreferences in other texts in similar or even different domains might produce very different results if the texts have much more semantic overlap (with the texts used to train these models). However, if the semantic overlap in these texts is also low (compared to the model's original training data), then I would expect to also see a low coreference resolution accuracy.

6.10 Conclusions

In NLP problems, it is common to use pre-trained models such as part-of-speech taggers or dependency parsers to pre-process text or extract features when building NLP models. In this chapter, two coreference resolution models were used to resolve coreferents within the essay text to parse causal relations involving anaphoric references. In this case, these systems were ineffective at resolving coreferents because the underlying models were trained on very different corpora to the scientific essay text studied. In Chapter 4, a different type of pre-trained model, a language model, was used to create word embeddings, which were then used within an RNN tagging model to improve its performance on the word tagging problem. In that case, the word embeddings were effective at improving the model's performance on that task. However, the embeddings were trained on a semantically related corpus, Wikipedia, which contained some scientific content. This implies that for NLP problems in general, using pre-trained models trained on one corpus to solve problems in a different corpus is only effective when the two corpora are semantically related. This is because the vocabulary used can differ substantially from one corpus to another, making generalization across different corpora very challenging. When no models exist that have been trained on related content, building a much simpler domain specific model may present a better solution to the problem, as we saw in this chapter.

While around 7% of causal relations in the dataset involved coreferences in the form of anaphora, around 4-5% of causal relations spanned multiple sentences without any form of explicit anaphoric reference (see Table 2.10 in Chapter 2). It was common for the students writing the essays to construct a causal chain over the course of several sentences, each sentence building on the preceding sentences, and the claims made within them. State-of-the-art coreference resolution parsers, such as the Stanford and Berkeley parsers used in this chapter, make use of information from the whole document when determining all of the coreferents of some entity. Information from the entire essay should therefore also be used when constructing a causal chain, without limiting the model to the information present only in the current sentence being classified. In the next chapter, I will extend the techniques described in this chapter and Chapters 4 and 5 to utilize information from the entire essay when constructing a causal model for each essay.

Chapter 7

Inferring a Causal Model of an Essay

Research Question 4 asks:

> Can a more accurate causal model of an entire essay be constructed by using information from the whole essay using either:
>
> A A Transition-Based Parsing Model to parse the entire essay
>
> B A Reranking Model to rerank possible causal models

As discussed in section 2.5, it was common for students to develop an explanation of the causal phenomena over several sentences. The methods for detecting causal relations described in Chapter 5 were limited to examining features in the current sentences only, and did not have access to information from the rest of the essay. In this chapter, I examine several different techniques for generating complete causal models from science essays, leveraging the structure of the whole essay to improve the accuracy of the causal relation detection. This is illustrated in Figure 7.1 below, which shows an example of an essay that has undergone word tagging, causal relation extraction and coreference resolution to produce a final causal model, representing the causal structure of the whole essay.

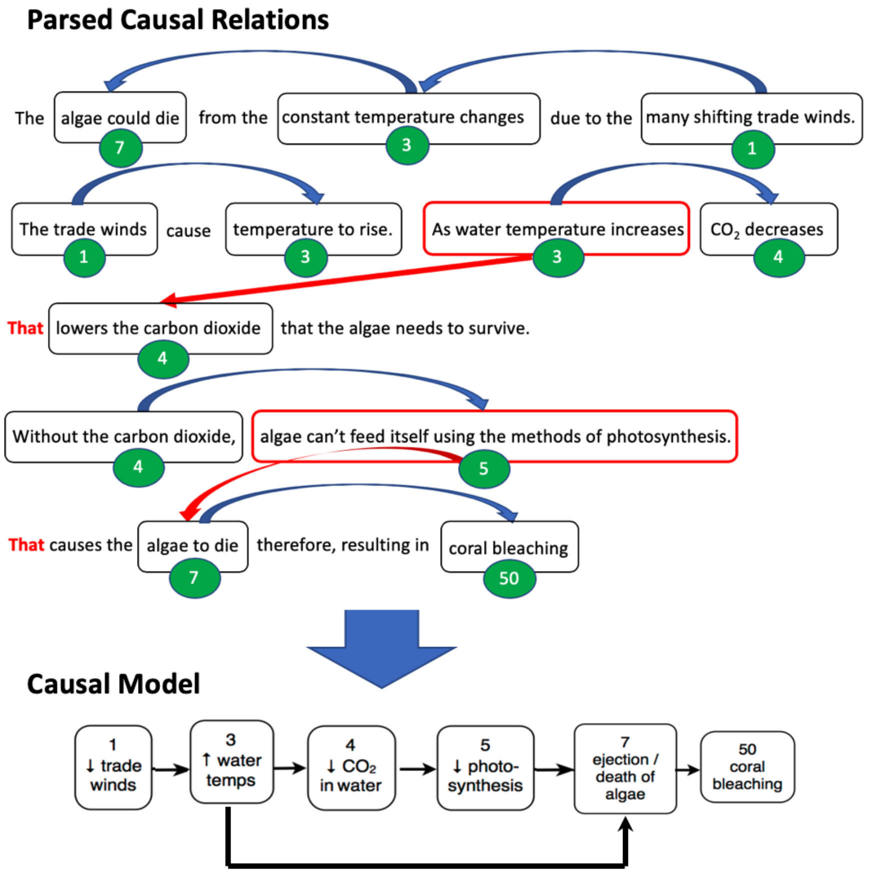


Figure 7.1: Generating a causal model from parsed causal relations. Each box represents a separate concept, with the code denoted by the green circle. The cause-effect relations are indicated by the blue arrows, each arrow going from causer to effect code. Causal relations resolved by coreference resolution have red arrows, with the coreferences using bold red text.

There are two principle ways that causal relation extraction could be extended to operate at the essay level to determine the full causal model for an essay. The first method is to build an essay parser to parse the essay sequentially, making use of both local and global features to inform each parsing decision. The second method is to first determine all of the causal relations likely to be present in an essay, and then select which subset of these causal relations form the most probable causal model, given the training data.

The first method was implemented by extending the existing Transition-Based Shift-Reduce Parsing Model to parse each essay in a single parse, instead of parsing each sentence individually and assuming independence between sentences. To implement the second approach, I introduce a novel generate and rerank algorithm, building upon the reranking approaches applied to structured learning problems, described in Section 3.4.6. The sentence parsing model developed in Chapter 5 was extended using a beam search to produce multiple different parses for each sentence. A set of candidate causal models were then generated by taking combinations of these parses across all sentences in the essay. A separate reranking model was then trained to select the best candidate causal model created from a subset of these relations. By controlling the beam size of the beam search, this algorithm presents a novel method of balancing recall and precision to better optimize the micro-F_1 score relative to the sentence and essay parsing models.

In the rest of this chapter, I will describe each of these algorithms in more detail, and compare their relative performance to the sentence parsing model from Chapter 5 to determine whether building a global model for the whole essay is more effective for solving this problem than a local model operating at the sentence level.

7.1 Evaluation Metrics

Similar to Chapter 5, I use the micro-precision, micro-recall and micro-F_1 score metrics computed over all observed causal-relations. However because the approaches in this chapter operate on entire essays and not sentences, the metrics are calculated and compared at the essay level, and not the sentence level. Consequently, the metrics only evaluate the set of unique causal relations extracted from each essay, ignoring the frequency and location of each distinct causal relation within each essay. Some casual relations span multiple sentences and were counted more than once when evaluating the sentence level classification accuracy. Evaluating unique causal relations therefore also addresses this issue.

7.2 Experimental Design

The same training and test datasets were used as in previous chapters (please see Section 4.2 and Figure 4.2 for a explanation of the data set partitions). However, because the metrics were computed at the essay level, the essay level accuracy of the existing parsing model from Chapter 5 were computed for comparison to the approaches used in this chapter.

7.3 Pre-Processing

The same pre-processing was performed on the essays as described in the previous chapters. In addition, the concept codes for each word were predicted using the parsing model from Chapter 5 and then augmented using the optimal coreference resolution approach from Chapter 6. For the training and validation datasets, the predictions were made using 5 fold cross-validation; the essays were partitioned into 5 different groups, and the predictions for each group were made using models trained on the remaining 4 groups. For the test dataset, the models were first trained on the training dataset and used to make predictions on the test dataset.

7.4 Model Evaluation

The goal of this chapter is to extend the causal relation parsing model to make use of the structure and content of the whole essay in order to improve upon the accuracy of the sentence parser. The essay level classification accuracy metrics for the sentence parser are listed in Tables 7.1 and 7.2 below, and act as a baseline for evaluating the approaches discussed in this chapter:

Table 7.1: Baseline - Shift Reduce Sentence Parser Causal Relation Classification Accuracy on the Coral Bleaching Dataset. All metrics were calculated at the essay level.

Data Partition	**Micro-F_1**	**Micro-Recall**	**Micro-Precision**
Training Data (CV)	0.783	0.762	0.805
Validation Data (CV)	0.744	0.729	0.759
Test Data	0.737	0.766	0.710

Table 7.2: Baseline - Shift Reduce Sentence Parser Causal Relation Classification Accuracy on the Skin Cancer Dataset. All metrics were calculated at the essay level.

Data Partition	**Micro-F_1**	**Micro-Recall**	**Micro-Precision**
Training Data (CV)	0.843	0.795	0.898
Validation Data (CV)	0.810	0.764	0.861
Test Data (CV)	0.827	0.800	0.856

The essay level parsing accuracy is noticeably higher on both test datasets than the sentence level accuracy, with the micro-F_1 score increasing from 0.728 to 0.737 on the coral bleaching dataset, and from 0.790 to 0.827 on the skin cancer dataset. On the coral bleaching data, the improvement in micro F_1 score was due to an improvement in precision alone while in the skin cancer dataset, both precision and recall improved when aggregating the predictions to the essay level.

To understand why both recall and precision increase when computing the micro-F_1 score across only unique relations, consider the kind of mistakes the parser can make and their impact on the precision, recall and F_1 scores. Consider the scenario where the sentence parser predicts two occurrences of a causal relation when only one is present in the essay. This would result in the parser producing an additional false positive prediction, which would lower the precision but not the recall score for that particular relation (see Equations 4.1 and 4.2). However, if we only consider the unique causal relations, this additional false positive would not be counted. The precision score is thus higher for this scenario when only unique causal relations are considered. Consider a second scenario where there are two instances of a particular causal relation, but only one instance is predicted by the model. In this scenario, the model has made a false negative error by failing to predict one of the 2 relations. This impacts recall but not precision (see Equations 4.1 and 4.2). However, if we only consider unique causal relations, this additional false negative prediction would be ignored, and recall would thus be higher. Therefore by considering only unique causal relations, both precision and recall can improve if both of the error types described above are present.

For both approaches evaluated in this chapter, features were extracted that encoded the concept codes and causal relations extracted from other sections of the essay, as well as features unique to each approach. As in previous chapters, for each model forward selection was used to select the optimal set of features, using the default settings for the model's hyper parameters. Using the optimal feature set, grid search was then used to determine the optimal set of hyper parameters. Five fold cross-validation was used to perform feature selection and hyper parameter tuning, and the optimal set of features and hyper-parameters were used to train a final model on the full training dataset, and then evaluated on the test set.

7.4.1 Transition-Based Essay Parsing Model

Three main changes were made to the transition-based parsing model to adapt it to parse each essay as a whole instead of sentence by sentence. First, the features described in section 5.4.3.2 were adjusted so that they encompassed the entire essay. For example, the *Valency* features count the number of parsed causal relations to the left and right of the two codes currently being evaluated within the current sentence. These features were subsequently extended to encompass all parsed causal relations across the whole essay occurring either side of these two codes. The second change was to include an additional set of feature templates to capture global features from the whole of the essay, as well as features from the adjacent sentences. Finally, the essays themselves were flattened from a sequence of sentences to a single contiguous sequence of words with their predicted concept codes (from the RNN model). A special token was inserted at the sentence boundaries to allow the model to learn when it was transitioning from one sentence to the next. This token was also labelled with a special concept code.

7.4.1.1 Features Evaluated

Five new sets of essay level features were evaluated for the essay parser model. The students generally constructed the essay answers sequentially, laying down a sequence of causal inferences resulting in the final causal outcome (coral getting bleached or the presence of skin cancer). Thus ideas developed earlier in the essay should influence those that come later on. The essay-level features attempt to capture this information, looking at the concept codes in surrounding sentences to see how they influence the causal relations in the current sentence, and examining all of the relations that have currently been parsed because those should influence which relations are remaining.

For these new feature sets, essay level features were extracted from either the entire essay or two different parts of the essay: the section of the essay prior to the current top of stack code, the S_0 code, and the section of the essay following the code at the front of the input stream, the I_0 code (the code at the front of the current input stream). Please refer to section 5.4.3 for a description of the shift-reduce parser and how it manipulates the stack and the input stream. Unlike the sentence-level features, most of these features are numeric and do not fit a simple template pattern, so I have not created a feature template similar to those in Tables 5.9 and 5.10.

The first set of features called *# Concept Codes* encoded the number of predicted concept codes in different sections of the essay. This set of features attempted to capture any relationship between the number of codes in an essay and the likelihood of a causal relation being parsed. These features included the total number of predicted concept codes, the number of predicted codes before the S_0 code, and the number of predicted codes remaining in the input stream. Another feature represented the percentage of all predicted concept codes that were before the S_0 token, and a second feature encoded the percentage of codes after the I_0 token.

The next set of features, called *Ratio Features*, described the ratios between the number of predicted codes, the number of parsed causal relations and the number of words and sentences in each essay. With the exception of the causal relations, separate counts were calculated for each of these items for the whole essay, as well as the part of the essay prior to the S_0 code and the section of the essay after the I_0 code. Each combination of these counts were then used to compute different ratios, each forming a unique feature. For example, one feature captured the ratio between the number of parsed causal relations and the number of predicted concept codes before the S_0 code. The purpose of this set of features was to determine if there was any relationship between these ratios and the likelihood of there being a causal relation in the current parser state, for example there might exist a relationship between the length of an essay and the number of causal relations, or between the number of concept codes and the number of causal relations.

It was common for the students to create a causal chain over the course of several sentences, or even to define causal relations that span several sentences. The next set of features captured information about the predicted codes in the previous and next sentence to help the model leverage this information. I will refer to these features as *Adjacent Sentence Codes*. The number of codes in the previous and next sentence were encoded as separate features, and for each code in each adjacent sentence, a unique bigram feature was created combining that code and its location (previous or next sentence) with the top of stack code S_0, the current input stream code I_0, and finally combined with both tokens to form a trigram feature. These features indicated to the model how inter-sentential sequences of codes influence the likelihood of certain causal relations being present in the current sentence based on how the students construct their causal arguments.

The fourth set of features were created to describe how far the parser has progressed through the essay, and is be referred to as the *Sentence Position* features. It is possible that certain causal relations tend to occur more frequently near the start, middle or end of the essays. For example, causal relations near the end of the causal chain in each causal model (see section 2.2) might occur more frequently near the end of an essay. A separate feature was created for the number of sentences before the S_0 code, the number of sentences after the I_0 code and the number of sentences between these two codes (because some causal relations span multiple sentences). Two more features described whether the S_0 and I_0 codes were in the start, middle or end of the essay, irrespective of essay length, and another feature computed what percentage of all of the essay's sentences were between these two codes.

The final set of features captured information about the causal relations that had been parsed by the parser in the essay before it entered its current parse state. These features will be referred to as *Causal Features*. The essays were written to describe the complete chain of causes and effects leading to some outcome. The presence of one causal relation therefore depends on the causal relations that have already been stated earlier in the essay, and this set of features attempts to capture this information. The number of causal relations parsed as well as the number of unique causers or effects across those relations were encoded as features. Additionally, the number and percentage of parsed relations that spanned more than one sentence formed new features. It is possible for the same causal relation to be mentioned multiple times in an essay. The number of times each causal relation has been parsed formed another set of features. Similarly, an additional set of binary features encoded which causal relations occurred more than once, ignoring the relative frequency of each relation. Usually the cause is mentioned before the effect in the essay, but sometimes the order is reversed and the effect is stated first. The proportion of causal relations where the cause preceded the effect in the essay text formed an additional feature.

The main purpose of the causal relation parser is to determine whether any two codes could be combined to form a causal relation, whether one of both should be discarded (as they are not part of any relation), or whether either of them should be stored in memory as they may participate in a causal relation with a code occurring later in the essay. In other words, S_0 and I_0 codes could be combined to form a causal relation, one or both may need to be discarded by the *Skip* transition, or the input stream token may need to be pushed onto the stack. Because this parser parses the whole essay, the S_0 and I_0 codes can span multiple sentences and are no longer restricted to being within the same sentence. One binary feature captures whether these two codes form a causal relation that has already been parsed, and another feature encodes how many times the potential relation has already been parsed. Finally, for each parsed causal relation three features were created, combining it first with the S_O code, then with the I_O code and then both codes forming bigram and trigram features. These features differ from the *Label Set* features described in Chapter 5, which encoded combinations of causal relations, stack and input stream codes with the words that constituted those codes, and were limited to only relations parsed within the same sentence.

7.4.1.2 Feature Selection

For each dataset, the model was first initialized with the optimal set of features used in the sentence parser. With five new feature sets, performing forward selection over all of the original feature sets in addition to these five would have been computationally intensive and impractical. Because most of these original features were local and not global in nature, it was assumed that the set of optimal sentence parser features were still useful for the essay parser and forward selection was performed only on the additional essay level feature sets. All word features were stemmed as before. Feature selection was again performed on sets of related features rather than individual features because there were too many individual features for model-based feature selection to be tractable. Logistic regression was used as the based model as before, and the values for the different hyper-parameters were chosen based on the optimal settings for the sentence level parser, as described in section 5.4.3.4.

To demonstrate the relative utility of the new feature sets, the micro-F_1 scores are listed in Table 7.3, where each feature set was added individually to the existing sentence level features. The most valuable essay level feature set across both datasets is the *# Concept Codes* feature set, when no other essay level features were present. This feature set produced the largest increase in micro-F_1 score in the Skin Cancer dataset, and the second largest increase in the coral bleaching dataset. Conversely, the *Causal Features* feature set which produced the biggest improvement in the coral bleaching dataset, made the skin cancer model slightly worse, and was ranked lowest in the skin cancer dataset when no other essay level features were selected. However, the change in performance in the skin cancer model from adding any of these features was very small and so the ranking of each of these features may not be a reliable indication of their relative utility. This implies instead that either any information provided by these features is already present in the sentence level features when extended to the essay level, or that essay level information was not useful in parsing the skin cancer essays. A much larger improvement in micro-F_1 score was also observed with the coral bleaching model.

Table 7.3: Each Essay Level Feature Set Ranked by Micro-F_1 Score When Used in Isolation with the Sentence Level Features. Adj. = Adjacent

	Coral Bleaching		**Skin Cancer**	
	Feature Set	F_1	**Feature Set**	F_1
1	Causal Features	0.741	# Concept Codes	0.806
2	# Concept Codes	0.740	Ratio Features	0.803
3	Adj. Sentence Codes	0.738	Sentence Position	0.802
4	Sentence Position	0.736	Adj. Sentence Codes	0.801
5	Ratio Features	0.732	Causal Features	0.800

Different features can have a different impact on a machine learning model when combined with other features than they can have when used on their own. Table 7.4 shows the order in which features were chosen by the forward selection algorithm and the impact on the overall micro-F_1 score. The first row, with 0 additional feature sets, represents the baseline performance before any additional features were added. In both datasets, the feature sets ranked the lowest when evaluated individually were the most useful feature sets once the first feature set had been selected. This implies that they did not provide enough information on their own to produce an accurate model, but added additional value once the top feature was present. In both datasets, only 2 sets of features were chosen by forward selection before the model's accuracy started to decline, indicating the model was overfitting the dataset when more than 2 essay level feature sets were added.

Table 7.4: Micro F_1 Score as Additional Feature Sets Are Added by Forward Selection. Adj. = Adjacent

	Coral Bleaching		**Skin Cancer**	
# Fts.	**Added Feature Set**	F_1	**Added Feature Set**	F_1
0	-	0.7356	-	0.8046
1	Causal Features	0.7414	# Concept Codes	0.8065
2	Ratio Features	**0.7433**	Causal Features	**0.8067**
3	Adj. Sentence Codes	0.7403	Sentence Position	0.8043

As with the previous models, we can compute the reduction in the number of features resulting from the feature selection process. Table 7.5 below shows the total number of additional features added to the sentence parser in developing the essay parser, and compares this with the number of features chosen by feature selection. The number of features were computed for the validation dataset and averaged across all cross-validation folds.

