Learning Negation Scope Semantics with Structure

Nicholas McKenna

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2019

Abstract

In natural language, negation is a semantic operation that inverts meaning and is triggered by a word or affix. The process of determining which words, and thus which parts of meaning, are affected by the trigger is called Negation Scope Detection (NSD). This project begins with a critical review of methods in NSD and presents a new approach to the task based on this research. It is hypothesized that negation scope may be determined purely through syntactic reasoning, and the methods developed test this theory. A Tree Recursive Neural Network, which uses only syntactic reasoning and does not identify words in a sentence, is shown to successfully detect negation scope with performance very close to state-of-the-art. This model confirms the project hypothesis and is further analyzed to better understand how it processes information.

Acknowledgements

Thank you to my supervisor, Professor Mark Steedman, for all your support, constructive criticism, seemingly unending knowledge, and most of all for believing in me.

Thank you to my family for supporting me all the way here, and for cheering me on every day. My accomplishments mean so much more when I share them with you.

Thank you to my friends, both those in the UK and those many time zones away. You enlighten me, delight me, and make me appreciate everything about the journey.

Finally, thank you to the city of Edinburgh which changes me every time I come back. You are a special place to me.

Table of Contents

Chapter 1

Introduction

1.1 Introduction to Negation and Scope Detection

In linguistics and formal logic, negation is an important semantic operator which inverts the meaning of an expression [16]. Properly understanding the impact of negation cues in language is thus vital for understanding an utterance, both for humans and algorithms. For example, imagine a software agent which makes rapid stock trading decisions by reading online news articles. It might make drastically different decisions if it encounters the sentence "British imports will be severely impacted" vs. "British imports will *not* be severely impacted." The simple word "not" critically shifts the meaning of what is expressed and has downstream consequences for decision making. Understanding negation is broadly important for many automated tasks beyond this, like web search, conversing with personal assistants, machine translation, and others.

Negation Scope Detection is typically a two-stage process for identifying which words of a sentence are affected by a negation cue. In the first stage, a negation cue is detected in the text, such as the words "not" and "without," or affixes like "un-". In the second stage, spans of text forming the negation scope are identified and judged "in scope" of the cue. This is most easily done by producing individual binary label judgments for each word of the sentence. This second stage is the focus of this project, as it is the most interesting part. The rules for determining scope can differ depending on parsing goals, but one useful method seeks to identify affected semantic events and related arguments. Such a goal formulation requires an automated system to learn the semantics of how these elements should be selected by leveraging other sources of information. Several approaches are explored in the Background chapter and an argument is made for relying on syntax to identify scope.

1.2 Research Hypothesis and Objectives

The Background chapter describes a history of approaching this task as a sequence tagging problem, whereby input sentences are considered as a linear sequence of words and scope judgments are made in this context. These techniques occasionally make use of syntactic features in the decision process, but infrequently make use of full and explicit syntactic structure despite theoretical evidence of syntactic ties [36, 39].

This project poses a research question to investigate this disparity. Can negation scope semantics be learned from explicit sentence structure, in terms of larger phrase constituencies? Further, negation scopes share common traits in their structural composition. Can scope be identified solely by inference from sentence structure?

The objective of this project is to reframe NSD from a sequence tagging problem to a tree tagging problem, leveraging explicit syntactic parse trees to make scope decisions about constituents at the surface text and also larger constituent units in the tree. A successful machine learner developed for this task should thus learn scope semantics for every syntactic composition and produce accurate predictions in the surface text.

1.3 Key Conclusions of this Report

Evaluating the experiments carried out in this report shows that explicit syntactic processing does provide an excellent means for detecting scope. The model developed here achieves near-state of the art performance on the *SEM2012 dataset, and does so without the use of a vocabulary or any other word features as do most successful published methods. This method relies solely on syntactic parse trees and a Tree Recursive Neural Network to process them compositionally, demonstrating that explicit syntactic processing alone can effectively model negation scope determination in a sentence.

1.4 Contribution of this Project

The major products of this project are a new understanding of negation scope detection using explicit syntactic processing, and a model demonstrating this. This result deviates from common methodologies on the task which frequently undervalue the contribution of syntax and combine many different features to compute scope judgments. This method uses only syntax and performs at nearly the state of the art.