Table 7.5: The Reduction in the Number of Additional Features Added for the Essay Parsing Model as a Result of Feature Selection. Addl. = Additional, Fts. = Features

Dataset	**# All Addl. Fts.**	**# Optimal Fts.**	**% Reduction**
Coral Bleaching	5,559.4	3,312.6	40.4%
Skin Cancer	5,322.0	5,189.0	2.5%

The new essay level features added a much smaller number of features to the skin cancer dataset than to the coral bleaching dataset, with an 8.2% increase compared to 14.9%. There were a lot more unique causal relations in the coral bleaching dataset than there were in the skin cancer dataset (see Table 2.9) which is the main reason for the difference in the number of new features. This difference may also explain why the essay level features had a much bigger impact on the coral bleaching model's accuracy because more novel information could be conveyed in this larger set of features.

7.4.1.3 Hyper-Parameter Tuning

The hyper-parameters for this model are the same set of hyper-parameters as for the sentence level parser. However, due to the changes made to the parser and the additional features added, the original set of hyper-parameters may no longer be optimal, and so a grid search was performed to determine the optimal settings for this model. The Logistic Regression model hyper-parameters were optimized as in previous experiments - C values of 0.1, 0.5, 1, 10, and 100 were evaluated using L1, L2 and dual mode regularization methods. The two additional SEARN model hyper-parameters were also tuned once more, the β parameter, which controls the degree of interpolation between consecutive runs, and the *max epochs* parameter, which represents the maximum number of SEARN training iterations. Values of β of 0.1, 0.2, 0.3, 0.4, 0.5, 0.75 and 1.0 were evaluated, along with *max epoch* values of 1, 2, 3, 5, 10, 15 and 20.

The essay parsing model took longer to converge than the sentence parsing model with a max epochs of 10 for the coral bleaching model and 5 for the skin cancer model. The optimal values for the other hyper parameters were similar to the sentence parsing model. For the coral bleaching model, a β value of 0.5 was optimal with L2 regularization using dual mode as before, but with a lower value for the C parameter of 0.1. For the skin cancer dataset, a C value of 0.5 and a β value of 0.3 were optimal. For this model, however, L2 regularization was more effective with dual mode on, which differs from the sentence parsing model where L1 regularization without dual mode was optimal.

These results indicate that the essay parser model took longer to converge on both datasets than the sentence parsing model. This is unsurprising due to the relative complexity of parsing an entire essay relative to a single sentence. There were on average 9.1 sentences per essay in the coral bleaching corpus and 9.8 sentences in the skin cancer corpus. Modifying the parser from operating at the sentence level to the essay level meant that for each parse, many more sequential decisions had to be made, resulting in a much larger search space. Finding an optimal search policy to parse essays rather than sentences was therefore a much more complex task, and it is not surprising that the essay parsing model took longer to converge on an optimal policy. The skin cancer dataset has fewer concept codes and thus fewer unique causal relations (refer to Tables 2.4 and 2.9) and so presents a simpler parsing problem with a smaller search space. This may explain why fewer epochs were needed to learn the optimal policy, even though the essays in that dataset were slightly longer.

7.4.1.4 Results

Tables 7.6 and 7.7 below show the accuracy of the essay parser across the training, validation and test datasets, and includes the sentence parser's performance on the test data for comparison. As a reminder, with my evaluation methodology the models typically perform better on the test dataset because they were trained on a larger amount of training data (see Section 4.2).

Table 7.6: Sentence Parser and Essay Parser Causal Relation Classification Accuracy on the Coral Bleaching Dataset. TD = Training Data, VD = Validation Data

Data Partition & Algorithm	**Micro-F_1**	**Micro-Recall**	**Micro-Precision**
Essay Parser TD (CV)	0.861	0.787	0.950
Essay Parser VD (CV)	0.748	0.710	0.791
Essay Parser Test Data	0.740	0.729	0.750
Sentence Parser Test Data	0.737	0.766	0.710

Table 7.7: Sentence Parser and Essay Parser Causal Relation Classification Accuracy on the Skin Cancer Dataset. TD = Training Data, VD = Validation Data

Data Partition & Algorithm	**Micro-F_1**	**Micro-Recall**	**Micro-Precision**
Essay Parser TD (CV)	0.898	0.851	0.951
Essay Parser VD (CV)	0.811	0.783	0.841
Essay Parser Test Data	0.821	0.813	0.829
Sentence Parser Test Data	0.827	0.800	0.856

Unusually, the essay parser actually performs worse on the coral bleaching test dataset than the validation dataset. This implies the model may have overfit the validation data during hyper-parameter tuning and feature selection. However, the performance on the skin cancer test dataset is higher than on the validation data, similar to that seen with the other models studied in the previous chapters.

Comparing the accuracy of the essay parser to the sentence parser on the test data, only the coral bleaching model shows a slight improvement over the sentence parser, while a small decrease in F_1 score is seen in the skin cancer dataset. There is also a much larger difference between the essay parser's performance on the training data and its performance on the validation and test data when compared with that of the sentence parser (see Tables 7.1 and 7.2). This implies that the essay parser is more susceptible to overfitting than the sentence parser. Therefore, extending the parsing model to work on whole essays instead of sentence by sentence does not produce a consistent improvement in parsing accuracy across both datasets.

There are several reasons why this approach is more susceptible to overfitting. The essay parser has more features, 8.2% more for the skin cancer model and 14.9% for the coral bleaching model. The more features a model has, the more likely it is to overfit because it has more degrees of freedom which can be used to memorize the training data, and learn arbitrary patterns that don't generalize to unseen data points. The SEARN model also has a much larger search space to explore when it parses a whole essay compared to a single sentence because it has to make a larger number of sequential decisions on each parse. This makes it harder to learn an optimal policy because an incorrect parsing decision made near the start of the essay could cause errors in parsing the rest of the essay, instead of just impacting a single sentence's parse.

Switching from a sentence parser to an essay parser had very different effects on recall and precision in the two datasets. In the coral bleaching test dataset, recall dropped but precision increased, while in the skin cancer test dataset the opposite occurred. The micro-F_1 score only improved in the coral bleaching test dataset when using the essay parsing model, and this is partly because there is a smaller difference between recall and precision (0.021 instead of 0.056) than for the sentence parsing model. F_1 score is the harmonic mean of recall and precision and penalizes larger differences in the two values; smaller differences result in an F_1 score that is closer to the midpoint between both numbers. The increase in precision from using the essay parser was also greater than the drop in recall for the coral bleaching test dataset, which also contributed to a higher overall F_1 score. While the gap between recall and precision also dropped for the skin cancer dataset, the drop in precision was about twice as large as the increase in recall (a decrease of 0.027 compared to an increase of 0.013), resulting in a lower overall test data micro-F_1 score when compared to the sentence parser. The skin cancer causal model contains fewer concept codes than the coral bleaching model, 9 codes compared to 13, and thus there are fewer observed causal relations resulting from combinations of these codes. This makes achieving a higher recall easier as there are more training examples for each concept code and each unique causal relation, allowing the model to generalize better to new data points. The essay parser model seems better able to take advantage of this than the sentence parsing model, causing improvements in recall with fewer false negatives but a higher false positive rate for the skin cancer model, resulting in a lower recall and a lower overall micro-F_1 score.

7.4.2 Reranking Model

Reranking is a technique used in machine learning where an initial model is used to generate a set of potential solutions to a problem, and then a more complex model is used to rerank these solutions to produce a more optimal ordering. The top solution or solutions can then be selected from this ranked list. Learning to rank models are widely used in information retrieval, where an initial set of results returned by a search engine are reranked by a machine learning model [17]. Reranking has also been used to improve the accuracy of parsing systems. Traditional greedy approaches to parsing construct a parse tree a word at a time by executing a sequence of locally optimal decisions, performing a greedy search through the search space of possible parse trees. However greedy searches, while computationally efficient, can get stuck in local optima and arrive at sub-optimal solutions [192]. In contrast, using an initial model to generate a top-n list of the best parses which are then reranked by a ranking model allows for a more thorough exploration of the search space of possible parse trees. Because the reranker is able to evaluate complete parse trees, it can attempt to find a globally optimal tree rather than being constrained to choose a sequence of locally optimal decisions.

In his 2002 paper, Collins showed how Rosenblatt's perceptron algorithm could be extended to handle structured learning problems by introducing a $GEN(x)$ function which generates all possible output structures for a given input x [35]. The structured perceptron is then used to rerank the output structures by their expected loss over the whole structure, so the best candidate structure with the lowest expected loss can be selected. In contrast to the essay parsing approach where the parsing decisions are made while the causal model is being constructed, the reranking algorithm allows us to select the globally optimal causal model, and allows for the evaluation of features that can be computed over the entire causal model, instead of a partially constructed one. Collins' reranking approach to structured prediction is therefore a good fit for this problem. To prevent over-fitting, Collins developed weight averaging as a form of regularization when training the perceptron [35], and this was used to regularize the causal model reranking model. For more information on the structured perceptron, please refer to section 3.4.5.

To adapt this approach to determine the best candidate causal model for an essay, the optimal sentence parsing model from Chapter 5 was used, the Shift-Reduce Parser, and extended to perform a beam search when parsing the causal relations from each sentence. A beam search algorithm modifies the best-first search algorithm by keeping a fixed number of the 'best' candidate solutions at each search step (see section 3.4.5 for more details). Rather than outputting a single set of causal-relations for each sentence, the beam search parser generates k sets of causal relations, or parses, where k is the beam size. At the start of the sentence, for each concept code it decides which transition from the transition system to apply (see Figure G.2 in the Appendix for the transition system). Instead of only executing the optimal transition according to the parsing model, it executes the optimal k transitions. It then continues in this way, dropping all but the top k decisions for each parse step, keeping k concurrent parses in memory as it processes the sentence's concept codes.

The beam search parser was used to construct the $GEN(x)$ function for the reranking model, as outlined in Algorithm 7.1 below.

Algorithm 7.1 The $GEN(x)$ Function for the Reranking Model

```
 1: function GenerateCausalModels($e$)                ▷ $e$ is an essay
 2:     $C \leftarrow \emptyset$                      ▷ $C$ is the set of causal models
 3:     for each $s \in e$ do                         ▷ $s$ is a sentence
 4:         $p \leftarrow$ BeamSearchParse($s$, $k$)  ▷ $k$ is the beam size
 5:         for $i \leftarrow 1$ to $k$ do
 6:             $C \leftarrow C \cup p_i$
 7:         end for
 8:     end for
 9:     return GetPermutations($C$)
10: end function
```

The *GenerateCausalModels* function works as follows. For each sentence s in essay e, the beam parser produces an array of sentence parses p. p contains the k best sentence parses, where k is the beam size. Each separate sentence parse p_i produces a set of causal relations. All of the causal relations parsed across all of the sentences are added to C, the set of unique parsed causal relations for the essay. Once all of the causal relations have been parsed for the essay, the *GetPermutations* function is called which returns the set of all possible permutations of the causal relations in C. Each of these permutations is a set of causal relations, i.e. a candidate causal model for the entire essay. If there are n relations in C, then there will be 2^n candidate causal models generated from C by this function. For some essays with a relatively large number of causal relations, this algorithm therefore generates a very large number of causal models, too many to rerank in an acceptable time frame. To address this problem, an upper limit was chosen for the number of parsed causal relations; if more causal relations were parsed, those assigned the lowest confidence by the parsing model were dropped, and the remainder used to generate the causal models.

To determine a reasonable upper bound for the number of causal relations used in the generative model, statistics were computed to describe the distribution of the number of causal relations per essay in the training data (labelled by humans), and the number of causal relations parsed per essay by the sentence parser (using cross-validation on the same dataset). This is shown in Figures 7.2 and 7.3 below.

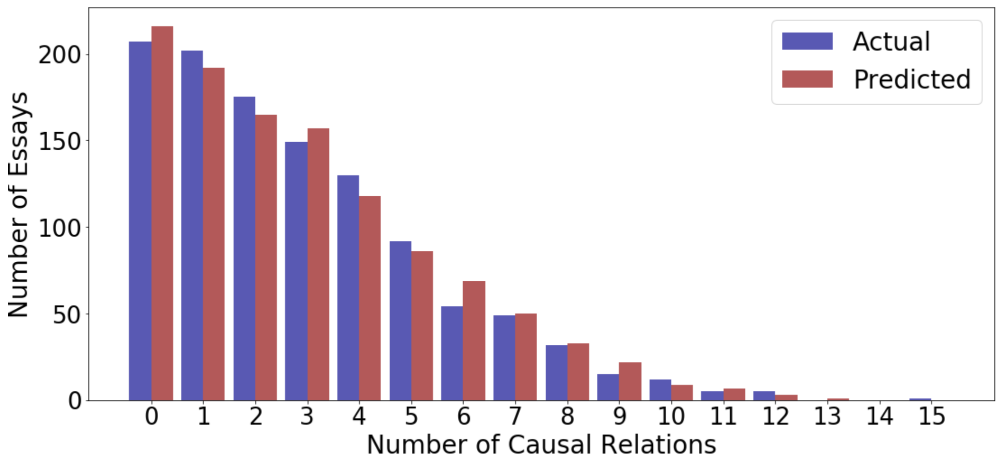


Figure 7.2: The Distribution of the Number of Causal Relations Per Essay in the Coral Bleaching Dataset (Actual vs. Predicted by the Sentence Parser)

In the human assigned labels, 95% of coral bleaching essays have 8 or fewer causal relations, while 95% of skin cancer essays have 10 or fewer causal relations. The causal relations parsed by the sentence parser closely follow the same distribution, with 8 causal relations also at the 95th percentile for the coral bleaching data, and 9 instead of 10 causal relations at the 95th percentile for the skin cancer data. Values of 8 and 9 respectively were therefore used as the maximum number of causal relations used to generate the set of potential candidate causal models.

Once sets of candidate causal models were generated for each essay, a micro-F_1 score was computed for each and used to rank the solutions by their quality. A reranking model was then trained to replicate this ranking

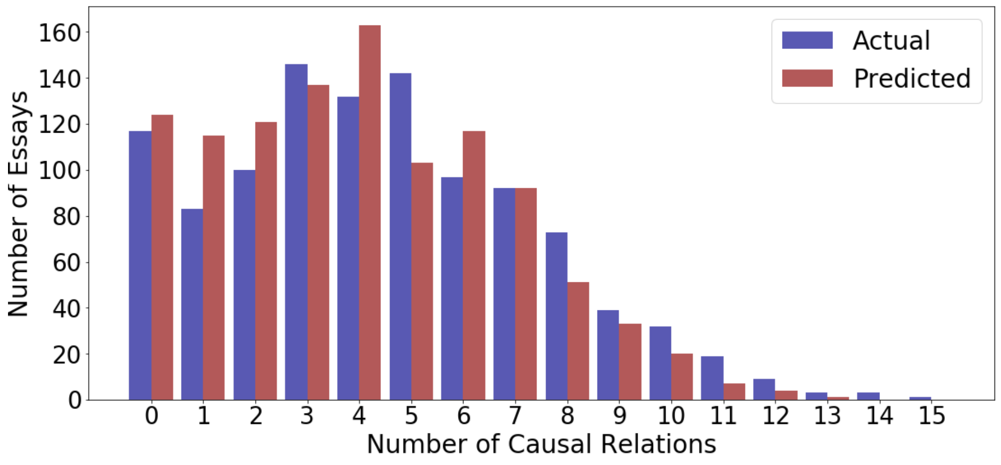


Figure 7.3: The Distribution of the Number of Causal Relations Per Essay in the Skin Cancer Dataset (Actual vs. Predicted by the Sentence Parser)

(optimizing the essay level micro-F_1 score) using a set of features computed from each causal model. The set of features used and how they were derived will be explained in the following section.

7.4.2.1 Features Evaluated

The features chosen for the reranking task were designed to capture information about the structure of the parsed model, the parser's confidence in the set of causal relations it contains, and possible errors in the model construction. To differentiate between the original *ideal* causal model that was defined by the team of cognitive psychologists who defined the original task and the causal models generated by the algorithm, the human constructed causal model will be referred to as the 'reference' causal model in the remainder of this chapter any time it is referred to specifically.

The different types of causal model features are listed below, with a brief description of each:

- **Causal Relations Frequency** - How many occurrences of each type of causal relation are in the model?
- **Probability Statistics** - Statistics computed over the set of probabilities the model assigned to each causal relation
- **Causal Relation Confidence** - The proportion of causal relations with a probability above a set of thresholds - 0.1, 0.2, 0.3, 0.5, 0.7, 0.8, 0.9 and 0.95

- **Total Number of Causal Relations** - How many causal relations were there in the model?
- **Causal Relation Types** - Percentage of different types of relation, including duplicate relations, relations crossing between 2 paths in the reference causal model, and relations where the direction of causality matches that of the reference causal model
- **Inverted Causal Relations** - If a relation A→B exists, does B→A exist? Frequency and proportion of inverted relations.
- **Causal Relation Span Statistics** - Mean, minimum and maximum distance (number of concept codes) between the pairs of concepts in each causal relation when computed from the reference causal model
- **Causal Chains** - A causal chain is a sequence of two or more causal relations connected in the reference causal model
- **Causal Chain Statistics** - Statistics include the number, frequency and maximum length of all chains
- **Concept Code Frequency** - How many occurrences of each concept code are there in the candidate causal model?

When training a structured perceptron model using numerical features for a reranking task, it is important that the features are normalized to all have the same range. This prevents features with large ranges from dominating the model. *Min-max rescaling* [114] was consequently used for feature normalization. For some feature x, the feature is re-scaled to x_{norm} by subtracting the minimum value and dividing by the range [114], as described in Equation 7.1 below:

$$x_{norm} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{7.1}$$

7.4.2.2 Feature Selection

As in previous experiments, forward selection was used to select the combination of feature sets with the highest overall micro-F_1 score. The forward selection procedure was executed until either the micro-F_1 score declined over the previous run, or a maximum combination of 6 feature sets had been evaluated. A number of important hyper-parameters control how the model works and help prevent over-fitting. These include the number of training iterations used to train the reranker, the *beam size* of the beam search algorithm and the learning rate parameter. Sensible default values for each of these parameters were chosen and fixed throughout the feature selection process, and then fine-tuned following feature selection.

Early-stopping was used to determine the number of iterations, as described in Chapter 4 in Section 4.7.4. The reranking model was trained initially using all features on 80% of the training data. Training was then stopped when the classification accuracy on the remaining 20% of the training data started to decline. From this procedure, the optimal number of training iterations was determined to be 2 for the coral bleaching dataset and 1 for the skin cancer data. These values were then used throughout the forward selection process. A beam size of 2 and a learning rate of 0.3 were used to train the model during feature selection. Section 7.4.2.3 will discuss these hyper parameters in more detail, and describe how they were tuned following feature selection.

To understand the importance of each feature in the absence of the other features, the micro-F_1 score of each feature set used individually is show in Table 7.8 below. These numbers were calculated during the first forward selection loop.

Table 7.8: Each Baseline Feature Set Ranked by Micro-F_1 Score When Used in Isolation. Stats. = Statistics, Conf. = Confidence, Rel. = Relation.

	Coral Bleaching		**Skin Cancer**	
	Feature Set	F_1	**Feature Set**	F_1
1	Probability Stats.	0.738	Causal Rel. Conf.	0.810
2	Causal Rel. Conf.	0.734	Probability Stats.	0.803
3	Concept Code Frequency	0.732	Causal Rels. Frequency	0.798
4	Number Causal Rels.	0.731	Concept Code Frequency	0.790
5	Causal Rels. Frequency	0.729	Number Causal Rels.	0.672
6	Causal Rel Span Stats	0.636	Causal Rel. Types	0.671
7	Causal Rel. Types	0.634	Causal Rel Span Stats	0.629
8	Causal Chain Stats.	0.624	Causal Chain Stats.	0.623
9	Causal Chains	0.339	Causal Chains	0.613
10	Inverted Causal Rels.	0.000	Inverted Causal Rels.	0.446

In both datasets, the *Probability Statistics* and the *Causal Relation Confidence* were the two most useful sets of features when used alone, the

Probability Statistics performing better in the coral bleaching dataset, whilst the *Causal Relation Confidence* had the higher micro-F_1 score in the skin cancer dataset. When a causal relation is parsed, it is possible to determine the parser's confidence in the prediction by examining the probability it assigned to that decision. Both of these sets of features measure the overall confidence of the parser in its predictions, with *Probability Statistics* computing the average, minimum and maximum probabilities across all parsed relations, while *Causal Relation Confidence* computes the same measures for each unique relation. Using these features to determine the overall quality of each causal model appears to be critical for the ranking task. Usually when combining models in this way, including the base model's confidence as well as its prediction improves the classification accuracy of the higher-level model consuming its predictions [221, 198].

However, the true value of a set of features is how much they contribute to improving the model's accuracy when combined with all other useful features. Table 7.9 below shows the order in which features were added by feature selection for each dataset:

Table 7.9: Micro F_1 Score as Additional Feature Sets Are Added by Forward Selection. Stats. = Statistics, Conf. = Confidence, Rel. = Relation.

	Coral Bleaching		**Skin Cancer**	
# Fts.	**Added Feature Set**	F_1	**Added Feature Set**	F_1
1	Probability Stats.	0.7376	Causal Rel. Conf.	0.8098
2	Causal Rel. Conf.	**0.7405**	Inverted Causal Rels.	0.8104
3	Causal Rel. Types	0.7382	Number Causal Rels.	0.8107
4	Inverted Causal Rels.	0.7384	Causal Chain Stats.	**0.8118**
5	Causal Rel Span Stats	0.7386	Causal Rel Span Stats	0.8113

For the coral bleaching dataset, the model only selected the two feature sets that describe the model's confidence in its predictions, indicating that information about the structure of the causal model, the causal chains within it and the size of the model were not useful for that dataset. These types of features did improve the accuracy of skin cancer reranking model, although the increase in micro-F_1 score over the *Causal Relation Confidence* feature set was small.

Concept Code Frequency was ranked in the top three and four features for each dataset when evaluated individually, however it did not appear in the final set of selected features, suggesting that it did not add any additional useful information when used in conjunction with other features. The same is true for the *Probability Statistics* feature set in the skin cancer dataset. The number of *Inverted Causal Relations* was not useful in the coral bleaching dataset, but was the second set of features to be added to the skin cancer dataset. This implies that the skin cancer parser was less certain about the direction of causality, generating causal relations in both directions due to the beam search. Adding this feature allowed it to penalize causal models where this occurred. Because the algorithm generates all possible causal models from the set of most probable parsed causal relations, some generated models may contain too many or too few causal relations when compared to the ground truth. The *Total Number of Causal Relations* feature encodes this information, and proved useful in the skin cancer dataset only.

The two causal chain feature sets capture information about the causal chains within the causal models. In this context, a causal chain is a sequence of two or more causal relations that are connected in the reference causal model. For example, a causal chain from the coral bleaching domain would be 'INCREASE IN WATER TEMPERATURES $\rightarrow$ DECREASE IN CO_2 $\rightarrow$ DECREASE IN PHOTOSYNTHESIS' (see Figure 2.1). The *Causal Chains* features contain a separate feature for each causal chain in the candidate causal model. However, it appears these individual features were too rare to be useful to the reranking model. *Causal Chain Statistics* encode the number of causal chains and their average and maximum length, and were the last feature set to be selected by the skin cancer reranking model. Providing high level statistics about the causal chains allowed the model to generalize better over the causal chain data. Their inclusion in the final dataset alongside the *Inverted Causal Relations* and the *Total Number of Causal Relations* indicates that information about the structure and size of the candidate causal models was useful for the reranking task.

The goal of feature selection is to produce a simpler, more robust model by eliminating redundant or noisy features. Table 7.10 below shows the reduction in the number of features in the parsing model as a result of feature selection.

Table 7.10: The Reduction in Number of Features for the Reranking Model as a Result of Feature Selection

Dataset	# All Features	# Optimal Features	% Reduction
Coral Bleaching	1,698.8	33	98.1%
Skin Cancer	798.8	88.6	88.9%

There was a smaller reduction in the total number of features in the skin cancer dataset as more sets of features were selected by forward selection. Overall, the features providing summary statistics about the quality of the causal model proved more useful to the ranking task than the high cardinality features that listed the specific causal chains or causal relations present in the causal models. Once the optimal set of features was determined, the model hyper parameters were tuned to maximize the F_1 score on the validation data set.

7.4.2.3 Hyper-Parameter Tuning

A number of important hyper-parameters control how the reranking approach works - the *beam size*, the number of training iterations and the learning rate. Beam size controls the size of the beam used in the beam search, in other words how many different parses are considered at each parse step, and thus determines the number of parses that are generated for each sentence parsed. Beam size is a critical parameter in the algorithm's design, as it influences how many different causal relations are parsed per essay and therefore how many candidate causal models are generated. A beam size of 1 generates the same set of causal relations as the original sentence parser studied in Chapter 5. However, setting a beam size of 2 or higher allows the parser to explore a larger search space, parsing additional causal relations which were not parsed by the original parser. A larger beam size therefore allows the algorithm to attempt to improve recall, although in doing so it may reduce precision by increasing the false positive rate.

As in previous chapters, a grid search was used to tune the beam size and the learning rate hyper-parameters. Learning rate is the coefficient that controls the size of each weight update applied. The other main hyper-parameter, the number of training iterations, was fixed throughout feature selection and hyper parameter tuning after an initial value was set using early-stopping (see Section 7.4.2.2). Beam sizes of 1, 3, 5, 7 and 10 were evaluated, along with learning rates of 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5 and 1. For the coral bleaching dataset, the optimal beam size was 1, with a learning rate of 0.1. In contrast, for the skin cancer dataset the optimal beam size was 5, with a learning rate of 0.05. This implies very different optimal search strategies for each dataset when generating models to rerank.

7.4.2.4 Results

The performance of the reranker model with the optimal features and hyper-parameters across the 3 different data partitions can be seen in Tables 7.11 and 7.12 below. The performance of the sentence parsing model on the test dataset is also shown for comparison.

Table 7.11: Sentence Parser and Reranker Causal Relation Classification Accuracy on the Coral Bleaching Dataset

Data Partition & Algorithm	**Micro-F_1**	**Micro-Recall**	**Micro-Precision**
Reranker Training Data (CV)	0.743	0.704	0.786
Reranker Validation Data (CV)	0.741	0.704	0.783
Reranker Test Data	0.750	0.748	0.752
Sentence Parser Test Data	0.737	0.766	0.710

Table 7.12: Sentence Parser and Reranker Causal Relation Classification Accuracy on the Skin Cancer Dataset

Data Partition & Algorithm	**Micro-F_1**	**Micro-Recall**	**Micro-Precision**
Reranker Training Data (CV)	0.812	0.779	0.849
Reranker Validation Data (CV)	0.814	0.780	0.850
Reranker Test Data	0.829	0.814	0.845
Sentence Parser Test Data	0.827	0.800	0.856

In both datasets, the highest micro-F_1 score was achieved on the test dataset due to a larger amount of training data. Little difference in F_1 score is seen between the training and validation data for both the coral bleaching and skin cancer datasets, indicating that this approach is not prone to over-fitting. This contrasts with the essay parser model, which was susceptible to over-fitting. Similar to the essay parsing model, precision is higher than recall for both datasets across the training, validation and test data partitions.

Comparing the reranking algorithm's performance to the sentence parsing model, the reranking model shows a consistent improvement in the test data micro-F_1 score across both datasets, indicating that this approach does improve upon the sentence parser's predictions. However, the gain in micro-F_1 score is very small for the skin cancer dataset. In the test data, the reranking approach improves the recall in the skin cancer dataset while lowering the precision. In contrast, the exact opposite is true for the coral bleaching dataset, the precision is increased at the cost of recall. The beam size parameter allows the algorithm to trade off precision and recall to optimize the micro-F_1 score. F_1 score is the *harmonic mean* of both recall and precision; when there is a difference in recall and precision, the F_1 score is below the mid-point between the two values and is closer to the lower value. Thus, it penalizes approaches that heavily favor either recall or precision, and do not balance both metrics. During the hyper parameter tuning, the optimal values for the beam size were determined to be 1 for the coral bleaching dataset, and 5 for the skin cancer dataset. A beam size of 1 means that the set of causal relations that are parsed are the same as those parsed by the regular sentence parser. With a beam size of 1, the reranker is essentially learning which of these relations to prune and which to keep, which has the effect of improving precision by reducing the false positive rate (see Equation 4.1), although recall may suffer as a result. Conversely, a larger beam size such as 5 means that additional causal relations are being added to the causal model, enabling improvements in recall also. Thus the beam search acts to improve recall, while the structured perceptron reranking model acts to improve precision. This behavior explains why the reranking approach improves precision in the coral bleaching dataset while improving recall in the skin cancer dataset due to the different beam sizes.

To illustrate this point, the impact of beam size on precision and recall during hyper-parameter tuning can be seen in Tables7.13 and 7.14 below.

Table 7.13: The Impact of Beam Size on the Accuracy Metrics and the Size of the Generated Causal Models in the Coral Bleaching Validation Dataset. Max. = Maximum, Rels. = Causal Relations. All Metrics are Micro Metrics.

Beam Size	**Max. Rels.**	**Mean Rels.**	F_1	**Recall**	**Precision**
1	13	2.96	**0.7414**	**0.7040**	0.7829
2	13	2.97	0.7410	0.7000	**0.7870**
3	15	3.64	0.7396	0.6982	0.7863
5	22	5.35	0.7392	0.7015	0.7813
7	22	5.36	0.7391	0.6985	0.7847
10	22	5.36	0.7383	0.6985	0.7828

Table 7.14: The Impact of Beam Size on the Accuracy Metrics and the Size of the Generated Causal Models in the Skin Cancer Validation Dataset. Max. = Maximum, Rels. = Causal Relations. All Metrics are Micro Metrics.

Beam Size	**Max. Rels.**	**Mean Rels.**	F_1	**Recall**	**Precision**
1	13	3.97	0.8095	0.7578	**0.8689**
2	13	3.98	0.8092	0.7583	0.8674
3	15	4.81	0.8133	0.7783	0.8516
5	25	7.75	**0.8138**	0.7804	0.8502
7	25	7.75	0.8136	**0.7809**	0.8491
10	25	7.75	0.8131	0.7794	0.8498

To provide some context, the maximum number of unique causal relations observed in the manually labelled data (ground truth) per essay was 15 causal relations in each dataset. Thus the larger beam sizes must be generating a number of false positives for some essays. When modifying an algorithm to improve recall, often precision suffers because we are trading one type of error (false negatives) for another (false positives). In other words, broadening the coverage of the algorithm also increases the number of bad matches we get for a particular class. The opposite is also true for the same reason, improving precision often hurts recall. From Table 7.14 we see that for the skin cancer dataset, as we increase beam size, recall improves while precision drops. However, for the coral bleaching dataset, increasing beam size hurts both recall and precision. This implies that for the coral bleaching data, the beam search does not generate additional valid causal relations, but the reranking model is still effective at pruning invalid causal relations.

Overall, the sentence parser's predictions are much more accurate on the skin cancer data than the coral bleaching data, and this performance naturally carries over to the reranker model which consumes those predictions. Consequently, the reranker has a much larger impact on improving the accuracy of the coral bleaching model - an improvement in micro-F_1 score of 1.76% compared to 0.24% on the test data, because there are more errors to correct. The reranking algorithm also appears to be better at correcting for low precision than low recall. The gain in precision on the coral bleaching test dataset as a result of the reranking algorithm (5.92%) was much higher than the subsequent gain in recall on the skin cancer test dataset (0.49%).

7.5 Summary and Discussion

To evaluate the efficacy of these two approaches at improving the accuracy of causal relation parsing, at the essay level, I compare the algorithm's performance using both the micro-F_1 score, and the macro-F_1 score, which is more sensitive to model's performance at detecting the less frequent labels. Additionally, I perform a number of statistical tests to determine of the differences in model performance are statistically significant or likely due to chance, and finally I discuss the extensibility of these techniques to other texts and other domains.

7.5.1 Micro-F_1 Performance

Summary Tables 7.15 and 7.16 below compare the test data classification accuracy for the essay parser and reranking algorithm with the sentence parser discussed in Chapter 5.

Table 7.15: Test Data Accuracy by Algorithm on the Coral Bleaching Dataset. All Metrics are Micro Metrics. Sent. = Sentence, Feats. = Features

Algorithm	F_1	**Recall**	**Precision**	**Features**
Sent. Parser	0.737	**0.766**	0.710	Templated Parsing Feats.
Essay Parser	0.740	0.729	0.750	Causal and Ratio
Reranker	**0.750**	0.748	**0.752**	Causal Probabilities

Table 7.16: Test Data Accuracy by Algorithm on the Skin Cancer Dataset. All Metrics are Micro Metrics. Sent. = Sentence, Feats. = Features, Probs. = Probabilities

Algorithm	F_1	**Recall**	**Precision**	**Features**
Sent. Parser	0.827	0.800	**0.856**	Templated Parsing Feats.
Essay Parser	0.821	0.813	0.829	# Concept Codes, Causal
Reranker	**0.829**	**0.814**	0.845	Causal Probs, Structural

When compared to the micro-average classification metrics computed at the sentence level (see Tables 5.18 and 5.19 in Chapter 5), we see a clear improvement in the micro-F_1 score, and the micro-precision and micro-recall scores are the same or better, even when comparing the sentence-level parser's performance on both sets of metrics. This indicates that when ignoring the specific location of the causal relations (i.e. the sentence in which they occur), the model is more accurate at determining the causal relations present.

When compared to the sentence parsing model discussed in Chapter 5, only the reranking model shows a consistent improvement in micro-F_1 score on the test data for both datasets. The essay parser produces a smaller gain in F_1 score on the coral bleaching dataset, but has a lower F_1 score on the skin cancer dataset than the sentence parser. The reranking algorithm was designed to improve upon the predictions of the sentence parser model while utilizing additional features computed over the final parsed causal models. Empirically, it improves the sentence parser's accuracy in a different way for each dataset. For the coral bleaching dataset, it improves the micro-precision by pruning relations that were assigned a low probability by the sentence parser. However, no new causal relations were parsed because the beam search was not used (beam size was 1) so the recall dropped slightly relative to the sentence parser. In contrast, for the skin cancer dataset recall was improved by using a beam search to parse additional causal relations but similarly incurring a slight drop in precision as a result.

The goal of Research Question 4 was to determine whether we could make use of the structure of the whole essay and the predicted causal model to improve the causal relation parsing accuracy at the essay level. Extending the sentence parser to parse whole essays using some structural features produced a small improvement in F_1 score on the test data (0.41%), but only in the coral bleaching dataset. The reranking algorithm produced a larger accuracy gain on the same dataset (1.76%) but only used features derived from probabilities assigned each causal relation by the sentence parser, and did not use any structural features derived from the predicted causal models. The reranker did produce an increase in micro-F_1 score by using some structural features on the skin cancer dataset. However, the additional accuracy produced by adding these structural features (see Table 7.9) was very small, as was the overall increase in F_1 score over the sentence parser model on the skin cancer data. Therefore any advantage conferred from using structural information seems to be very small, if any. Nevertheless, the flexibility of the reranking approach to trade off recall and precision and correct for different types of errors made by the sentence parser was effective in improving the accuracy of the predicted causal models. Based on the feature selection results, using information about the sentence parser's own confidence in its own predictions was also critical to the success of the reranking approach on both datasets, and was much more important than the structural information.

7.5.2 Macro-F_1 Performance

The micro-average classification metrics take into account the relative frequency of each class, or in this case each causal relation, in the entire dataset when evaluating the overall F_1, recall and precision metrics. As in previous chapters, it is also useful to look at the macro-average metrics to understand how the different models perform on the rarer causal relations in the dataset. Tables 7.17 and 7.18 below show the macro-average classification accuracy metrics for the each algorithm at detecting causal relations at the essay level. For a detailed explanation of the differences between micro and macro averages, please refer to Section 4.1.3.

Table 7.17: Macro Test Data Metrics by Algorithm on the Coral Bleaching Dataset

Collection	Macro-F_1	Macro-Precision	Macro-Recall
Sent. Parser	**0.295**	**0.284**	**0.308**
Essay Parser	0.249	0.254	0.245
Reranker	0.253	0.247	0.260

Table 7.18: Macro Test Data Metrics by Algorithm on the Coral Bleaching Dataset

Collection	**Macro-F_1**	**Macro-Precision**	**Macro-Recall**
Sent. Parser	0.319	0.335	0.304
Essay Parser	**0.336**	0.331	**0.340**
Reranker	0.327	**0.340**	0.315

From these tables we can see that the macro-average metrics are considerably lower than the micro-average metrics. This indicates that the three different algorithms studied in this chapter are much more accurate at predicting the more common classes (causal relations) than the rarer classes. In general, machine learning models produce more accurate predictions when there are more examples of the target class to learn from. Many of the individual causal relations occur rarely in the data (less than 5 times). The errors on the rare classes are much higher than on the more common classes, and this is reflected in the much lower macro-average scores. This disparity was also observed between the micro and macro average metrics scores for the causal relation classification task studied in Chapter 5 (see Tables 5.23 and 5.24). This disparity has carried over to the models studied in this chapter which either consume predictions from these models or further extend them.

When comparing the essay-level with the sentence-level macro averages, the essay-level metrics are lower than the sentence-level averages (see Chapter 5). This contrasts with the micro-averages, where the opposite was observed. When computing these metrics at the essay level, duplicate causal relations within each essay are ignored. It is inherently more likely to observe a more common causal relation multiple times within an essay, so by ignoring duplicate causal relations the essay level macro averages are even more sensitive to the rarer causal relations, explaining this difference in relative performance across the different averaging methods.

Examining the macro-average metric performance by algorithm, a very different pattern emerges when compared to the micro-averages. The algorithm that has the lowest micro-F_1 score in each dataset has the highest macro-F_1 score in that dataset. Specifically, the sentence parser has the highest macro-F_1 score on the coral bleaching dataset, and the essay parser has the highest macro-F_1 score on the skin cancer dataset. The F_1 metric is non differentiable, and cannot be optimized directly via gradient-based methods, and thus the micro and macro-F_1 metrics are also non differentiable (see Section

4.1.4 for a more detailed explanation). The two shift-reduce parser models and the reranking algorithm each try to optimize the micro-F_1 score, the parsers use a custom cost function within the SEARN algorithm (see Appendix G, Section G.5) while the reranking approach uses the beam size hyper-parameter to vary the precision-recall trade off. The approaches most effective at optimizing the micro-F_1 score perform worse at optimizing the macro-F_1 metric, thus by focusing more on reducing the errors for the more frequent classes, the rarer classes perform worse. However, in spite of this the reranking algorithm still has the second highest macro-F_1 score across both tasks.

7.5.3 Statistical Analysis

As in previous chapters, I ran a Cochran's Q test to compare the performance of the 3 different approaches at detecting the causal relations at the essay level. The approach to statistical testing used in this thesis is described in detail in Section 4.3 in Chapter 4. Cochran's Q test produces a χ^2 value of 7.0, with a p-value 0.03, on the coral bleaching dataset, rejecting the null hypothesis. Thus the 3 techniques studied, the sentence parser, the essay parser, and the reranking approach, have statistically significant differences in accuracy on this dataset. Running McNemar's test to determine if there is a statistically significant difference between the reranking approach and the other two algorithm produces the results shown in Table 7.19 below:

Table 7.19: Comparing the Performance of the Different Algorithms on the Coral Bleaching Dataset with the Reranking Approach Using McNemar's Test

Algorithm	F_1	**p-value vs Reranker**
Sentence Parser	0.737	0.002
Essay Parser	0.740	0.437
Reranker	0.750	-

Once again, the α value has to be corrected to 0.025 to account for 2 comparisons, using the Bonferroni correction. Based on McNemar's test, there is a significant difference between the performance of the reranker and the sentence parser, but when compared to the essay parser. Thus there is not a statistically significant difference between the accuracy of the reranker and both of the other techniques, but the improvement in performance over the sentence parser alone does not appear to be due to chance, on the coral bleaching dataset.

Running Cochran's Q test to compare the algorithm performance on the skin cancer dataset, a χ^2 value of 4.13 is attained, which has an associated p-value of 0.127. This is above the α value and is not significant. The null hypothesis holds, and there does not seem to be a significant difference between the performance of the three techniques on the skin cancer dataset. The micro-F_1 scores differ only slightly on the skin cancer dataset, so this is unsurprising. Using the reranking approach or extending the sentence parser to parse whole essays does not appear to improve upon the sentence parser, for the reasons discussed earlier in this section.

7.5.4 Extensibility to Other Texts and Domains

The techniques in this chapter focus on extending the causal relation parsing model from the sentence to the essay level. However, based on the improvements in micro-F_1 score and the results of the statistical testing, there is only a significant improvement in the coral bleaching dataset, and not the skin cancer dataset. Given that consistent results are not seen across both datasets, it is hard to predict whether an improvement would be seen in other domains, where a sentence parsing model for detecting binary relations was extended in the same way using a reranking approach. In order to better answer this question, this experiment would need to be repeated on a number of additional datasets (preferably in domains other than scientific essays) to see if the same results are found, or whether these approaches are more or less successful when applied to other texts. Based purely on the results from this thesis, I would argue that it is possible but unlikely that these results would extend to other domains, given the small effect size observed in the coral bleaching dataset, despite the fact that the performance improvement over the sentence parser was statistically significant.