Chapter 2

Background

Negation is a tool we use in language without even thinking about it. It appears to be superficially straightforward to understand how negation operators affect the meaning of our utterances. However, exploring deeper reveals that it can quickly become complicated. This chapter has two purposes: to describe some of the linguistic and technical phenomena that relate to negation scope, and to analyze the history of computational methods developed to solve the task of Negation Scope Detection (NSD). This background serves to motivate both the project hypothesis and methods.

It is important to first introduce the major NSD resources and to formally define negation scope, which will help frame the rest of the background review. NSD research became popularized with the releases of the Genia and BioScope medical corpora [19, 41]. These provided resources for many developments but are centered on the medical domain (and writing style), and until the *SEM2012 Shared Task competition in NSD there were no general-domain resources. Additionally, *SEM updated the guidelines for determining negation scope from the mostly syntactic annotation guidelines of BioScope [22] to a "semantic" definition of the task [1, 2]. *SEM defines NSD: "the negation cue will be such that it allows to determine which events are negated in the sentence," with scope that also encompasses semantic arguments and complements of those events. They provide a detailed handbook of these rules [35]. This project approaches NSD as defined by *SEM with analysis based on this goal.

2.1 Understanding Negation

2.1.1 Negation by Example

Negation scope is best understood by example. This section walks through several sample sentences from the *SEM dataset of Conan Doyle writing [1] to introduce concepts and nuances in negation, and set the context for review of existing methods in NSD. In these sentences, negation triggers are written in **boldface** and starred* while their resulting scopes are underlined. As per *SEM convention, a trigger is never contained in its own scope. A simple example is useful to start.

(a) Well, Sir, I thought **no*** good could come of it.

Sentence (a) contains two events: the first which describes a "thinking" event, and the second a "coming" event which is negated. It is clear from this example that sentences may reason about multiple events at the same time, sometimes using structures like control [8], so it is important to carefully recognize which of them are negated. While the "coming" event of (a) and its arguments are negated through the determiner "no" on the subject "good," there are many ways to trigger negation. Some of these are *verbal* like "not" directly applied to a verb, *affixal* as in the use of morphological affixes "un-" and "-less," and also by subject/object negation using words like "nobody" [15, 17]. In most instances a cue licenses a negation scope encompassing other words, and there are many examples of more complex cases [14].

The *SEM dataset provides a syntactic constituency parse tree (using a Context Free Grammar) of each sentence. Figure 2.1 shows the tree for (a), which demonstrates how scope can align cleanly with a syntactic subtree.

It appears that syntax has a strong role in determining scope. As Giannakidou [14] describes, there can be an appearance that negation operators take scope over lexical items within their c-command, which is essentially the set of a node's children, parent, and sibling subtrees [36]. However, this is not strictly true as example (a) itself includes a broader scope than predicted just by c-command. Further, the following example shows a case when scope is blocked from a subtree within the cue's c-command.

(b) I **never*** hurt man or woman in my life that I know of.

Figure 2.2 shows the parse tree for (b). Scope is blocked from encompassing the subordinate clause "that I know of." From these examples emerge a pattern about the organization of scope. It appears that scope is identifiable in larger grammatical units of phrases and clauses. Further examples develop this idea.

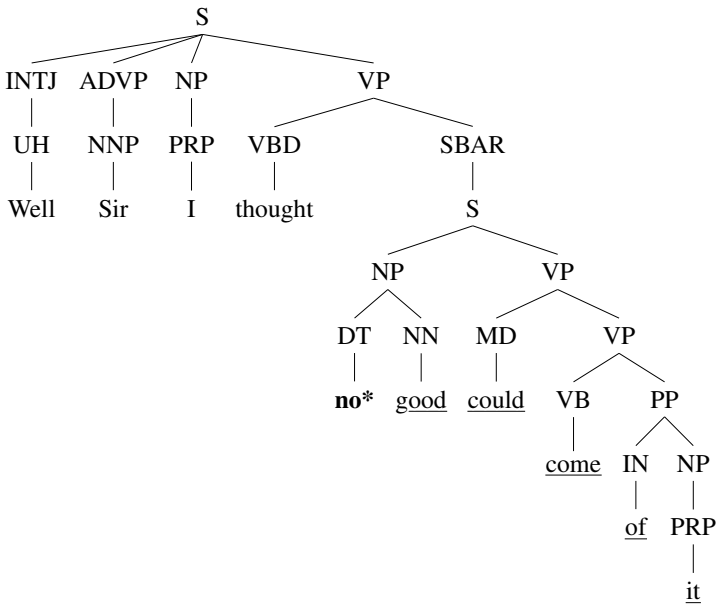


Figure 2.1: Syntactic parse of example sentence (a).