7.6 Conclusions

One key advantage of reranking approaches in structured learning is that they allow for the use of an objective function that seeks the globally optimal solution, rather than optimizing a series of local decisions [43]. In this chapter I presented a reranking model that was able to predict more accurate causal models by optimizing the overall micro-F_1 score per essay, rather than at the sentence level. Furthermore, the combination of a beam search followed by a reranking model allowed for the F_1 score to be optimized indirectly by better balancing recall and precision by tuning the beam size hyper-parameter. As was observed in Chapter 5, the closer we can get to optimizing the desired evaluation metric in our model design, the better the model will perform on that metric, and the results from this chapter further support this conclusion. Additionally, it was demonstrated that when using a beam parser as the $GEN(x)$ function in Collin's reranking approach [35], the beam-size can be varied to optimize the micro-F_1 score. This approach could be extended to optimize any aggregate F_1 metric in other multi-label classification problems, provided a suitable $GEN(x)$ function could be used to generate different combinations of labels.

The goal of Research Question 4 was to determine whether using information from the whole document would improve the accuracy of the causal relation extraction, compared to an approach that processes each sentence independently. My results from this chapter were inconclusive. On the one hand, using the reranking approach did produce an improvement in the classification accuracy on both datasets. However, the reranking models relied principally on features that encoded the confidence of the sentence parser in its own predictions; the structural features had little impact on the model's accuracy, and were removed by feature selection on the coral bleaching dataset, where the biggest accuracy gains were observed. While it is possible that different models or different sets of features could produce a larger improvement in accuracy when using structural features, these results imply that for scientific explanatory essays, most of the information needed to determine causality is encoded locally at the sentence level. One possible reason for this is the presence of coreferences within the essays. An analysis of the causal relations within the essays show a significant number of causal relations involving coreferences (see Section 2.5). Coreference resolution is a very difficult NLP problem, the Stanford and Berkeley systems evaluated in Chapter 6 achieving an F_1 score of only 0.62 and 0.65 when they were evaluated on the dataset they were trained on [31, 59]. They then performed much worse when applied to the essay text. If a better coreference resolution model were developed that was more accurate at resolving coreferences on this dataset, a whole essay parsing approach could be more successful.

These results also have implications for building pipelines of machine learning models, where the predictions from some models feed into other downstream models. When models are stacked in this way, the success of the downstream models is dependent on the overall accuracy of the models earlier in the pipeline. Errors tend to propagate through the system, and it is hard for the downstream models to correct for errors earlier in the pipeline. This is demonstrated by the performance of the reranking algorithm studied in this chapter. While precision was improved by removing some false positive predictions from the base model (the sentence parser), the main gains in micro-F_1 score came from better balancing recall and precision. In each dataset, it improved either precision or recall, but not both; an improvement to one resulted in a decrease in the other. Because reranking algorithms rely on the predictions from a base model, this is a well known limitation of this approach [43].

Chapter 8

Summary and Conclusions

In conjunction with the READI grant [70] (funded by the National Center for Education Research), a study was undertaken by a team of cognitive psychologists to evaluate how effectively students are able to synthesize and combine information from different source texts. Two essay topics were chosen describing different scientific phenomena, the bleaching of coral reefs and the causes of skin cancer, and source texts were selected that described the different causes underlying these phenomena. The sources were selected so that no single text contained all of the information necessary to answer the essay questions; the students were forced to integrate information from multiple sources to answer each essay question. 1,100 explanatory essays explaining the two science topics were collected from high school students in the Chicagoland area and annotated according to 2 pre-defined causal models. Each causal model described the causal reasoning structure that the students were expected to produce in an ideal essay answer. These annotated essays were then used to train the machine learning models evaluated in this thesis.

The goal of this research was to produce a machine learning system which can parse the causal structure of an essay in either scientific domain, mapping the causal inferences in the essay to the reference causal model. The problem was decomposed into 4 different sub-problems, producing 4 different research questions. Solutions to these 4 sub-problems comprise a system for inferring causal models from essay text in each of these two domains. It is hoped that this system could be used as a general framework for extracting causal models from text in many different domains, not just limited to scientific essays. Existing essay grading software relies on shallow surface-level lexical and grammatical features [185, 48]. While these features are effective at predicting an essay's grade by evaluating a student's writing proficiency, they are not able to effectively judge the quality or validity of the student's arguments [164, 56, 247]. Instead, new techniques are needed which can evaluate the quality of the arguments and the causal reasoning of the student, as expressed through their writing [94]. The approaches outlined in this research could be used to build such a system.

8.1 Summary

The micro-F_1 metric was chosen as the principle evaluation metric for this research because it is designed for evaluating classification accuracy in multi-class multi-label classification problems, where the class labels are imbalanced (see Section 4.1 for a more detailed definition). The goal of Research Question 1 was to determine the most effective machine learning approach for labelling words with concept codes from the causal model. Five different word tagging models were trained to predict the concept or concepts associated with each word in the essays - a window-based tagging model, a Conditional Random Field (CRF), a Hidden Markov Model (HMM), a Structured Perceptron and a Bidirectional Recurrent Neural Network (RNN). The bidirectional RNN was most effective at this task when evaluated across both datasets. The key to this model's success lay in its ability to use and fine-tune pre-trained word embeddings, produced by a language model that was trained on a semantically related dataset (*GloVe* vectors [88, 168] trained on Wikipedia, which contains scientific content). Furthermore, the bidirectional and recurrent nature of the RNN model gave it a unique advantage, as it was able to learn from all of the labels and words to the right *and* left of the target word. Also, the RNN was not restricted to a fixed-size word window or only learning from the left context of the target word. The window-based tagging model also achieved a high classification accuracy on this task, comparable to the RNN on one of the datasets. Unlike the other 4 approaches, this model only used features extracted from the surrounding words, and did not learn from the labels assigned to preceding or subsequent words. This implies that for this task, the surrounding words and not their concept codes, were the most important features for this task.

Research Question 2 addressed the problem of learning causal relations between concepts. Using the predictions from the most accurate word tagging model (the bidirectional RNN), three different models were trained to determine the causal relations linking the predicted concepts within each sentence. The first model evaluated was a bidirectional RNN, which extended the word tagging model from Research Question 1 to tag words with causal relations. The second approach applied the idea of 'stacked generalization'

[246]. Sentence-level features were extracted from the word tagging model's predictions and used to train a meta-classifier, a logistic regression model, to predict the causal relations present in each sentence. Finally, a shift-reduce parser was developed which used the SEARN imitation learning algorithm [45] to train a transition-based dependency parsing model to parse binary relations between predicted concept codes.

The shift-reduce parser achieved the highest performance across both datasets. Critical to the success of this approach was its use of a custom cost function which allowed the model to directly optimize the micro-F_1 score (see Section G.5 in Appendix G). Moreover, instead of learning to predict each causal relation as a separate label, this model solved a more general problem - it was able to learn to determine if any relation existed between any pair of codes. Because there are more occurrences of causal relations between pairs of concept codes than there are occurrences of individual relations, this lowered the sample complexity of the problem [232, 231] and allowed the model to generalize more effectively to classify new data points. Therefore, this model was able to achieve better generalization because it was designed to take advantage of the structure of the problem.

Research Question 3 addressed the problem of resolving anaphoric references to words associated with a concept code. Two existing state-of-the-art systems were evaluated on this task, the Stanford and Berkeley coreference resolution systems [90, 89]. However, both of these systems proved ineffective at resolving the labelled coreferences within the two essay datasets. Consequently, an alternative solution was developed to solve this problem. A bidirectional RNN was trained to predict which words were labelled as anaphors in the training data. Using predictions from this model, the antecedents were then resolved to their coreferents using a simple heuristic - match the most recently mentioned concept code occurring before the target coreferent in the essay. This combination of a word tagging model and a simple heuristic was more accurate at resolving anaphors than the two coreference resolution models. This illustrates the relative effectiveness of a simple domain-specific approach when compared to a more complex approach that was designed to solve a much more general problem.

The intent of Research Question 4 was to determine the most effective method for extracting a causal model from an entire essay. Two different approaches were developed to solve this problem. The first approach extended the shift-reduce dependency parser developed for Research Question 2 (see Section 5.4.3) to parse whole essays, rather than just sentences. While this approach produced a small improvement in the classification performance on one dataset, it performed worse on the other dataset when compared to the sentence parser. Building upon the work of Collins [35, 36], the second approach used a reranking model to rerank the causal models produced by a generative approach. A beam search was used to generate the top n parses per sentence using the sentence parser. Multiple different candidate causal models were then generated for each essay by computing all of the possible permutations of the causal relations generated from the parsed sentences. Finally, the reranking model was trained to rerank the candidate models and select the best model for each essay. By using an objective function that evaluates the accuracy of the whole causal model rather than optimizing the individual decisions used to construct it, the reranking model is much closer to optimizing the key evaluation metric, the overall micro-F_1 score. This approach consequently out-performed the essay parsing method, attaining a small improvement in performance across both datasets. Determining the optimal beam size was also critical to the success of this approach. Each dataset had a different optimal beam size, and finding the right beam size allowed for a higher overall micro-F_1 score by better balancing recall and precision.

8.2 Conclusions

A number of general conclusions can be drawn from this work that can be applied to the design of other machine learning systems:

- **Directly optimize the evaluation metric**. In Research Question 2, the shift reduce parser attained a higher classification accuracy by using a custom cost function that directly optimized the critical measure of classification performance, the micro-F_1 score. This was also observed in Research Question 4, when the reranking approach out-performed the sentence and essay parsing models by directly optimizing the micro-F_1 score for each essay. Furthermore, the algorithm's overall design allowed it to better balance recall and precision, thus optimizing the micro-F_1 score

- **Utilize the problem's structure**. For structured prediction problems, models that can make use of the problem's structure will generally out-perform more general purpose algorithms. In Research Question 2, I attempted to use the optimal word tagging model to label words with causal relations. However, the parsing model attained a higher overall performance by solving a more general problem - determining whether a causal relation exists between any two concept codes. By re-framing the problem to make better use of its structure, the parser model was able to generalize more effectively to new data. In Research Question 4, I also tried to extend the sentence parsing model from Research Question 2 to parse whole essays. Similarly, the reranking approach achieved a higher overall classification accuracy by optimizing the overall micro-F_1 score of each essay, rather than trying to learn to make a sequence of locally optimal parsing decisions.

- **Transfer learning between models is effective only if both models were trained on related datasets.** In Research Question 1, GloVe vectors [88, 168] trained on Wikipedia, which includes scientific content, enabled the RNN model to achieve a higher classification accuracy. However, in Research Question 3, two state-of-the-art pre-trained coreference resolution models achieved a very low classification accuracy when used to resolve coreferences in the essay data. Unlike the GloVe vectors, the Stanford and Berkeley Coreference Resolution Systems were trained on a very different dataset with little semantic overlap to the scientific essay text. This prevented the models from generalizing effectively on the essay text.

- **When stacking models, use the base model's confidence estimates**. Although the stacking approach evaluated in Research Question 2 was not the most accurate method for detecting causal relations, features computed from the base model's predicted probabilities were amongst the top features evaluated on the skin cancer dataset. Similarly, for the reranking model studied in Research Question 4, the top features for both datasets were the sentence parser's probability estimates for each parsed causal relation. When stacking machine learning models, better results are usually obtained when the confidence as well as the predictions of the base-models are used as the features [221, 198].

- **When stacking models, errors propagate upwards**. With the exception of the first research question, the models developed to solve each subsequent problem built upon the predictions of the previous models. When building a pipeline of models in this manner, errors from the earlier models tend to propagate to the downstream models [200]. For example, the performance of the causal-relation parsing models was very dependent on the accuracy of the word tagging models at detecting concept codes. Overall the techniques studied in Research Questions 3 and 4 (the coreference resolution model, the essay parser and the reranking algorithm), were the least effective at solving their respective problems. These models were also furthest removed from the original data, being reliant on predictions from the word tagging and causal relation extraction models. An end-to-end model, capable of learning to solve all of these sub-problems in one combined approach may mitigate these issues and produce a more accurate solution to the problem.

8.3 Future Work

There are a number of limitations to this work that could be addressed in future research. One limitations is the overall system design. Designing the system as a pipeline of models, where the predictions from each step in the pipeline are fed into the models used in later steps, has led to errors propagating through the pipeline, from the earlier models to the models used in the final pipeline steps. This is one reason why the techniques used to address Research Questions 3 and 4 were less effective than the models applied to the earlier Research Questions. Deep learning models have been developed than can solve many different problems using the same neural network architecture, for example the work of Collobert et al [39] who trained a single model capable of solving a variety of NLP tasks, including named entity recognition and semantic role labeling. It is possible that a system could be designed which could solve all 4 research questions using a single model. In some cases, multi-task learning can help a machine learning system generalize better to new data points due to transfer learning between tasks, as found in [39].

In chapter 4, for Research Question 1 we saw how the GloVe word embeddings improved the performance of the RNN model at the word tagging task. However, a limitation of word embedding models is that they learn a single representation for each word, regardless of how many different word senses (meanings) that word has. Recently, deep language models have been developed that can learn *contextualized word embeddings*, embeddings for words that are adjusted based on the context in which the word appears. These models have been to used to improve the state-of-the-art performance of a number of neural models on a variety of NLP tasks, such as question answering, sentiment analysis and coreference resolution [52, 171, 249]. In each of these systems, a pre-trained deep language model was incorporated into a deep neural network, and then fine-tuned on some other task. Use of a pre-trained deep-language model could improve how well the word tagging model generalizes on new data points, provided it was trained on a semantically related dataset. This would also improve the accuracy of the models reliant on its predictions, those used to solve research questions 2, 3 and 4. Furthermore, it might reduce the number of labeled essays needed to attain high classification accuracy, by utilizing transfer learning to allow the model to learn more efficiently on new data points.

The results of the statistical testing indicate that in most cases that it is hard to determine which of the algorithms studied was the most effective at a particular task. With only two datasets evaluated, it is hard to determine how well these techniques will generalize to other sets of scientific essays, or to other domains where causal relations could be extracted. Replicating this work using other datasets, especially datasets in domains other than scientific essays, would more effectively help determine the most accurate technique for solving each of these problems. This would also help us to better understand how well these techniques generalize to other topics, other types of essays and other problem domains, such as extracting different types of binary relations between concepts.

Additionally, one question that this work fails to answer is how well these techniques would perform at detecting causal relations in general, without being limited to only detecting the causal relations present in a pre-defined causal model. A number of techniques have been developed in discourse parsing that look for general occurrences of causal relations in natural language discourse (see Section 3.2). Future research could investigate methods of extending these techniques to detect models that are capable of detecting any type of causal relation, in texts from many different domains.

One final criticism of this work is that the models used to solve Research Question 2 don't accurately detect the rarer causal relations in the two datasets. Techniques exist for enabling machine learning models to learn more effectively when only a few labels are present. One area of research, called *few-shot learning*, develops techniques that are able to learn effectively from only a small number of examples of each label [238]. An extreme variant, called *one-shot learning*, studies models that can learn effectively given only a single example of each training label [66]. Utilizing techniques from few-shot or even one-shot learning could enable the causal-relation parsing models to detect the rare causal relations with a higher level of accuracy. Making use of transfer learning, for example using a deep-language model as part of a neural parsing model, could also help improve the classification accuracy on the rarer causal relations.

References

[1] Towards robust linguistic analysis using ontonotes. Web page, URL: http://cemantix.org/data/ontonotes.html accessed Jan. 04, 2019, 2013.

[2] Ghulum Bakiri and Thomas G Dietterich. Achieving high-accuracy text-to-speech with machine learning. *Data mining in speech synthesis*, 10, 1999.

[3] Ana LC Bazzan and Karim Pichara. *Advances in Artificial Intelligence–IBERAMIA 2014: 14th Ibero-American Conference on AI, Santiago de Chile, Chile, November 24-27, 2014, Proceedings*, volume 8864. Springer, 2014.

[4] R. Bellman and Rand Corporation. *Dynamic Programming*. Rand Corporation research study. Princeton University Press, 1957.

[5] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[6] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[7] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[8] Anders Björkelund and Jonas Kuhn. Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. In *Proceedings of the Association for Computational Linguistics*, 2014.

[9] Ezra Black, Fred Jelinek, John Lafferty, David M Magerman, Robert Mercer, and Salim Roukos. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings of the workshop*

on Speech and Natural Language, pages 134–139. Association for Computational Linguistics, 1992.

[10] Paul Black and Dylan Wiliam. *Inside the black box: Raising standards through classroom assessment.* Granada Learning, 1998.

[11] E. Blanco, N. Castell, and D. Moldovan. Causal relation extraction. In *LREC*, 2008.

[12] Matthew R Boutell, Jiebo Luo, Xipeng Shen, and Christopher M Brown. Learning multi-label scene classification. *Pattern recognition*, 37(9):1757–1771, 2004.

[13] E. Brill. *A corpus-based approach to language learning.* PhD thesis, University of Pennsylvania, Philadelphia, PA, 1993.

[14] M. A. Britt, P. Wiemer-Hastings, A. Larson, and C. Perfetti. Using intelligent feedback to improve sourcing and integration in students' essays. *International Journal of Artificial Intelligence in Education*, 14:359–374, 2004.

[15] M.A. Britt and C. Aglinskas. Improving students' ability to identify and use source information. *Cognition and Instruction*, 20(4):485–522, 2002.

[16] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

[17] Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.

[18] Jill Burstein, Karen Kukich, Susanne Wolff, Chi Lu, and Martin Chodorow. Enriching automated essay scoring using discourse marking. 2001.

[19] Jill Burstein, Daniel Marcu, and Kevin Knight. Finding the write stuff: Automatic identification of discourse structure in student essays. *IEEE Intelligent Systems*, 18(1):32–39, 2003.

[20] Xavier Carreras. Experiments with a higher-order projective dependency parser. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007.

[21] Du-Seong Chang and Key-Sun Choi. Causal relation extraction using cue phrase and lexical pair probabilities. In *International Conference on Natural Language Processing*, pages 61–70. Springer, 2004.

[22] Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180. Association for Computational Linguistics, 2005.

[23] Danqi Chen and Christopher D Manning. A fast and accurate dependency parser using neural networks. In *EMNLP*, pages 740–750, 2014.

[24] Weiwei Cheng and Eyke Hüllermeier. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2-3):211–225, 2009.

[25] Everton Alvares Cherman, Maria Carolina Monard, and Jean Metz. Multi-label problem transformation methods: a case study. *CLEI Electronic Journal*, 14(1):4–4, 2011.

[26] M. Chi, R. Roscoe, J. Slotta, M. Roy, and C. Chase. Misconceived causal explanations for emergent processes. *Cognitive Science*, 36:1–61, 2012.

[27] Jason PC Chiu and Eric Nichols. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370, 2016.

[28] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[29] Jinho D Choi and Martha Palmer. Guidelines for the clear style constituent to dependency conversion. *Technical Report 01–12*, 2012.

[30] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[31] Kevin Clark and Christopher D Manning. Deep reinforcement learning for mention-ranking coreference models. *arXiv preprint arXiv:1609.08667*, 2016.

[32] William G Cochran. The comparison of percentages in matched samples. *Biometrika*, 37(3/4):256–266, 1950.

[33] R. Cohen. Analyzing the structure of argumentative discourse. *Computational Linguistics*, 13(1-2):11–24, 1987.

[34] Michael Collins. Head-driven statistical methods for natural language parsing. *Unpublished PhD thesis, University of Pennsylvannia*, 1999.

[35] Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics, 2002.

[36] Michael Collins and Terry Koo. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70, 2005.

[37] Michael Collins and Brian Roark. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 111. Association for Computational Linguistics, 2004.

[38] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In *ICML*, pages 160–167, 2008.

[39] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[40] William Condon. Large-scale assessment, locally-developed measures, and automated scoring of essays: Fishing for red herrings? *Assessing Writing*, 18(1):100–108, 2013.

[41] Joana Costa, Catarina Silva, Mário Antunes, and Bernardete Ribeiro. Adaptive learning models evaluation in twitter's timelines. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[42] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):30–42, 2012.

[43] Harold Charles Daume. *Practical structured learning techniques for natural language processing*. ProQuest, 2006.

[44] Hal Daumé III, John Langford, and Daniel Marcu. Searn in practice. *Unpublished (available at http://pub. hal3. name/)*, 2006.

[45] Hal Daumé Iii, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

[46] Hal Daumé III and Daniel Marcu. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the 22nd international conference on Machine learning*, pages 169–176. ACM, 2005.

[47] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[48] Paul Deane. On the relation between automated essay scoring and modern views of the writing construct. *Assessing Writing*, 18(1):7 – 24, 2013. Automated Assessment of Writing.

[49] Krzysztof Dembczynski, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. On label dependence in multi-label classification. In *Workshop proceedings of learning from multi-label data*, pages 5–12. Citeseer, 2010.

[50] Janez Demšar. On the appropriateness of statistical tests in machine learning. In *Workshop on Evaluation Methods for Machine Learning in conjunction with ICML*, page 65, 2008.

[51] Li Deng, Jinyu Li, Jui-Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Mike Seltzer, Geoffrey Zweig, Xiaodong He, Julia Williams, et al. Recent advances in deep learning for speech research at microsoft. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8604–8608. IEEE, 2013.

[52] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[53] Thomas G Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923, 1998.

[54] Thomas G Dietterich. Machine learning for sequential data: A review. In *Structural, syntactic, and statistical pattern recognition*, pages 15–30. Springer, 2002.

[55] Thomas G Dietterich, Pedro Domingos, Lise Getoor, Stephen Muggleton, and Prasad Tadepalli. Structured machine learning: the next ten years. *Machine Learning*, 73(1):3–23, 2008.

[56] Semire Dikli. Automated essay scoring. *Online Submission*, 7(1):49–62, 2006.

[57] Timothy Dozat and Christopher D Manning. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*, 2016.

[58] Chris Drummond. Machine learning as an experimental science (revisited). In *Proceedings of the Twenty-First National Conference on Artificial Intelligence: Workshop on Evaluation Methods for Machine Learning*, pages 1–5, 2006.

[59] Greg Durrett and Dan Klein. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics*, 2:477–490, 2014.

[60] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*, 2015.

[61] Elad ET Eban, Mariano Schain, Alan Mackey, Ariel Gordon, Rif A Saurous, and Gal Elidan. Scalable learning of non-decomposable objectives. *arXiv preprint arXiv:1608.04802*, 2016.

[62] André Elisseeff and Jason Weston. A kernel method for multi-labelled classification. In *Advances in neural information processing systems*, pages 681–687, 2001.

[63] J.L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[64] Eleazar Eskin, Wenke Lee, and Salvatore J Stolfo. Modeling system calls for intrusion detection with dynamic window sizes. In *DARPA Information Survivability Conference &amp; Exposition II, 2001. DISCEX'01. Proceedings*, volume 1, pages 165–175. IEEE, 2001.

[65] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

[66] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

[67] Vanessa Wei Feng and Graeme Hirst. A linear-time bottom-up discourse parser with constraints and post-editing. In *ACL (1)*, pages 511–521, 2014.

[68] Peter W Foltz, Paula Hidalgo, and Alistair Van Moere. Improving student writing through automated formative assessment: Practices and results. 2014.

[69] Marisha Fonseca. The correct way to report p values. Web page, URL: https://www.editage.com/insights/the-correct-way-to-report-p-values accessed Nov. 17, 2019, 2013.

[70] Institute for Education Sciences. Reading for understanding across grades 6 through 12: Evidence-based argumentation for disciplinary learning. washington, d.c.: National center for education research, 2010. retrieved from `http://www.ies.ed.gov/ncer/projects/results.asp?ProgID=62&NameID-351` last accessed 2015-01-20.

[71] Yoav Freund and Robert E Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296, 1999.

[72] Daniel Fried, Mitchell Stern, and Dan Klein. Improving neural parsing by disentangling model combination and reranking effects. *arXiv preprint arXiv:1707.03058*, 2017.

[73] Katrin Fundel, Robert Küffner, and Ralf Zimmer. Relex—relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371, 2007.

[74] Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. Learning continuous phrase representations for translation modeling. *Proc. of ACL. Association for Computational Linguistics, June*, 2014.

[75] Salvador García, Alberto Fernández, Julián Luengo, and Francisco Herrera. A study of statistical techniques and performance measures for

genetics-based machine learning: accuracy and interpretability. *Soft Computing*, 13(10):959, 2009.

[76] Stuart Geman and Mark Johnson. Dynamic programming for parsing and estimation of stochastic unification-based grammars. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 279–286. Association for Computational Linguistics, 2002.

[77] Richard J Gerrig and Gail McKoon. The readiness is all: The functionality of memory-based text processing. *Discourse Processes*, 26(2-3):67–86, 1998.

[78] C Lee Giles. *Special issue on dynamic recurrent neural networks*. IEEE, 1994.

[79] R. Girju and D. Moldovan. Mining answers for causation questions. In *Proc. The AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases*, 2002.

[80] R. Girju, P. Nakov, V. Nastase, S. Szpakowicz, P. Turney, and D. Yuret. Semeval-2007 task 04: Classification of semantic relations between nominals. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, page 13–18, 2007.

[81] Claudio Giuliano, Alberto Lavelli, and Lorenza Romano. Exploiting shallow linguistic information for relation extraction from biomedical literature. In *EACL*, volume 18, pages 401–408. Citeseer, 2006.

[82] Shantanu Godbole and Sunita Sarawagi. Discriminative methods for multi-labeled classification. In *Advances in Knowledge Discovery and Data Mining*, pages 22–30. Springer, 2004.

[83] Yoav Goldberg and Michael Elhadad. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750. Association for Computational Linguistics, 2010.

[84] Yoav Goldberg and Joakim Nivre. A dynamic oracle for arc-eager dependency parsing. *Proceedings of COLING 2012*, pages 959–976, 2012.

[85] Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech*

and Signal Processing (ICASSP), 2013 IEEE International Conference on, pages 6645–6649. IEEE, 2013.

[86] Alex Graves et al. *Supervised sequence labelling with recurrent neural networks*, volume 385. Springer, 2012.

[87] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(5):855–868, 2009.

[88] Stanford University NLP Group. Pre-trained GloVe Word Embedding Vectors. `http://nlp.stanford.edu/data/glove.6B.zip`, 2017. [Online; accessed 25-March-2017].

[89] The Berkeley NLP Group. Berkeley Entity Resolution System. `http://nlp.cs.berkeley.edu/projects/entity.shtml`, 2018 (last accessed December 8th, 2018). [Online; accessed 8-December-2018].

[90] The Stanford Natural Language Processing Group. Stanford Coreference Resolution System. `https://stanfordnlp.github.io/CoreNLP/coref.html`, 2018 (last accessed December 8th, 2018). [Online; accessed 8-December-2018].

[91] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[92] Zellig S Harris. Distributional structure, 1954.

[93] Muhammad Fahim Hasan, Naushad UzZaman, and Mumit Khan. Comparison of Unigram, Bigram, HMM and Brill's POS tagging approaches for some South Asian languages. 2007.

[94] P. Hastings, M.A Britt, K. Rupp, K. Kopp, and S. Hughes. Computational analysis of explanatory essay structure. In Keith Millis, Debra Long, Joseph P. Magliano, and Katja Wiemer, editors, *Multi-Disciplinary Approaches to Deep Learning*. Routledge, New York, 2019.

[95] Peter Hastings, Simon Hughes, Anne Britt, Dylan Blaum, and Patty Wallace. Toward automatic inference of causal structure in student essays. In *Intelligent Tutoring Systems*, pages 266–271. Springer, 2014.

[96] Peter Hastings, Simon Hughes, and M Anne Britt. Active learning for improving machine learning of student explanatory essays. In *International Conference on Artificial Intelligence in Education*, pages 140–153. Springer, 2018.

[97] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[98] J Hemmerich and Jennifer Wiley. Do argumentation tasks promote conceptual change about volcanoes. In *Proceedings of the twenty-fourth annual conference of the Cognitive Science Society*, pages 453–458. Erlbaum Hillsdale, NJ, 2002.

[99] Jerry R Hobbs. Coherence and coreference. *Cognitive science*, 3(1):67–90, 1979.

[100] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[101] D Holt and TM Fred Smith. Post stratification. *Journal of the Royal Statistical Society: Series A (General)*, 142(1):33–46, 1979.

[102] Matthew Honnibal, Yoav Goldberg, and Mark Johnson. A non-monotonic arc-eager transition system for dependency parsing. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 163–172. Citeseer, 2013.

[103] H. Seeker W. Haußman E. Honnibol, M. Peters and Montani I. spacy. `https://github.com/spacy-io/spaCy`, 2013.

[104] D Hubel and T. Wiesel. Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160:106–154, 1962.

[105] Simon Hughes, Peter Hastings, M Anne Britt, Patricia Wallace, and Dylan Blaum. Machine learning for holistic evaluation of scientific essays.

[106] Simon Hughes, Peter Hastings, Joseph Magliano, Susan Goldman, and Kimberly Lawless. Automated approaches for detecting integration in student essays. In *Intelligent Tutoring Systems*, pages 274–279. Springer, 2012.

[107] Achieve Inc. *Next Generation Science Standards.* Achieve Inc., 2013.

[108] Mohit Iyyer, Jordan L Boyd-Graber, Leonardo Max Batista Claudino, Richard Socher, and Hal Daumé III. A neural network for factoid question answering over paragraphs. In *EMNLP*, pages 633–644, 2014.

[109] Nathalie Japkowicz and Mohak Shah. *Evaluating learning algorithms: a classification perspective*. Cambridge University Press, 2011.

[110] MI Jordan. Serial order: a parallel distributed processing approach. technical report, june 1985-march 1986. Technical report, California Univ., San Diego, La Jolla (USA). Inst. for Cognitive Science, 1986.

[111] Shafiq Joty, Giuseppe Carenini, and Raymond T Ng. A novel discriminative framework for sentence-level discourse analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 904–915. Association for Computational Linguistics, 2012.

[112] Shafiq R Joty, Giuseppe Carenini, Raymond T Ng, and Yashar Mehdad. Combining intra-and multi-sentential rhetorical parsing for document-level discourse analysis. In *ACL (1)*, pages 486–496, 2013.

[113] D. Jurafsky and J. Martin. *Speech and Language Processing: An introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice-Hall, New York, 2000.

[114] Piotr Juszczak, D Tax, and Robert PW Duin. Feature scaling in support vector data description. In *Proc. ASCI*, pages 95–102. Citeseer, 2002.

[115] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[116] Ronald T Kellogg and Bascom A Raulerson. Improving the writing skills of college students. *Psychonomic bulletin & review*, 14(2):237–242, 2007.

[117] Christopher SG Khoo, Syin Chan, and Yun Niu. Extracting causal knowledge from a medical database using graphical patterns. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 336–343. Association for Computational Linguistics, 2000.

[118] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[119] W. Kintsch. The role of knowledge in discourse comprehension: A constructive integration model. *Psychological Review*, 95:163–182, 1988.

[120] Walter Kintsch. Text comprehension, memory, and learning. *American Psychologist*, 49(4):294, 1994.

[121] Terry Koo and Michael Collins. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11. Association for Computational Linguistics, 2010.

[122] Alex Krizhevsky and G Hinton. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40, 2010.

[123] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[124] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[125] Ludmila I Kuncheva. *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2014.

[126] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[127] Thomas K Landauer, Karen E Lochbaum, and Scott Dooley. A new formative assessment technology for reading and writing. *Theory into Practice*, 48(1):44–52, 2009.

[128] Leah S Larkey and W Bruce Croft. A text categorization approach to automated essay grading. *Automated Essay Scoring: A Cross-discipline Perspective: Mahwah, NJ, Lawrence Erlbaum*, 2003.

[129] Phuong Le-Hong, Xuan-Hieu Phan, and The-Trung Tran. On the effect of the label bias problem in part-of-speech tagging. In *Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2013 IEEE RIVF International Conference on*, pages 103–108. IEEE, 2013.

[130] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[131] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *ACL (2)*, pages 302–308, 2014.

[132] Linguistic Data Consortium. Ontonotes release 4.0. Web page, 2011. url: https://catalog.ldc.upenn.edu/LDC2011T03, accessed April 14, 2019.

[133] Linguistic Data Consortium. Ontonotes release 5.0. Web page, 2011. url: https://catalog.ldc.upenn.edu/LDC2013T19, accessed April 14, 2019.

[134] Dimitris Liparas and Evangelia Pantraki. An evolutionary improvement of the mahalanobis–taguchi strategy and its application to intrusion detection. In *International Conference on Advanced Information Systems Engineering*, pages 16–30. Springer, 2014.

[135] Oscar Luaces, Jorge Díez, José Barranquero, Juan José del Coz, and Antonio Bahamonde. Binary relevance efficacy for multilabel classification. *Progress in Artificial Intelligence*, 1(4):303–313, 2012.

[136] Oscar Luaces, Jorge Díez, José Barranquero, Juan José del Coz, and Antonio Bahamonde. Binary relevance efficacy for multilabel classification. *Progress in Artificial Intelligence*, 1(4):303–313, 2012.

[137] Thang Luong, Richard Socher, and Christopher D Manning. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113, 2013.

[138] William C Mann and Sandra A Thompson. *Rhetorical structure theory: A theory of text organization*. University of Southern California, Information Sciences Institute, 1987.

[139] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Massachusetts, 1999.

[140] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[141] M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330, 1993.

[142] Andrew McCallum, Dayne Freitag, and Fernando CN Pereira. Maximum entropy markov models for information extraction and segmentation. In *ICML*, volume 17, pages 591–598, 2000.

[143] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[144] Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947.

[145] Eitan Menahem, Lior Rokach, and Yuval Elovici. Troika–an improved stacking schema for classification tasks. *Information Sciences*, 179(24):4097–4122, 2009.

[146] Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Honza Černockỳ, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE, 2011.

[147] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[148] T. Mitchell. *Machine Learning (Mcgraw-Hill International Edit)*. McGraw-Hill Education (ISE Editions), 1st edition, October 1997.

[149] S Sorower Mohammad. A literature survey on algorithms for multi-label learning. *Oregon State University, Corvallis*, 2010.

[150] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[151] Renato Negrinho, Matthew Gormley, and Geoffrey J Gordon. Learning beam search policies via imitation learning. In *Advances in Neural Information Processing Systems*, pages 10652–10661, 2018.

[152] Dat Quoc Nguyen and Karin Verspoor. An improved neural network model for joint pos tagging and dependency parsing. *arXiv preprint arXiv:1807.03955*, 2018.

[153] Nam Nguyen and Yunsong Guo. Comparisons of sequence labeling algorithms and extensions. In *Proceedings of the 24th international conference on Machine learning*, pages 681–688. ACM, 2007.

[154] Joakim Nivre. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT*. Citeseer, 2003.

[155] Joakim Nivre. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57. Association for Computational Linguistics, 2004.

[156] Joakim Nivre. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553, 2008.

[157] Joakim Nivre. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 351–359. Association for Computational Linguistics, 2009.

[158] Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135, 2007.

[159] Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiğit, and Svetoslav Marinov. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 221–225. Association for Computational Linguistics, 2006.

[160] How to write a spelling corrector. `http://norvig.com/spell-correct.html`.

[161] The Council of Chief State School Officers. The common core standards for english language arts and literacy in history/social studies and science and technical subjects. Washington, DC: National Governors Association for Best Practices, 2010. `http://www.corestandards.org`.

[162] Naoaki Okazaki. Crfsuite: a fast implementation of conditional random fields (crfs), 2007.

[163] Ellis Batten Page. Computer grading of student prose, using modern concepts and software. *The Journal of experimental education*, 62(2):127–142, 1994.

[164] Ellis Batten Page. Project essay grade: Peg. *Automated essay scoring: A cross-disciplinary perspective*, pages 43–54, 2003.

[165] Mariusz Paradowski, Michał Spytkowski, and Halina Kwaśnicka. A new f-score gradient-based training rule for the linear model. *Pattern Analysis and Applications*, 22(2):537–548, 2019.

[166] Romain Paulus, Richard Socher, and Christopher D Manning. Global belief recursive neural networks. In *Advances in Neural Information Processing Systems*, pages 2888–2896, 2014.

[167] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[168] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[169] Leonid Peshkin, Avi Pfeffer, and Virginia Savova. Bayesian nets in syntactic categorization of novel words. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers-Volume 2*, pages 79–81. Association for Computational Linguistics, 2003.

[170] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[171] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[172] Barbara Plank, Anders Søgaard, and Yoav Goldberg. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *arXiv preprint arXiv:1604.05529*, 2016.

[173] Vasin Punyakanok, Dan Roth, Wen-tau Yih, and Dav Zimak. Learning and inference over constrained output. In *IJCAI*, volume 5, pages 1124–9, 2005.

[174] Ning Qian and Terrence J Sejnowski. Predicting the secondary structure of globular proteins using neural network models. *Journal of molecular biology*, 202(4):865–884, 1988.

[175] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[176] Sebastian Raschka. Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack. *J. Open Source Software*, 3(24):638, 2018.

[177] Adwait Ratnaparkhi. A linear observed time statistical parser based on maximum entropy models. *arXiv preprint cmp-lg/9706014*, 1997.

[178] Adwait Ratnaparkhi. *Maximum entropy models for natural language ambiguity resolution.* PhD thesis, University of Pennsylvania, 1998.

[179] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine learning*, 85(3):333–359, 2011.

[180] David Reitter. Simple signals for complex rhetorics: On rhetorical analysis with rich-feature support vector models. In *LDV Forum*, volume 18, pages 38–52, 2003.

[181] Mehwish Riaz and Roxana Girju. In-depth exploitation of noun and verb semantics to identify causation in verb-noun pairs. In *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 161, 2014.

[182] Mehwish Riaz and Roxana Girju. Recognizing causality in verb-noun pairs via noun and verb semantics. *EACL 2014*, page 48, 2014.

[183] Bryan Rink, Cosmin Adrian Bejan, and Sanda M. Harabagiu. Learning textual graph patterns to detect causal event relations. In Hans W. Guesgen and R. Charles Murray, editors, *FLAIRS Conference*. AAAI Press, 2010.

[184] Lior Rokach, Alon Schclar, and Ehud Itach. Ensemble methods for multi-label classification. *Expert Systems with Applications*, 41(16):7507–7523, 2014.

[185] Rod D Roscoe, Scott A Crossley, Erica L Snow, Laura K Varner, and Danielle S McNamara. Writing quality, knowledge, and comprehension correlates of human and automated essay scoring. In *The Twenty-Seventh International Flairs Conference*, 2014.

[186] Frank Rosenblatt. *Principles of neurodynamics; perceptrons and the theory of brain mechanisms.* Spartan Books, 1962.

[187] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *International Conference on Artificial Intelligence and Statistics*, pages 661–668, 2010.

[188] Dan Roth and Wen-tau Yih. A linear programming formulation for global inference in natural language tasks. Technical report, DTIC Document, 2004.

[189] Jean-François Rouet, M Anne Britt, Robert A Mason, and Charles A Perfetti. Using multiple sources of evidence to reason about history. *Journal of Educational Psychology*, 88(3):478, 1996.

[190] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[191] Rudolph J Rummel. Understanding correlation. *Honolulu: Department of Political Science, University of Hawaii*, 1976.

[192] Stuart Russell, Peter Norvig, and Artificial Intelligence. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25:27, 1995.

[193] Steven L Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data mining and knowledge discovery*, 1(3):317–328, 1997.

[194] Enrique Sánchez-Villamil, Mikel L Forcada, and Rafael C Carrasco. Unsupervised training of a finite-state sliding-window part-of-speech tagger. In *Advances in Natural Language Processing*, pages 454–463. Springer, 2004.

[195] Virginia Savova and Leonid Peshkin. Dependency parsing with dynamic bayesian network. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 20, page 1112.

Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

[196] Robert E Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine learning*, 39(2):135–168, 2000.

[197] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[198] Alexander K Seewald. Exploring the parameter state space of stacking. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 685–688. IEEE, 2002.

[199] Burr Settles. Active learning literature survey. university of wisconsin. Technical report, Madison Tech. Report, 2010.

[200] Martin Sewell. Ensemble learning. *RN*, 11(02), 2008.

[201] Mohammad Shahrokh Esfahani and Edward R Dougherty. Effect of separate sampling on classification accuracy. *Bioinformatics*, 30(2):242–250, 2013.

[202] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Gregoire Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110. ACM, 2014.

[203] Mark D Shermis and Jill C Burstein. *Automated essay scoring: A cross-disciplinary perspective*. Routledge, 2003.

[204] Mark D Shermis and Ben Hamner. Chapter 19: Contrasting state-of-the-art automated scoring of essays. *Handbook of automated essay evaluation: Current applications and new directions*, page 313, 2013.

[205] MD Shermis, JC Burstein, and L Bliss. The impact of automated essay scoring on high stakes writing assessments. In *annual meeting of the National Council on Measurement in Education*, 2004.

[206] R. Socher, J. Pennington, E. Huang, A. Ng, and C. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP*, pages 151–161. ACL, 2011.

[207] Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. Parsing with compositional vector grammars. In *ACL (1)*, pages 455–465. Association for Computer Linguistics, 2013.

[208] Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 129–136, 2011.

[209] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 1631, page 1642. Citeseer, 2013.

[210] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45(4):427–437, 2009.

[211] Jialin Song, Ravi Lanka, Albert Zhao, Yisong Yue, and Masahiro Ono. Learning to search via retrospective imitation. *arXiv preprint arXiv:1804.00846*, 2018.