The *SEM definition of negation scope is a semantic one, meaning that it covers all semantic objects in a negated event [27]. Scope discontinuity is thus possible if these semantic objects are not located adjacently in the sentence, as in the following example.

(c) He saw him once or twice but he is a deep one and gives **nothing*** away.

Sentence (c) (parse tree shown in Figure 2.3) has a cue which licenses a discontinuous scope. "He" is the subject of "gives" and thus falls in its scope, despite the fact that "he" is a distant token in the sentence. This case also demonstrates how scope interacts with a VP coordination. One verb phrase, "gives nothing away," can be in scope while the other verb phrase, "is a deep one," remains out of scope because they describe two different events.

Negation can apply to many different elements of a sentence. The following example shows negation of a prepositional phrase with parse tree shown in Figure 2.4.

(d) Some people **without*** possessing genius have a remarkable power of stimulating it.

This example is useful to demonstrate how the prepositional phrase modifies the subject "some people" of the main verb "have." Notably, the negation scope only covers the subject, because "have" is understood to be positively polarized.

Finally, it's interesting to consider that negation scope does not always need to be assessed in "one-way" from triggers to tokens. Some specific words like "any" and

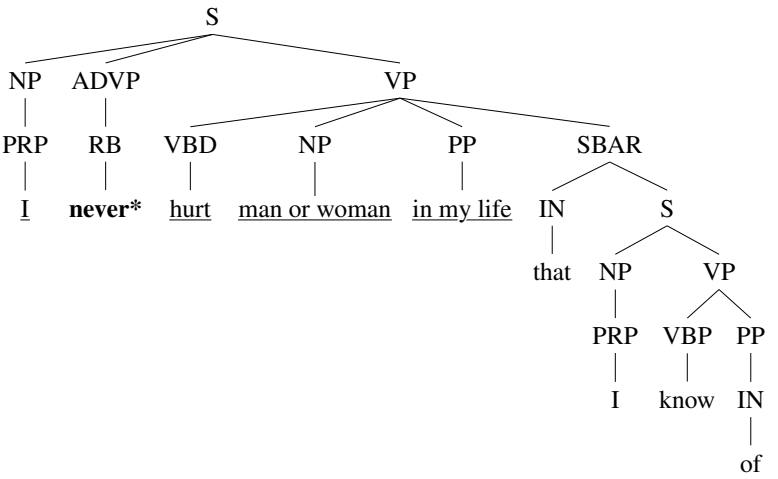


Figure 2.2: Syntactic parse of example sentence (b). Note that the subordinate clause "that I know of" is not in scope.

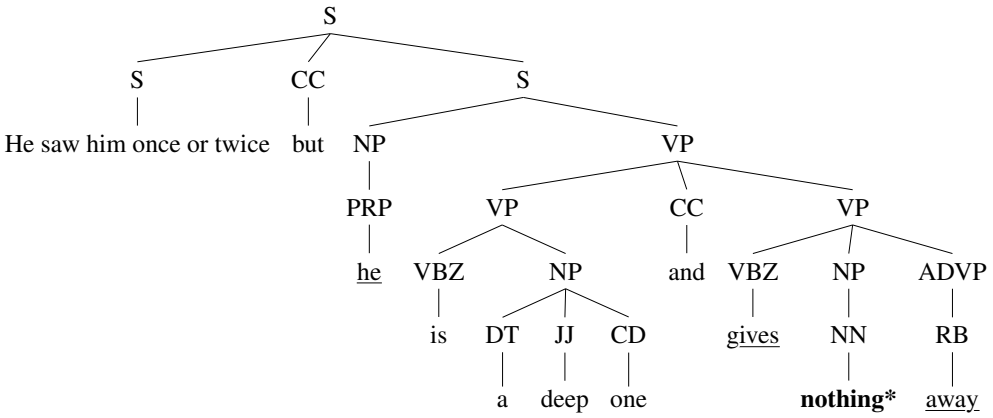


Figure 2.3: Syntactic parse of example sentence (c). Scope is discontinuous in order to cover the subject "he."

"anybody," called Negative Polarity Items (NPIs), occur only in negatively polarized contexts, questions, and conditionals [14, 15, 17]. Due to this fact, they frequently indicate the presence and extent of a negation scope. An example follows.