[212] Antonio Sorgente, Giuseppe Vettigli, and Francesco Mele. Automatic extraction of cause-effect relations in natural language text. In *DART@AI*IA*, pages 37–48. Citeseer, 2013.

[213] Radu Soricut and Daniel Marcu. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 149–156. Association for Computational Linguistics, 2003.

[214] Eleftherios Spyromitros, Grigorios Tsoumakas, and Ioannis Vlahavas. An empirical study of lazy multilabel classification algorithms. In *Artificial Intelligence: Theories, Models and Applications*, pages 401–406. Springer, 2008.

[215] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[216] Christian Stab and Iryna Gurevych. Identifying argumentative discourse structures in persuasive essays. In *EMNLP*, pages 46–56, 2014.

[217] Rajen Subba and Barbara Di Eugenio. An effective discourse parser that uses rich linguistic information. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 566–574. Association for Computational Linguistics, 2009.

[218] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[219] Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd international conference on Machine learning*, pages 896–903. ACM, 2005.

[220] Simone Teufel. Argumentative zoning. *Univ. of Edinburgh, Edinburgh*, 1999.

[221] Kai Ming Ting and Ian H Witten. Issues in stacked generalization. *J. Artif. Intell. Res.(JAIR)*, 10:271–289, 1999.

[222] Kristina Toutanova, Aria Haghighi, and Christopher D Manning. Joint learning improves semantic role labeling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 589–596. Association for Computational Linguistics, 2005.

[223] Son N Tran, Qing Zhang, Anthony Nguyen, Xuan-Son Vu, and Son Ngo. Improving recurrent neural networks with predictive propagation for sequence labelling. In *International Conference on Neural Information Processing*, pages 452–462. Springer, 2018.

[224] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. In *Journal of Machine Learning Research*, pages 1453–1484, 2005.

[225] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *Dept. of Informatics, Aristotle University of Thessaloniki, Greece*, 2006.

[226] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Random k-labelsets for multilabel classification. *Knowledge and Data Engineering, IEEE Transactions on*, 23(7):1079–1089, 2011.

[227] Grigorios Tsoumakas, Ioannis Partalas, and Ioannis Vlahavas. An ensemble pruning primer. In *Applications of supervised and unsupervised ensemble methods*, pages 1–13. Springer, 2009.

[228] Peter D Turney, Patrick Pantel, et al. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188, 2010.

[229] Paul Van den Broek, Kirsten Risden, Charles R Fletcher, and Richard Thurlow. A "landscape" view of reading: Fluctuating patterns of activation and the construction of a stable memory representation. *Models of understanding text*, pages 165–187, 1996.

[230] Teun Adrianus Van Dijk, Walter Kintsch, and Teun Adrianus Van Dijk. *Strategies of discourse comprehension*. Citeseer, 1983.

[231] V Vapnik and Vlamimir Vapnik. Statistical learning theory wiley. *New York*, pages 156–160, 1998.

[232] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[233] Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. Decision trees for hierarchical multi-label classification. *Machine Learning*, 73(2):185–214, 2008.

[234] Ludo Verhoeven and Charles Perfetti. Advances in text comprehension: Model, process and development. *Applied Cognitive Psychology*, 22(3):293–301, 2008.

[235] Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269, 1967.

[236] Svitlana Volkova, Doina Caragea, William H Hsu, John Drouhard, and Landon Fowles. Boosting biomedical entity extraction by using syntactic patterns for semantic relation discovery. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, volume 1, pages 272–278. IEEE, 2010.

[237] Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. A unified tagging solution: Bidirectional lstm recurrent neural network with word embedding. *arXiv preprint arXiv:1511.00215*, 2015.

[238] Yaqing Wang and Quanming Yao. Few-shot learning: A survey. *CoRR*, abs/1904.05046, 2019.

[239] William C Ward and Randy Elliot Bennett. *Construction versus choice in cognitive measurement: Issues in constructed response, performance testing, and portfolio assessment*. Routledge, 2012.

[240] Ronald L Wasserstein, Nicole A Lazar, et al. The asa's statement on p-values: context, process, and purpose. *The American Statistician*, 70(2):129–133, 2016.

[241] Lloyd R Welch. Hidden markov models and the baum-welch algorithm. *IEEE Information Theory Society Newsletter*, 53(4):10–13, 2003.

[242] B. White and J. Frederiksen. Causal model progressions as a foundation for intelligent learning environments. *Artificial Intelligence*, 42:99–157, 1990.

[243] J. Wiley, S.R. Goldman, A. Graesser, C. Sanchez, I. Ash, and J. Hemmerich. Source evaluation, comprehension, and learning in internet science inquiry tasks. *American Educational Research Journal*, 46(4):1060–1106, 2009.

[244] Jennifer Wiley. Supporting understanding through task and browser design. In *Proceedings of the twenty-third annual conference of the Cognitive Science Society*, pages 1164–1169. Erlbaum Hillsdale, NJ, 2001.

[245] Jennifer Wiley and James F Voss. Constructing arguments from multiple sources: Tasks that promote understanding and not just memory for text. *Journal of Educational Psychology*, 91(2):301, 1999.

[246] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

[247] Yongwei Yang, Chad W Buckendahl, Piotr J Juszkiewicz, and Dennison S Bhola. A review of strategies for validating computer-automated scoring. *Applied Measurement in Education*, 15(4):391–412, 2002.

[248] Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W Cohen. Breaking the softmax bottleneck: A high-rank rnn language model. *arXiv preprint arXiv:1711.03953*, 2017.

[249] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.

[250] Majid Yazdani and James Henderson. Incremental recurrent neural network dependency parser with search-based discriminative training. 2015.

[251] Seunghyun Yoon, Hyeongu Yun, Yuna Kim, Gyu-tae Park, and Kyomin Jung. Efficient transfer learning schemes for personalized language modeling using recurrent neural network. *arXiv preprint arXiv:1701.03578*, 2017.

[252] Heng Yu, Liang Huang, Haitao Mi, and Kai Zhao. Max-violation perceptron and forced decoding for scalable mt training. In *EMNLP*, pages 1112–1123, 2013.

[253] Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *Knowledge and Data Engineering, IEEE Transactions on*, 18(10):1338–1351, 2006.

[254] Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048, 2007.

[255] Xiang Zhang and Yann LeCun. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*, 2015.

[256] Yuchen Zhang and Nianwen Xue. Neural ranking models for temporal dependency structure parsing. *arXiv preprint arXiv:1809.00370*, 2018.

[257] Yue Zhang and Stephen Clark. Syntactic processing using the generalized perceptron and beam search. *Computational linguistics*, 37(1):105–151, 2011.

[258] Yue Zhang and Joakim Nivre. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 188–193. Association for Computational Linguistics, 2011.

[259] Rolf A Zwaan. Situation models the mental leap into imagined worlds. *Current Directions in Psychological Science*, 8(1):15–18, 1999.

Appendices

Appendix A

Essay Prompts

The essay prompts below were created by researchers from the Northern Illinois University Department of Psychology.

A.1 Coral Bleaching

One purpose of reading in science is to understand the causes of scientific phenomena; in other words, we read to understand how and why things happen. To do this, we often need to gather information from multiple sources.

Today you will be reading about what causes "coral bleaching". Coral, which lives in the ocean, can be many different colors, but sometimes it loses its color and turns white. You will have to piece together important information across multiple sources to construct a good explanation of how and why this happens. No single source will provide all of the important parts of the explanation. Instead, you are the one making connections across sources and coming up with an explanation.

Your task is to read the following set of sources to help you understand and explain what leads to differences in the rates of coral bleaching.

While reading, it is important to show your thinking by making notes in the margins or on the texts.

You will be asked to answer questions and use specific information from the sources to support your ideas and conclusions.

You can read the sources in any order you wish, but you should read the sheet called "Background: What is 'Coral Bleaching?'" first, because it gives general information on the topic.

A.2 Skin Cancer

One purpose of reading in science is to understand the causes of scientific phenomena; in other words, we read to understand how and why things happen. To do this, we often need to gather information from multiple sources.

Today you will be reading about what causes some people to experience abnormal cell growth like skin cancer. You will have to piece together important information across multiple sources to construct a good explanation of how and why this happens. No single source will provide all of the important pieces of the explanation. Instead, you are the one making connections across sources and coming up with an explanation.

Your task is to read the following set of sources to help you understand and explain what leads to differences in the risk of developing skin cancer.

While reading, it is important to show your thinking by making notes in the margins or on the texts.

You will be asked to answer questions and use specific information from the sources to support your ideas and conclusions.

You can read the sources in any order you wish, but you should read the sheet called "Background: Skin Damage" first, because it gives general information on the topic.

Appendix B

Example Source Documents

The sample source documents below were taken from a set of source texts compiled for the essay writing task, to help the students answer the two essay prompts listed in appendix A. They were created by researchers from the Northern Illinois University Department of Psychology after the initial causal models were decided upon.

B.1 Coral Bleaching

B.1.1 Background: What is "Coral Bleaching?"

Marine biologists have discovered many different types of corals living in the oceans. These invertebrate animals live together in colonies and tend to stay in one place. The different types of algae living within the coral polyps give the corals their varying colors (see Figure B.1).

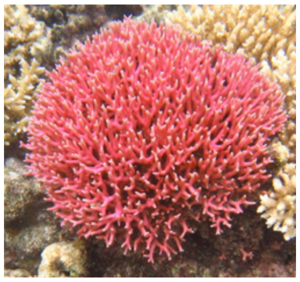


Figure B.1: A healthy coral

Some coral have been “bleached” a plain white. Coral bleaching is a phenomenon in which coral loses its color. Events leading to coral bleaching are a serious problem with a serious impact on the world’s coral reefs.

As can be seen in Figure B.2, coral bleaching is most noticeable in the Pacific Ocean. This ocean covers about 1/3 of the surface of the entire globe, and contains double the amount of water found in the Atlantic Ocean.

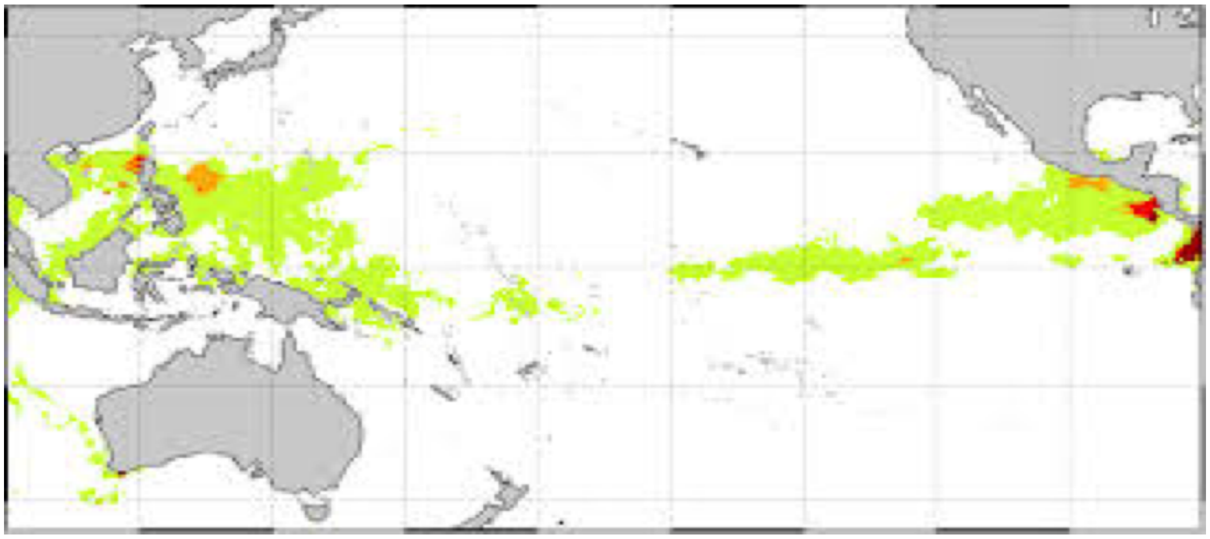


Figure B.2: This map of the Pacific Ocean shows coastal regions most affected by coral bleaching (darker areas near land masses).

B.1.2 Coral Bleaching Reports per Year

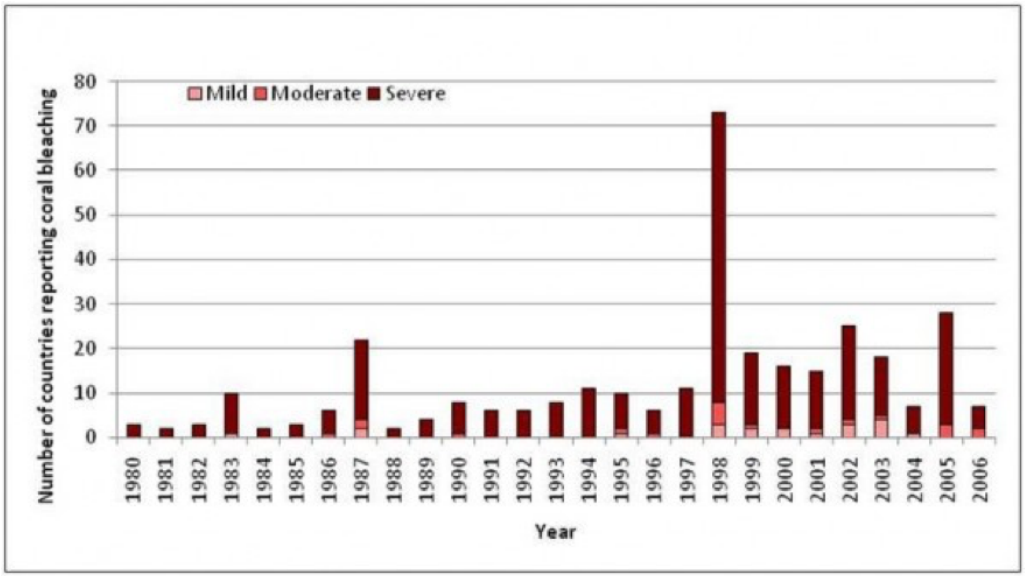


Figure B.3: Coral Bleaching Reports Per Year

B.2 Skin Cancer

B.2.1 Background: Skin Damage

It may surprise you to learn that the skin on our bodies is our largest organ (see Figure B.4). It covers every region of our bodies in order to protect our inner tissue from infection and loss of water. In addition, our skin helps regulate our body temperature. Although we take our skin for granted, there are several ways for things to go wrong with our skin.

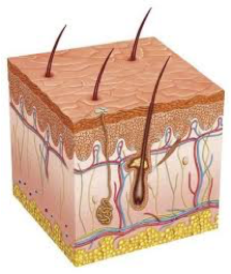


Figure B.4: A section of healthy skin

There are numerous skin disorders, conditions, and diseases. Of these, skin cancer is among the most feared because everyone is at some risk of developing skin cancer, but some people are at a higher risk than others. Additionally, skin cancer is the most common form of cancer in the United States (Figure B.5 shows a patch of skin cancer). Skin cancer is the uncontrolled growth of abnormal skin cells. The variety of skin cancer that develops depends on the type of skin cell that reproduces irregularly.

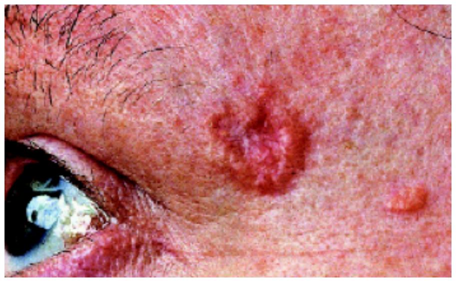


Figure B.5: The dark patch to the right of this person's eye is caused by a basal cell carcinoma.

There are three main varieties of skin cancer: basal cell carcinoma, squamous cell carcinoma, and malignant melanoma. Together, basal and squamous cell carcinomas make up approximately 95 percent of skin cancers. Malignant melanoma only occurs in approximately 5 percent of skin cancer cases. However, malignant melanoma is responsible for the most deaths from skin cancer. Checking your skin for suspicious changes can help with detecting skin cancer at its earliest stages. Early detection of skin cancer gives you the maximum chance for successful treatment.

B.2.2 Latitude and Direct Sunlight

A common way to locate points on the surface of the earth is by geographic coordinates (see Figure B.6).

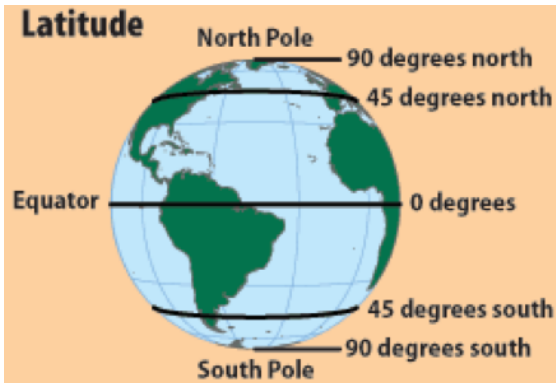


Figure B.6: Important lines of latitude

These geographic coordinates are called latitude and longitude. Latitude and Longitude are measured in degrees and represent distances from the center of the Earth. We can imagine the Earth as a sphere, with an axis around which it spins. The ends of this axis are the North and South Poles. The Equator is an imaginary line around the Earth at 0 degrees latitude. Latitude values indicate the distance between the Equator and points north or south of it. Depending on location, amount of direct sunlight may vary a lot or a little throughout the year. As a rule of thumb, the closer you are to the Equator, the more consistent direct sunlight will be. This means people closer to the equator need to be more aware of the dangers of direct sunlight.Those who live in areas with less direct sunlight may be less concerned about any dangers, although there are still some risks associated with sun exposure wherever you live and at any time of year. Some locations are more likely to have moderate to extreme levels of year-round direct sunlight. Examples of this include the Northern third of Australia and the Southern parts of the United States. The most year-round direct sunlight occurs between the Tropics of Cancer (23°N) and Capricorn (23°S). Due to the amount of direct sunlight in these areas, the amount of UVb radiation is also high. Generally speaking, the more direct sunlight there is, the more UVb radiation there is.

Appendix C

Essay Scoring and Inter-Rater Reliability Procedure

Three individuals were trained as coders for the pre and post EBA essay data. Coders were trained on materials for each topic (coral bleaching or skin cancer) using annotated versions of the five documents, causal models indicating numberings associated with concepts in the models, and spreadsheets of ideal answers and vague answers. Two individuals were trained on only one topic, and a third coder was trained on both. The single topic coders were responsible for scoring every essay on their topic, whereas the double topic coder was responsible for scoring 20% of the essays for each topic. Thus, two coders scored 20% of the essays, and one coder scored the remaining 80

Training on the scoring process began with a meeting to discuss the causal models, the scoring structure, and the numberings associated with each concept code. All three coders were given practice essays from a previous round of data collection using the same science topics and tasks. Cohen's Kappa was used to establish inter-rater reliability. To do this, the 13 concept codes in the coral bleaching model and the 9 concept codes in the skin cancer model were displayed vertically in a spreadsheet for each participant's essay. The cells in the adjacent columns were filled with 1s and 0s depending on whether a given concept code was included in that coder's compiled claim. A Kappa score was calculated based on these sets of 1s and 0s. The Kappa scores for the three rounds of training were .76, .84, and .94 for the coral bleaching essays and .90, .93, and .97 for the skin cancer essays.

Following this, the two single topic coders began scoring sets of essays with each set consisting of about 1/6th of the total set of essays. After each set of essays was scored, the double topic coder randomly selected 20score. Kappa scores were calculated for each round of essays, and disagreements were reconciled through discussion. This allowed for consistency in scoring throughout the essay scoring time frame. The Kappa scores for the coral bleaching essay sets were .75, .89, .85, .86, .86, and .93. The Kappa scores for the skin cancer essay sets were .64, .92, .88, .89, .85, and .93.

Appendix D

Example Essays

D.1 Coral Bleaching

D.1.1 Example 1

The world's temperatures are constantly changing. This is one reason coral bleaching goes on the in the world. Another reason coral bleaching happens is because of us humans. And because sometimes they can't get what they need to stay healthy. The first reason coral bleaching happens is because of the tempatures that constantly change. The weather of course affects water temperature. When the water is too warm, the carbon dioxide decreases. According to "Coral & Photosynthesis" "Changes in the amount of CO2 threaten the delicate balance required to keep corals healthy." So if coral don't get the right amount, they could turn white. Second, us humans are good at destroying many things and coral is one of them. "Examples of this include, blast fishing and tourists who drop anchors or walk on reefs." From this, I think that with those things, the zooxantheallae are caused either crushed or scratched off from the coral. And the last cause of coral bleaching is not having the right circumstances to keep them healthy. If they don't have enough zooxanthallae, then they turn white. They loose them because they might get stressed. I believe this happens because of weather changes. Now that I have learned what coral bleaching is, I believe it happens because of the temperatures, people, and not getting enough of what they need.

D.1.2 Example 2

Coral bleaching can be caused by many things, they all have to do with something that happens to the symbiotic relationship they have with the zooxanthellae. The reason for this is because of the fact that if something happens to the zooxanthellae that causes them to basically throw out the food they had, which the coral eat, then the coral starts to starving and looses its color. Since the zooxanthellae alos gives the coral its color. This leads to the "look" of coral bleaching which gives it it's name. In reality it has nothing to do with bleach like you'd think it would it's just nameed like that because of, again, the "look." The rates are kind of different/inconsistent every year. The highest report though was in 1998 for more than 70 reports. The reates aren't really specific, at least not to me, so I infer that the rates of coral bleaching happening are risning since in the text it clearly states that coral bleaching happens because a disturbance in the environment caused the zooxanthellae to be ejected from the coral. With that being said we know how bad weather condtions invasive we're become when it comes to animals and their habitats. That's why I infer that the differences in the rates of coral bleaching are increasing due to environment changes in the coral's habitats that are primarily caused by us humans.

D.1.3 Example 3

There are many things that lead to differences in the rates of coral bleaching. One of them is how things change and threatens the corals health. For example; in the article coral & photosynthesis it says, "As water temperature increases, the amount of carbon dioxide (CO2) in water decreases. Changes in the amount of Co2 threaten the delicate balance required to keep corals healthy." Meaning that they need a specific amount, and if it changes, the corals won't be healthy. What is coral bleaching? Coral bleaching is a phenomenon in which coral loses its color. Coral bleaching can lead to serious impact on the world's coral reefs. The # of countries reporting coral coral bleaching every year goes up down, up down, up down throughout the years. Not until 1998 were about 73 or 74 countries reported coral bleaching. On the article coral and zooxanthallae it says, "For example, a massive coral bleaching event in 1998 is considered one of the worst ever observed. This event resulted in the death of 16% of the worlds coral reefs" Meaning that all of the years before they had trouble but not as bad as the one in that year. In conclusion, there are many things that leads to differences in the rates of coral bleaching most of it is the environmental change, which can cause stress to corals.

D.2 Skin Cancer

D.2.1 Example 1

What leads into the risk of developing skin cancer is different ways such as really bad sunburn, being exposed to some kind of dangerous radiation, or something that goes wrong in our skin. As stated in the text, there are three kinds of skin cancer, basal Cell carcinoma, squamous cell carcinoma, and malignant melanoma. The most hamrful type of skin ancer is malignant melanoma. It has caused the most deaths. Lets talk about how sun burn is a big way to get skin cancer. When sunburn happens, blood gets directed from your body to the sunburnt skin to try and repair it. The damaged cells will get replaced with healthy ones. When sunburn is red, it is the blood flow trying to repair the skin. If you have severe enough sunburn, it is called sun poising, which can lead to infection, shock, or even death. So, if you have severe sunburn that does not heal, you will have many dead skin cells which is not healthy. After looking at this evidence, I think skin cancer is caused by really bad sunburn which causes skin cells to die and something going wrong inside skin cells. We may never know exactly how or why but looking at evidence it looks like those are a part of the cause.

D.2.2 Example 2

Skin cancer is most likely the cancer that is in the back of everyones mind, yet its the most common form of cancer in the United States. With there being three main varieties of skin cancer, basal cell carcinoma, squamous cell carcinoma, and malignant melanoma, it can be pretty easy to develop skin cancer. Two major ways you can develop skin cancer is sunburn and basically where you live. Sunburn can cause skin cancer because it happens when you are just out in the sun to long and that means you are damaging skin cells, so your body needs to replace it which is additional blood flow, hense the skin turning red. However if a sunburn is severe enough it is less likely that the damaged cells will be removed which causes the cancer. Where you live can cause skin cancer because the closer you are to the equator the greater risk you have. That is due to the strength of the uvb radiation, because the more sunlight you receive the stronger it gets. The risks of exposure are increased greatly. Overall the difference with skin cancer is that you can't really help it, it can happen naturally, the only thing that can control it is the sun and you can't control that.

D.2.3 Example 3

You can get skin cancer from excessive UVb exposure. You can also get skin cancer if you have certain skin disorders.

Appendix E

Word Tagging Performance By Algorithm and Code

E.1 Coral Bleaching

Table E.1: Test Data Metrics for the Window-Based Tagger on the Coral Bleaching Dataset

Code	F_1	**Recall**	**Precision**
1	0.826	0.796	0.858
2	0.740	0.687	0.802
3	0.827	0.781	0.879
4	0.832	0.819	0.845
5	0.449	0.311	0.805
5b	0.030	0.023	0.045
6	0.836	0.802	0.873
7	0.838	0.760	0.934
11	0.899	0.877	0.923
12	0.863	0.780	0.966
13	0.734	0.692	0.781
14	0.748	0.744	0.753
50	0.904	0.880	0.929
Micro	0.842	0.802	0.885
Macro	0.740	0.689	0.800

Table E.2: Test Data Metrics for the CRF on the Coral Bleaching Dataset

Code	F_1	**Recall**	**Precision**
1	0.836	0.801	0.876
2	0.762	0.741	0.784
3	0.820	0.774	0.871
4	0.793	0.792	0.795
5	0.319	0.208	0.688
5b	0.000	0.000	0.000
6	0.834	0.779	0.896
7	0.826	0.739	0.936
11	0.927	0.931	0.922
12	0.863	0.780	0.966
13	0.729	0.679	0.787
14	0.700	0.684	0.718
50	0.900	0.884	0.917
Micro	0.835	0.797	0.878
Macro	0.725	0.676	0.781

Table E.3: Test Data Metrics for the HMM on the Coral Bleaching Dataset

Code	F_1	**Recall**	**Precision**
1	0.784	0.852	0.726
2	0.441	0.844	0.299
3	0.733	0.728	0.738
4	0.708	0.765	0.658
5	0.152	0.245	0.111
5b	0.073	0.295	0.042
6	0.821	0.784	0.861
7	0.731	0.653	0.830
11	0.869	0.912	0.830
12	0.698	0.807	0.615
13	0.702	0.794	0.630
14	0.715	0.877	0.603
50	0.871	0.864	0.878
Micro	0.747	0.799	0.702
Macro	0.657	0.725	0.602

Table E.4: Test Data Metrics for the Structured Perceptron on the Coral Bleaching Dataset

Code	F_1	**Recall**	**Precision**
1	0.839	0.819	0.860
2	0.783	0.735	0.837
3	0.808	0.751	0.876
4	0.827	0.825	0.828
5	0.446	0.330	0.686
5b	0.031	0.023	0.050
6	0.836	0.802	0.873
7	0.820	0.725	0.943
11	0.893	0.882	0.905
12	0.882	0.826	0.947
13	0.717	0.654	0.792
14	0.731	0.734	0.727
50	0.901	0.871	0.934
Micro	0.837	0.794	0.884
Macro	0.737	0.691	0.789

Table E.5: Test Data Metrics for the Bidirectional RNN on the Coral Bleaching Dataset

Code	F_1	**Recall**	**Precision**
1	0.819	0.859	0.781
2	0.712	0.673	0.756
3	0.827	0.774	0.889
4	0.845	0.889	0.806
5	0.587	0.670	0.522
5b	0.308	0.273	0.353
6	0.833	0.784	0.888
7	0.840	0.765	0.931
11	0.909	0.902	0.915
12	0.939	0.917	0.962
13	0.753	0.740	0.766
14	0.687	0.677	0.697
50	0.908	0.903	0.912
Micro	0.842	0.830	0.855
Macro	0.769	0.756	0.783

E.2 Skin Cancer

Table E.6: Test Data Metrics for the Window-Based Tagger on the Skin Cancer Dataset

Code	F_1	Recall	Precision
1	0.826	0.789	0.867
2	0.852	0.844	0.861
3	0.835	0.818	0.852
4	0.733	0.679	0.797
5	0.852	0.834	0.870
6	0.679	0.552	0.880
11	0.621	0.485	0.865
12	0.529	0.409	0.750
50	0.836	0.825	0.847
Micro	0.814	0.779	0.853
Macro	0.761	0.693	0.843

Table E.7: Test Data Metrics for the CRF on the Skin Cancer Dataset

Code	F_1	Recall	Precision
1	0.801	0.742	0.871
2	0.835	0.843	0.826
3	0.792	0.770	0.814
4	0.731	0.667	0.809
5	0.851	0.825	0.879
6	0.706	0.587	0.886
11	0.660	0.530	0.875
12	0.529	0.409	0.750
50	0.831	0.792	0.873
Micro	0.804	0.759	0.855
Macro	0.756	0.685	0.843

Table E.8: Test Data Metrics for the HMM on the Skin Cancer Dataset

Code	F_1	**Recall**	**Precision**
1	0.712	0.799	0.642
2	0.745	0.778	0.715
3	0.725	0.844	0.635
4	0.529	0.586	0.481
5	0.802	0.816	0.788
6	0.580	0.698	0.495
11	0.607	0.561	0.661
12	0.403	0.341	0.492
50	0.641	0.677	0.609
Micro	0.675	0.731	0.628
Macro	0.644	0.678	0.613

Table E.9: Test Data Metrics for the Structured Perceptron on the Skin Cancer Dataset

Code	F_1	**Recall**	**Precision**
1	0.813	0.759	0.876
2	0.846	0.837	0.856
3	0.828	0.806	0.850
4	0.730	0.662	0.813
5	0.854	0.846	0.864
6	0.673	0.544	0.883
11	0.604	0.485	0.800
12	0.557	0.443	0.750
50	0.845	0.825	0.866
Micro	0.814	0.773	0.860
Macro	0.757	0.690	0.840

Table E.10: Test Data Metrics for the Bidirectional RNN on the Skin Cancer Dataset

Code	F_1	**Recall**	**Precision**
1	0.853	0.821	0.888
2	0.853	0.857	0.850
3	0.836	0.787	0.892
4	0.750	0.688	0.824
5	0.862	0.847	0.877
6	0.751	0.684	0.832
11	0.667	0.515	0.944
12	0.472	0.341	0.769
50	0.870	0.861	0.880
Micro	0.837	0.807	0.869
Macro	0.779	0.711	0.862

Appendix F

Causal Relation Performance by Relation and Frequency

F.1 Coral Bleaching

Causal Relation	% Words	% Sents.	Bidirectional RNN	Stacked Classifier	Shift-Reduce Parser
1→2	0.696	1.048	0.840	0.667	0.840
1→3	1.753	2.595	0.651	0.591	0.682
1→4	0.053	0.078	0.000	0.000	0.000
1→5	0.044	0.059	0.000	0.000	0.500
1→6	0.004	0.010	0.000	0.000	0.000
1→7	0.046	0.069	0.000	0.000	0.333
1→11	0.008	0.010	0.000	0.000	0.000
1→13	0.021	0.020	0.000	0.000	0.000
1→14	0.017	0.029	0.000	0.000	0.000
1→50	2.570	4.261	0.809	0.788	0.822
2→1	0.006	0.010	0.000	0.000	0.000
2→3	0.413	0.500	0.583	0.640	0.615
2→6	0.013	0.010	0.000	0.000	0.000
2→50	0.071	0.078	0.000	0.000	0.750
3→1	0.114	0.186	0.000	0.000	0.500
3→2	0.011	0.010	0.000	0.000	0.000
3→4	0.942	1.577	0.852	0.923	0.868
3→5	0.936	1.254	0.769	0.683	0.744
3→5b	0.029	0.029	0.000	0.000	0.000

Causal Relation	% Words	% Sents.	Bidirectional RNN	Stacked Classifier	Shift-Reduce Parser
3→6	0.098	0.147	0.667	0.000	0.000
3→7	0.178	0.235	0.500	0.333	0.500
3→13	0.024	0.039	0.000	0.000	0.000
3→14	0.050	0.059	0.000	0.000	0.000
3→50	2.480	3.839	0.591	0.752	0.774
4→3	0.013	0.020	0.000	0.000	0.000
4→5	0.574	0.803	0.722	0.778	0.739
4→5b	0.170	0.186	0.000	0.000	0.000
4→6	0.007	0.010	0.000	0.000	0.000
4→7	0.057	0.078	0.000	0.000	0.500
4→11	0.008	0.010	0.000	0.000	0.000
4→13	0.005	0.010	0.000	0.000	0.000
4→14	0.720	0.872	0.600	0.824	0.842
4→50	0.769	1.009	0.514	0.615	0.600
5→3	0.008	0.020	0.000	0.000	0.000
5→4	0.007	0.010	0.000	0.000	0.000
5→5b	0.296	0.372	0.286	0.000	0.286
5b→5	0.033	0.039	0.000	0.000	0.000
5→7	0.088	0.157	0.000	0.000	0.800
5b→7	0.066	0.088	0.000	0.000	0.000
5→11	0.005	0.010	0.000	0.000	0.000
5→13	0.007	0.010	0.000	0.000	0.000
5→14	0.013	0.029	0.000	0.000	0.000
5b→14	0.013	0.020	0.000	0.000	0.000
5→50	0.357	0.568	0.667	0.308	0.667
5b→50	0.267	0.353	0.000	0.000	0.250
6→5	0.007	0.010	0.000	0.000	1.000
6→5b	0.004	0.010	0.000	0.000	0.000
6→7	1.165	1.518	0.844	0.863	0.894
6→14	0.410	0.548	0.333	0.444	0.519
6→50	0.484	0.735	0.455	0.737	0.750
7→1	0.014	0.010	0.000	0.000	0.000
7→4	0.011	0.010	0.000	0.000	0.000
7→5	0.007	0.010	0.000	0.000	0.000
7→5b	0.029	0.039	0.000	0.000	0.000
7→13	0.007	0.010	0.000	0.000	0.000
7→14	0.033	0.049	0.000	0.000	0.000

Causal Relation	**% Words**	**% Sents.**	**Bidirectional RNN**	**Stacked Classifier**	**Shift-Reduce Parser**
7→50	3.318	4.780	0.798	0.788	0.783
11→3	0.027	0.059	0.000	0.000	0.000
11→4	0.014	0.020	0.000	0.000	0.500
11→6	0.017	0.029	0.000	0.000	0.000
11→11	0.001	0.010	0.000	0.000	0.000
11→12	0.527	0.862	0.778	0.778	0.824
11→13	0.724	1.058	0.595	0.737	0.791
11→14	0.104	0.186	0.400	0.000	0.545
11→50	0.404	0.715	0.522	0.720	0.750
12→5b	0.005	0.010	0.000	0.000	0.000
12→7	0.020	0.029	0.000	0.000	0.000
12→11	0.008	0.010	0.000	0.000	0.000
12→13	0.641	0.823	0.696	0.957	0.870
12→14	0.038	0.049	0.000	0.000	1.000
12→50	0.060	0.069	1.000	0.000	0.000
13→4	0.007	0.010	0.000	0.000	0.000
13→5	0.005	0.010	0.000	0.000	0.000
13→6	0.028	0.029	0.000	0.000	1.000
13→7	0.023	0.039	0.000	0.000	0.000
13→11	0.021	0.039	0.000	0.000	0.000
13→12	0.017	0.029	0.000	0.000	0.000
13→14	0.633	0.930	0.700	0.923	0.880
13→50	0.659	0.970	0.421	0.483	0.522
14→6	0.005	0.010	0.000	0.000	0.000
14→7	0.026	0.029	0.000	0.000	0.000
14→50	0.612	0.578	0.471	0.533	0.545
50→1	0.018	0.029	0.000	0.000	0.000
50→3	0.008	0.020	0.000	0.000	0.000
50→7	0.041	0.078	0.000	0.000	0.000
50→50	0.011	0.029	0.000	0.000	0.000

Table F.1: Causal Relation Classification Accuracy (micro-F_1 score) by Algorithm, Relation and Frequency

F.2 Skin Cancer

Causal Relation	% Words	% Sents.	Bidirectional RNN	Stacked Classifier	Shift-Reduce Parser
1→2	3.774	6.043	0.842	0.812	0.855
1→3	0.567	0.764	0.812	0.692	0.800
1→4	0.030	0.039	0.000	0.000	0.000
1→5	0.192	0.264	0.667	0.000	0.000
1→50	3.372	5.338	0.747	0.725	0.815
2→1	0.011	0.020	0.000	0.000	0.000
2→2	0.013	0.020	0.000	0.000	0.000
2→3	1.355	2.723	0.804	0.757	0.785
2→4	0.444	0.774	0.500	0.533	0.267
2→5	0.610	1.077	0.605	0.333	0.357
2→6	0.028	0.029	0.000	0.000	0.000
2→11	0.004	0.010	0.000	0.000	0.000
2→50	3.982	6.396	0.818	0.701	0.769
3→2	0.011	0.020	0.000	0.000	0.000
3→4	1.393	2.302	0.835	0.867	0.909
3→5	0.206	0.411	0.444	0.400	0.400
3→6	0.207	0.284	0.500	0.444	0.500
3→11	0.007	0.010	0.000	0.000	0.000
3→50	2.232	3.820	0.746	0.742	0.753
4→4	0.009	0.020	0.000	0.000	0.000
4→5	1.268	2.439	0.881	0.792	0.845
4→6	0.340	0.539	0.222	0.000	0.000
4→11	0.004	0.010	0.000	0.000	0.000
4→12	0.004	0.010	0.000	0.000	0.000
4→50	0.602	0.979	0.529	0.231	0.412
5→4	0.288	0.656	0.372	0.000	0.400
5→5	0.038	0.069	0.000	0.000	0.000
5→6	3.194	5.260	0.878	0.881	0.867
5→12	0.007	0.010	0.000	0.000	0.000
5→50	5.313	9.158	0.852	0.843	0.847
6→3	0.003	0.010	0.000	0.000	0.000
6→4	0.013	0.020	0.000	0.000	0.000
6→5	0.050	0.078	0.000	0.000	0.500
6→50	2.240	4.192	0.912	0.876	0.876

Causal Relation	**% Words**	**% Sents.**	**Bidirectional RNN**	**Stacked Classifier**	**Shift-Reduce Parser**
11→3	0.093	0.127	0.000	0.000	0.000
11→4	0.009	0.010	0.000	0.000	0.000
11→5	0.029	0.049	0.000	0.000	0.000
11→12	0.687	1.704	0.808	0.809	0.800
11→50	0.421	0.617	0.667	0.632	0.571
12→2	0.071	0.176	0.400	0.000	0.000
12→3	0.950	2.840	0.825	0.774	0.840
12→4	0.008	0.020	0.000	0.000	0.000
12→5	0.017	0.049	0.000	0.000	0.000
12→12	0.003	0.010	0.000	0.000	0.000
12→50	0.097	0.186	0.250	0.000	0.286
50→2	0.028	0.039	0.000	0.000	0.000
50→3	0.014	0.020	0.000	0.000	0.000
50→4	0.007	0.010	0.000	0.000	0.000
50→5	0.008	0.020	0.000	0.000	0.000

Table F.2: Causal Relation Classification Accuracy (micro-F_1 score) by Algorithm, Relation and Frequency

Appendix G

The Shift-Reduce Dependency Parser Algorithm

A shift-reduce parser is an efficient approach to table-driven bottom-up parsing that is used to parse formal languages such as programming languages, in addition to its use in natural language parsing. Parsers operate on an input stream of tokens, transforming them into a parse tree. In the case of natural-language parsers, the input stream usually consists of a sequence of words forming a sentence. Shift-reduce parsers construct the parse tree incrementally, bottom-up and left to right without backtracking, by making a combination of *shift* and *reduce* steps, while manipulating a stack. The stack represents the current state of the parser at any step in the current parse, and can hold one or more partially complete parse trees that are eventually combined into one final parse tree. The *shift* step advances the input stream by one token, and creates a new single-node parse tree out of the shifted token and pushes it onto the top of the stack. A *reduce* step then applies some grammar rule to some of the parse trees in the stack, joining them together into a single tree with a new root node. To parse a sentence, the parser applies shift and reduce steps as needed until all of the input sequence has been consumed and all partial parse trees on the stack have been combined into a single parse tree.

In the following Appendix, I describe the implementation details of the causal relation parsing model, which adapts a shift-reduce dependency parser to the task of parsing causal relations between pre-identified concept codes. The parser itself uses a modified arc-eager transition system, the SEARN algorithm to train the parser, and a dynamic oracle to evaluate parse states not observed directly in the training data. The shift-reduce dependency parsing algorithm is described in detail in the following sections.

G.1 The Arc-Eager Transition System

Natural language dependency parsers are typically *transition-based parsers*. This means that they are defined in terms of a transition system - an abstract state machine that consists of a set of rules governing how the system transitions from one state to another. A number of different transition systems have been developed to adapt shift-reduce parsing to the task of natural-language dependency parsing, including Nivre's *arc standard* and *arc-eager* systems [154, 155]. The arc-eager system was adapted for this model because it makes a number of improvements over the arc-standard system. In the arc-standard transition system, a dependency arc can only be created between two nodes if the dependent node has all of its dependents, which means it is often necessary to delay attaching some right-dependents. In contrast, the arc-eager system adds an arc as soon as possible, allowing it more flexibility to build parts of the dependency tree in a top-down fashion [155].