(e) I can **not*** say anything definite, for I do not know anything definite.

The NPI "anything" must occur within a negative context and is a strong clue to the presence of a negation cue and scope.

The examples in this section show structural correspondence with scope in many cases, but also show a variety of special cases which make it difficult to list a concise set of rules for scope determination. This presents an opportunity for machine learning methods which learn by example to generalize about this linguistic phenomenon.

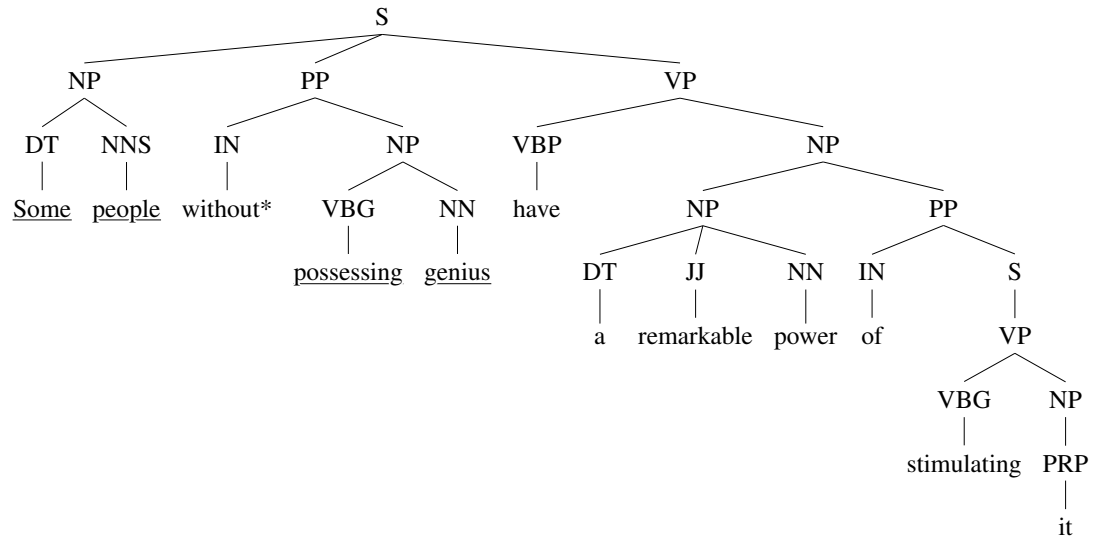


Figure 2.4: Syntactic parse of example sentence (d).

However, it is very helpful to machine learners if variability and ambiguity in the problem space are minimized, and CFG parses are sometimes inadequate. For example, CFG nonterminal nodes have an unrestricted number of children, and this variability may be difficult for a machine learning algorithm. A specific example is seen in sentence (d) (Figure 2.4), where the top level constituent *S* has three children: *NP*, *PP*, and *VP*. The prepositional phrase containing the cue, "without possessing genius," attaches to the noun phrase to its left. In addition to handling the structural variation, the algorithm must learn not to associate it with the verb phrase to its right while also learning the scoping semantics of the task. This can instead be solved using a more appropriate system of grammatical structure to reduce variation and ambiguity. Thus, turning to a different parsing formalism is motivated, which provides more optimal trees from which to learn scope semantics.

2.1.2 Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG) is a kind of phrase structure grammar which generates parse trees of constituencies, like CFG [38]. However, it is "mildly context sensitive" which allows it to capture phenomena like coordination [40], and enforces a predictable, binary tree-like structure due to its use of combinators. The CCG tagset is also much larger than a CFG tagset, and this gives CCG constituents a more specific description of their grammatical purpose. These features are highly useful and produce trees which better align with negation scope. Figure 2.5 shows a CCG parse for

example sentence (d), as compared to the CFG parse shown in Figure 2.4. With CCG the prepositional phrase has been attached to the subject "some people," more clearly demonstrating why the negation cue scopes over them both.

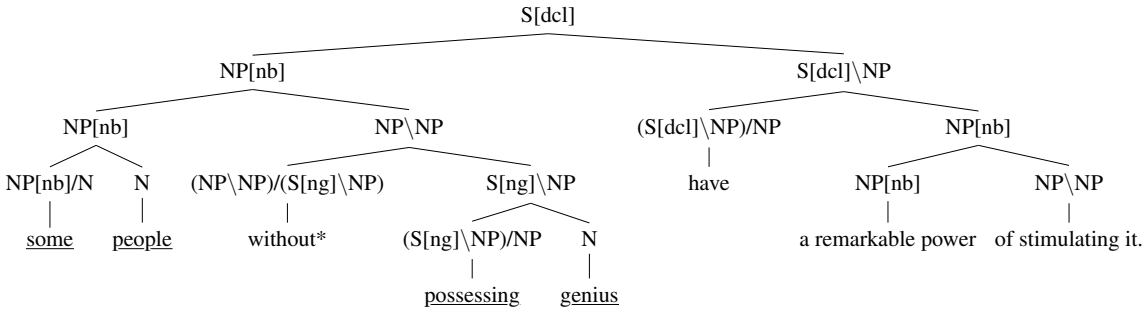


Figure 2.5: CCG syntactic parse of example sentence (d).

Another benefit of CCG is the transparency between syntax and semantics. A CCG syntactic parse may be procedurally transformed into a logical form representing the semantics. Steedman [39] describes a formal method of computing the negation polarity of lexical items using a theoretical extension to CCG semantic parsing. This method requires additional information for the category tagset about polarity preservation, but proves that this kind of calculation can produce polarity judgments at the granularity of lexical items. This is very encouraging for the project hypothesis.

For these reasons CCG parses are used throughout this project in place of the provided CFG parses from the *SEM2012 dataset.

2.2 Previous Work on Negation Scope Detection

Negation scope detection has been an important computational task for some time, gaining popularity with the release of the BioScope [41] and Genia [19] corpora, which focus on data mining in the biomedical domain. Machine learning techniques have grown considerably in the intervening time and this has lead to a variety of models and inspirations for the task. This section analyzes the major trends in NSD models and sets the context for this project. As described earlier, the work in this project is centered around the *SEM2012 Shared Task for NSD, so models will be evaluated with this dataset and task in mind.

2.2.1 Rule-Based Models

Some of the first approaches to Negation Scope Detection are based on prior linguistic knowledge. These methods tend to use hard-coded heuristics or learn basic patterns in syntax in order to make judgments of scope.

The *SEM2012 competition results [28] show that generally these rule-based models display promise on the task but are less performant than other types of methods, specifically classical machine learning models. A few of these models are worth discussing, however, since they demonstrate the learnability of NSD through syntax.

Apostolova [6] crawls syntactic parse trees and collects a set of syntactic patterns. The rules learned relate cue tokens to scope tokens using syntactic paths (a list of the constituents forming the shortest path in a parse tree between two tokens). Apostolova then manually sorts and simplifies these patterns using prior linguistic knowledge. This process makes the rules more generalizable since they become less specific to particularities of the training data like irregularities in sentence structure or even genre norms. This is very effective compared to classical machine learning methods applied to the BioScope corpus, possibly because the crawler isn't reliant on the correctness of parse trees. Test set trees need only be consistent with training tree parses for the crawler to be effective.

Rosenberg [34] developed a primarily rule-based method which is focused almost entirely on the application of prior knowledge in syntax. The model contains a comprehensive hand-written database of heuristic rules for scope detection. It was published a few years after the *SEM2012 competition, and they claim to retroactively win it. While it is a seemingly impractical model to produce or analyze without specialized knowledge, it is still impressive given that the winner of *SEM (UiO_1 [32]) used a classical machine learning method, which otherwise typically perform better than rule-based methods.

These examples hint at the predictability and thus learnability of negation scope using syntax. Rule-based methods are impractical and in many cases specific to their training domains, but do provide some direction for development of future models.

2.2.2 Classical Machine Learning Models

Many high-performing NSD models make use of classical machine learning methods. They apply linguistic knowledge with the generalizability of learning algorithms in creative ways, to good effect.