The traditional *arc-eager* system can be defined as follows, based on the description in [155]. The arc-eager transition system constructs a labeled dependency tree $D = (W, A)$, where W is a string of words $W = w_i \dots w_n$, A is a set of arcs $(w_i, w_j)(w_i, w_j \in W)$ and w_i precedes w_j in the string W (i.e. $i < j$). If word w_i is currently at the top of the stack, and word w_j is the next token in the input stream, then the parser chooses between four different transitions to process the next token:

1. The **Left-Arc** transition creates an arc $w_j \rightarrow w_i$ from the next input token w_j to the token at the top of the stack w_i, and pops this token off the top of the stack.

2. The **Right-Arc** transition creates an arc $w_i \rightarrow w_j$ from the token w_i at the top of the stack to the next input token w_j, and pushes w_j on to the top of the stack.

3. The **Reduce** transition pops the stack, removing the top token w_i.

4. The **Shift** transition pushes the next input token w_i on to the top of the stack.

G.2 The Modified Arc-Eager Transition System

In a traditional dependency parser, the parser operates on a sequence of words and turns them into a dependency tree. However, when parsing causal relations we have a sequence of concept codes, each of which can span multiple words. In addition, not all of the codes will always be linked into a single tree structure as some causal relations may not be connected to the others, and some concept codes do not form part of a causal relation. In order to adapt the arc-eager system to parse causal relations, two modifications were made to the standard arc-eager transition system. First, the parser operates only on a sequence of concept codes; if multiple words are assigned to one concept code, it will be represented as a single token in the input. Second, to ignore codes that do not form part of a causal relation, a fifth *Skip* transition was created. The *Skip* transition discards the current input token w_j and advances the input by one token. Note that now W now represents the sequence of concept codes in the sentence, and not the set of words.

More formally, a parser configuration can be represented by the triple $\langle S, I, A\rangle$, where S is the stack, I is the list of remaining input tokens, and A is the current set of arc relations for the dependency tree. Starting with an input string W, the parser is initialized with an empty stack - $\langle nil, W, \emptyset\rangle$, and terminates when it reaches a configuration $\langle S, nil, A\rangle$ (for any list S and set of arcs A), when all of the input has been consumed. The arc-eager system then be defined by the states and transitions given in figure G.1. The first two lines in the figure represent initialization and termination states, while the remaining four lines represent the four different transition states mentioned above. The $\rightarrow$ symbols in the digram represent the transition from one parser configuration on the left to the configuration to the right of the arrow. The expressions to the right of the first three transitions represent the necessary conditions that need to first be satisfied for that transition to take place.

Figure G.1: Arc-Eager Transition System, from [155]

Initialization	$\langle nil, W, \emptyset\rangle$	
Termination	$\langle S, nil, A\rangle$	
Left-Arc	$\langle w_i\vert S, w_j\vert I, A\rangle \to \langle S, w_j\vert I, A \cup \{(w_j, w_i)\}\rangle$	$\neg\exists w_k (w_k, w_i) \in A$
Right-Arc	$\langle w_i\vert S, w_j\vert I, A\rangle \to \langle w_j\vert w_i\vert S, I, A \cup \{(w_i, w_j)\}\rangle$	$\neg\exists w_k (w_k, w_j) \in A$
Reduce	$\langle w_i\vert S, I, A\rangle \to \langle S, I, A\rangle$	$\exists w_j (w_j, w_i) \in A$
Shift	$\langle S, w_i\vert I, A\rangle \to \langle w_i\vert S, I, A\rangle$	

In order to adapt the arc-eager system to parse causal relations, two modifications were made to the standard arc-eager transition system. First, the parser operates only on a sequence of concept codes; if multiple words are assigned to one concept code, it will be represented as a single token in the input. Second, to ignore codes that do not form part of a causal relation, a fifth *Skip* transition was created. The *Skip* transition discards the current input token w_j and advances the input by one token. This gives the modified transition system in figure G.2.

Figure G.2: Modified Arc-Eager Transition System for Parsing Causal Relations

Initialization	$\langle nil, W, \emptyset\rangle$	
Termination	$\langle S, nil, A\rangle$	
Left-Arc	$\langle w_i\vert S, w_j\vert I, A\rangle \to \langle S, w_j\vert I, A \cup \{(w_j, w_i)\}\rangle$	$\neg\exists w_k (w_k, w_i) \in A$
Right-Arc	$\langle w_i\vert S, w_j\vert I, A\rangle \to \langle w_j\vert w_i\vert S, I, A \cup \{(w_i, w_j)\}\rangle$	$\neg\exists w_k (w_k, w_j) \in A$
Reduce	$\langle w_i\vert S, I, A\rangle \to \langle S, I, A\rangle$	$\exists w_j (w_j, w_i) \in A$
Shift	$\langle S, w_i\vert I, A\rangle \to \langle w_i\vert S, I, A\rangle$	
Skip	$\langle S, w_j\vert I, A\rangle \to \langle S, I, A\rangle$	

One assumption of dependency parsers is that all of the dependencies are projective [155], i.e. the arcs between the nodes don't cross one another. Traditional dependency parsers cannot therefore parse non-projective dependencies, and this will also be a limitation of the causal relation parser. Section

G.4 of this Appendix will illustrate how the parser attempts to parse a pair of non-projective dependencies.

G.3 The Dynamic Oracle

To train a dependency parser, an *oracle* is used to derive optimal transition sequences from gold parse trees. These transition sequences can then be used to train a machine learning classifier to approximate the oracle at parsing time [84]. In traditional natural language parsing, there can be multiple gold parse trees for a given sentence. However, in these 2 datasets there is only one golden parse for any given sentence because this represents the set of causal relations for that sentence.

Initial work on training dependency parsers focused on *static oracles* that produce a single canonical sequence of transitions based on a golden parse tree. Often when parsing sentences not present in the training data, the parser makes mistakes or sub-optimal decisions that leads it to deviate from the golden parse. The parser subsequently encounters configurations that are not present in the training data generated by a static oracle. In some of these configurations, the golden parse is no longer reachable. The static oracle does not provide training examples that can teach the model how to handle these configurations or recover from earlier errors. It would then be preferable for the parser to explore non-gold configurations at training time, and learn how best to handle these situations. Another limitation of this approach is in handling *spurious ambiguity*, where the same golden parse tree is can be potentially produced by multiple derivation sequences. Some derivation sequences are easier to learn than others, and the canonical sequence produced by a static oracle may not be the easiest sequence for a supervised parsing model to learn [84].

In 2012, Goldberg and Nivre presented the idea of a *dynamic oracle* to address the limitations of a static oracle [84]. A dynamic oracle is capable of determining the optimal parse decision provided any parser configuration, including those that deviate from the golden parse tree. In configurations where the golden tree is no longer accessible, the dynamic oracle returns the parser derivation that produces the parse tree with the minimum loss when compared to the golden tree. A dynamic oracle also permits all transitions that can lead to the golden tree and does not force a single canonical derivation sequence.

Designing a dynamic oracle for causal relation extraction is simpler than designing one for dependency parsing; the optimal parsing decision is the one that parses all remaining causal relations that exist between the concept codes that remain in the input stream W and the stack S. The binary relations in a dependency parse have a direction; they consist of a head word and its dependents. Each word has exactly one single head word (or has the special $ROOT$ symbol as the head), but a head word can have multiple dependents. For the causal relations, a causal concept can have multiple effects, and an effect can have multiple causes. I am therefore unable to use the direction of causality in place of the head-dependent relation because there is not always a single cause or a single effect for each concept code in a causal relation. To address this problem, the parser is only responsible for determining when a causal relation exists between 2 concept codes, and a separate binary classifier then decides which code is the cause and which is the effect.

I define the dynamic oracle in Algorithm G.1, where C is the set of remaining causal relation tuples in the sentence (that aren't currently in A), S represents the stack, and w_i and w_j represent the top of the stack and the next input token as before. A causal relation is defined as a tuple of two concept codes (w_i, w_j) where $(w_i, wj \in W)$. The tokens returned (*LEFT-ARC*, *RIGHT-ARC*, etc) represent the optimal parse decision to make given the current parser configuration, which is dependent on w_i,w_j and S, and the set of unparsed causal relations C. C is maintained by the oracle throughout the parse, and updated as relations are parsed.

Algorithm G.1 works as follows. On lines 2 and 9 of the algorithm, the oracle checks to see if there is a causal relation between the code at the top of the stack w_i and the next input token w_j in C (i.e. that is not currently in A). If there is, then the parser needs to create an arc between these two codes. *LEFT-ARC* creates an arc between w_i and w_j and then discards w_i, while *RIGHT-ARC* creates an arc between w_j and w_i and pushes w_j onto the stack for use later (see Figure G.2). If there are no other causal relations including w_i (lines 4 and 11), then *LEFT-ARC* is invoked, and w_i is discarded, otherwise RIGHT-ARC is invoked, keeping w_i and pushing w_j onto the stack. In either case, C is updated to remove the detected causal relation (lines 3 and 10), so that it maintains the set of the unparsed relations. If there is no causal relation between w_i and w_j but a causal relation does exist between w_j and some other token w_k in the stack S (line 16), then the *REDUCE* command is invoked, popping the stack so that other causal relations can be considered between the items on the stack and w_j. If one or more causal relations exist in C that include w_j (line 18), then *SHIFT* is invoked, pushing w_j on to the stack for use later. Finally, if line 18 is false (i.e. w_j is not involved in any remaining causal relations), the *SKIP* operation is invoked, discarding the token and advancing the input.

Algorithm G.1 The Dynamic Oracle Algorithm

```
function DynamicOracle($w_i$,$w_j$,$S$,$C$)
    if $(w_i, w_j) \in C$ then
        $C \leftarrow C \setminus (w_i, w_j)$
        if $\neg\exists w_k (w_k, w_i) \in C$ then
            return LEFT-ARC
        else
            return RIGHT-ARC
        end if
    else if $(w_j, w_i) \in C$ then
        $C \leftarrow C \setminus (w_j, w_i)$
        if $\neg\exists w_k (w_k, w_i) \in C$ then
            return LEFT-ARC
        else
            return RIGHT-ARC
        end if
    else if $\exists w_k (w_k, w_j) \in C \wedge w_k \in S$ then
        return REDUCE
    else if $\exists w_k (w_k, w_j) \in C$ then
        return SHIFT
    else
        return SKIP
    end if
end function
```

The dynamic oracle always makes the optimal parsing decision given the parser's current configuration and the set of unparsed causal relations. To prove this, two conditions must be true:

1. The oracle must not make any parsing decisions that prevent any causal relations from being parsed between concepts in the stack S and in the remaining input stream W.

2. All remaining causal relations present in a sentence must be parsed.

For condition 1 to be true, the parser must never discard the token w_i at the top of the stack or the current input token w_j if they form part of a relation in C. From Figure G.2, the only parse decisions that discard w_i are *LEFT-ARC* and *REDUCE*. From lines 4 and 11 in Algorithm G.1, we see that *RIGHT-ARC* will be invoked instead of *LEFT-ARC* if w_i has a causal relation with some other concept, and because of the preceding if clauses, *REDUCE* will only be invoked if there is no relation between w_i and w_j, preventing this token from being discarded incorrectly. However, it is possible that *REDUCE* could be invoked when there is a causal relation between w_i and some other concept code in W, aside from w_j. However, this only arises when w_i forms part of a non-projective relation, which is not handled correctly by our parser as discussed in Section 5.4.3.1. How the parser handles non-projective relations will be discussed in the next section. Finally, the only parsing decision that discards the next input token w_j without pushing it onto the stack is the *SKIP* decision. This will only be invoked when no more relations exist in C that include that concept code (see line 18). Therefore, provided all causal relations are non-projective, the oracle will never discard concept codes that form causal relations.

It is also clear that the oracle will always detect all projective relations, satisfying condition 2. When the oracle encounters the first concept code in a causal relation, it will be pushed onto the stack by the *SHIFT* operation (due to line 18) if it does not form a relation with an existing item on the stack. If it does form a relation with another item on the stack, that item will be popped to the the top of the stack by the *REDUCE* action (lines 16 and 17) if it is not already at the top. Then w_j will either be pushed on to the stack by *RIGHT-ARC*, or by a *LEFT-ARC* followed by a *SHIFT* action. All tokens pushed onto the stack will then eventually be parsed into a causal relation with the other concept code in the relation when it becomes the next input token w_j. This is because of line 16, which invokes the *REDUCE* action, popping the stack until the top of the stack token w_i forms a causal relation with w_j by *LEFT-ARC* or *RIGHT-ARC*.

G.4 The Parser and Oracle in Action

The example in Figure G.3 shows an example of a sentence containing 2 projective causal relations, and Table G.1 shows a trace of how the oracle and shift-reduce parser parse this sentence. There are five concept codes in the sentence, a, b, c, d and e, with two causal relations, one between a and b and a second between d and e. The parser is initialized with a special token *Root* on the stack. The oracle then invokes the *SHIFT* action pushing the a token on top of the stack because it forms part of a causal relation but not with the *Root* symbol. Then *LEFT-ARC* is invoked because $w_i = a$ and $w_j = b$ and no other relations exist involving the a code (thus *RIGHT-ARC* is not invoked). The new relation $a \rightarrow b$ is then added to the set of arcs A, and removed from the set of remaining relations C. The oracle then invokes *SKIP* twice, discarding the b and c codes because they are not involved in any remaining causal relations. The oracle then invokes *SHIFT* to push d onto the top of the stack, and then *LEFT-ARC* to form a relation with e. As before, the token d is discarded by *LEFT-ARC*, and the new relation $d \rightarrow e$ is removed from C and added to A. Finally *SKIP* discards the last remaining input token e. Note that a second classifier is invoked after each arc is created to determine the direction of causality, which in this case would invert the last relation to be $e \rightarrow d$.

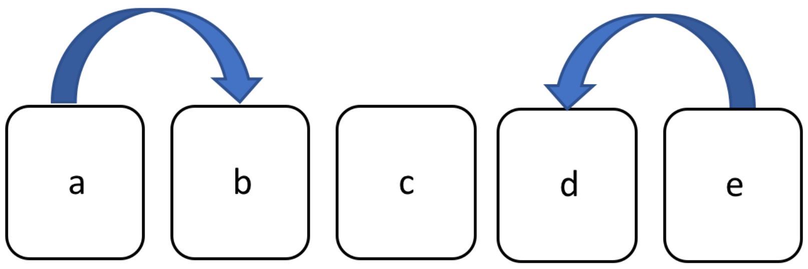


Figure G.3: Parser Example - Projective Causal Relations

Table G.1: Parser Trace for Projective Causal Relations Example

Step	Action	Stack (S)	Input (W)	Arcs (A)	Remaining Causal Relations (C)
0		Root	a|b|c|d|e		a→b,d→e
1	SHIFT	Root|a	b|c|d|e		a→b,d→e
2	LEFT-ARC	Root	b|c|d|e	a→b	d→e
3	SKIP	Root	c|d|e	a→b	d→e
4	SKIP	Root	d|e	a→b	d→e
5	SHIFT	Root|d	e	a→b	d→e
6	LEFT-ARC	Root	e	a→b,d→e	
7	SKIP	Root		a→b,d→e	

A limitation of traditional dependency parsers is that they can only parse projective dependencies [157]. If non-projective dependencies exist in the sentence, some of them will fail to be parsed. Figure G.4 below shows an example of a sentence containing two non-projective dependencies, while Table G.2 shows the parser trace illustrating how the parser would attempt to parser the sentence, failing to parse the second dependency arc: b→d.

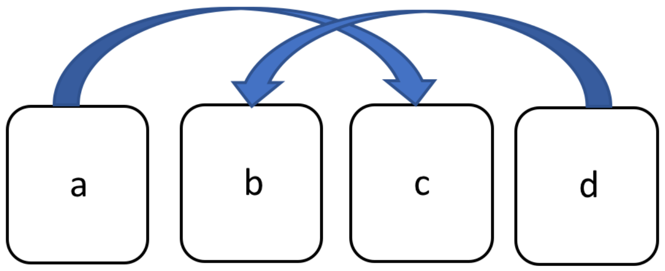


Figure G.4: Parser Example - Non-Projective Causal Relations

Table G.2: Parser Trace for Non-Projective Causal Relations

Step	**Action**	**Stack (S)**	**Input (W)**	**Arcs (A)**	**Remaining Causal Relations (C)**
0		Root	a|b|c|d		a→c,b→d
1	SHIFT	Root|a	b|c|d		a→c,b→d
2	SHIFT	Root|a|b	c|d		a→c,b→d
3	REDUCE	Root|a	c|d		a→c,b→d
4	LEFT-ARC	Root	c|d	a→c	b→d
5	SKIP	Root	d	a→c	b→d
6	SHIFT	Root|d		a→c	b→d

I have described how a dynamic oracle can produce the optimal parse for any sentence containing non-projective causal relations. In the next section, I will describe how such a parser can learn to imitate the dynamic oracle using machine learning techniques.

G.5 Imitation Learning: Applying Reinforcement Learning to Train a Shift-Reduce Parser

As described in Section 3.4.8, reinforcement learning is a form of machine learning where an agent learns by interacting with its environment, receiving a reward or punishment for each action based on how the action impacted its goals. Imitation learning then adapts reinforcement learning to solve supervised learning problems. Reinforcement learning is a good fit for solving structured learning problems because structured prediction usually involves making a sequence of decisions before a final prediction is made. Usually the loss associated with a particular decision is not known, only the loss associated with the final prediction, once it has been made. This results in a credit assignment problem; it is not clear which of the decisions contributed most to the success or failure of the final prediction. This is the exact problem reinforcement learning attempts to solve.

SEARN is an imitation learning algorithm used to solve structured prediction problems. SEARN tackles structured prediction by decomposing each structured prediction problem into a sequence of cost-sensitive classification decisions, and then trains a set of cost-sensitive classification algorithms to handle each of these decisions [43]. SEARN does not make use of a novel machine learning algorithm, instead it can use any cost-sensitive classification algorithm. For each decision made on the training data, a new cost-sensitive classification example is created for each possible action that can be taken from that decision. To create these examples, a cost function is required that estimates the relative cost of making each possible decision. These examples are then used to train the set of cost-sensitive classifiers before the next iteration. SEARN and imitation learning are described in more detail in Section 3.4.8.

In order to address Research Question 2, the SEARN algorithm was used to train the causal relation parser described in the previous sections. To achieve this, a cost function was defined to compute the cost of invoking each action from a particular parse state and was designed to estimate the impact on the micro-F_1 score of taking each action. Micro-precision and micro-recall are computed by calculating the false positives, false negatives and true positives for all classes across all data points, then the micro-F_1 score is calculated as the harmonic mean of the micro-precision and micro-recall (see Section 4.1). For a parse action a_i, the cost of the action, $cost(a_i)$, is defined as the increase in the number of false positive and false negative causal relations that were parsed when compared to the optimal action a_{opt} supplied by the oracle. This reflects the impact that action will have on reducing the final micro-F_1 score. Given the set R_{opt} of relations parsed following invocation of the optimal action a_{opt}, and the set R_i of relations parsed following invocation of action a_i, then the increase in false positives is the number of parsed relations that would otherwise not have been parsed by following the optimal action: $fp = |R_i \setminus R_{opt}|$. Similarly, the increase in false negatives is the number of relations in the optimal parse that were not parsed by following action a_i: $fn = |R_{opt} \setminus R_i|$. The cost for the optimal action a_{opt} is the number of true positives, which is defined as the size of R_{opt}. This number is negative because this action has a negative associated cost. Equation G.1 below describes the cost function in full.

$$cost(a_i) = \begin{cases} -\left|R_i\right|, & \text{if } a_i = a_{opt} \\ \left|R_i \setminus R_{opt}\right| + \left|R_{opt} \setminus R_i\right|, & \text{otherwise} \end{cases} \tag{G.1}$$

In this cost function, the costs are computed relative to the optimal parse given the current parse state (as determined by the oracle) in place of the golden parse from the training data labels. This is because in the middle of a parse, a number of mistakes may have already been made that cannot be corrected, and I do not want to include those mistakes when computing the costs. Instead I compute costs only relative to the optimal achievable parse given the current parse state. The cost function was then used to create the cost-sensitive training examples used to train the machine learning algorithm for each iteration of SEARN. A logistic regression classifier from the *LibLinear* package [65] package was chosen to train the SEARN models because it is capable of cost-sensitive learning, robust to overfitting, and scales well. Using Logistic Regression also enabled the use of the same model for feature selection and model training because it scales well to datasets with large amounts of features.