UiO_1 [32] and UiO_2 [23] (both submissions from the University of Oslo) are two top entries in the *SEM2012 competition, achieving respective F_1 scores of 85.26 and 83.73 [28] on the task of identifying individual tokens in scope of the cue. Both models use a Support Vector Machine to classify cue tokens (a separate task not in scope of this project) but use different methods to select features and detect negated scopes. UiO_1 uses a second SVM to classify the whole scope of a cue by examining the larger constituents which contain it, used as features. It additionally uses certain lexical features as well as several hard-coded heuristics stemming from prior linguistic knowledge. Instead of classifying constituents UiO_2 uses a Conditional Random Field, modeling the problem as a probabilistic process, to produce labelings for each token in the sequence. Abu-Jbara [4] also models the problem with a CRF to produce classifications for each token. They achieve results similar to UiO_2 with a *SEM scope tokens F_1 score of 83.51. However, neither of the CRF models make substantial use of syntactic tree structure in the models, so it is possible that one differentiating feature of the winning model, UiO_1, was the use of explicit syntactic reasoning. Despite this, both the constituency judging SVM and sequence-based CRF models are reasonable ways to model negation scope detection, and provide valuable insight into the problem.

Zou [44] attempts to directly address syntax using "tree kernels," which use syntactic paths compressed into linear descriptions. The tree kernels then compare pairs of subtrees (collections of syntactic paths). The model performs well on BioScope, but not across testing domains. It is possible that using syntactic data in this way created a sparse data problem, and this resulted in overfitting to the training data.

These classical machine learning models look at the problem of NSD from different angles and provide interesting insights. They show the usefulness of generalized learning algorithms while also exposing limitations and directions for future work, both in terms of modeling with syntax and making design decisions which help avoid overfitting.

2.2.3 Neural Network Models

Neural Network models combine aspects of the previous two classes of models in an interesting way. They can be flexibly designed to directly model linguistic phenomena such as what is captured in the rule-based models, but also leverage robust learning theory which additionally provide automatic and generalizable performance.

Qian's approach [31] is similar to Zou in its analysis of syntactic paths. Qian

develops a Convolutional Neural Network which learns patterns within these paths in order to predict the scope of a negation operator. This approach is very successful on the BioScope corpus, achieving state of the art performance. However, the use of a CNN in the setting of a language task is somewhat experimental and it may be risky to learn exact patterns from cue to token, which may be highly irregular depending on context, like with Zou [44]. It is possible that this may be unhelpful when attempting to deploy a model trained in the biomedical domain to some other genre, for instance.

Fancellu [10] uses a Bidirectional Long Short-Term Memory model (BiLSTM) to consume sentences as sequences of words and make per-word classifications of scope. The model is conceptually simple and agnostic to the language used because it learns all discriminating patterns from the training data automatically. It also achieves high performance compared to the other methods discussed, with a *SEM scope tokens F_1 score of 88.72 on *SEM. This score is several points higher than the winner of *SEM and represents the highest score to date. The model interprets each word as a vector of features, with additional features like part-of-speech appended. This means that sentence syntax is implicitly supplied in the representation since it's read as a list of tokens, and not read as a tree. Likely because of this, the predicted negation scope sometimes leaks out of obvious syntactic boundaries. Additionally, further analysis showed the technique was susceptible to overfitting to punctuation tokens like "," which frequently mark scope boundaries in English, without learning any deeper semantics [11]. It is also notable that in this project Fancellu uses both jointly learned word embeddings and pretrained word embeddings, neither of which provides a significant improvement over the other. This is suspicious because it might be expected that more robust word representations trained on more data would provide better results. This suggests that the model is in fact leaning on other features of the words such as the explicit PoS tags or typical usage patterns of words, both of which point to the usefulness of syntactic information.

Fancellu took insights from the BiLSTM when developing follow-up Dependency-LSTM and Graph Convolutional Network models, which leverage syntactic structure more directly. Each network consumes a dependency tree of a sentence, with words represented as a collection of features including word embeddings and dependency relation embeddings. They both process sentences similarly, as well. The D-LSTM consumes words in order by following dependency relations. It uses an upward pass which combines word representations from leaves to root, and also a downward pass decomposing from root to leaves. The upward pass shares some information with the

downward pass to add context. The GCN model similarly processes words using the dependency tree structure but viewed as a graph, simultaneously computing negation scope for all words at once using information spread in local neighborhoods. Both of these networks underperform considerably, compared to the baseline BiLSTM model on the same task (a modified version of *SEM). This is surprising given that they make architectural changes in an otherwise promising direction, emphasizing the role of sentence structure in scope resolution. It may be that either the dependency parse structure or the means of processing it were not as informative as the BiLSTM, but it is difficult to judge absolutely that adding structure is ineffective, especially because model results are not available for *SEM. These two models serve as inspiration for this project, however, and will be conceptually revisited.

Neural approaches to NSD show great improvement over classical machine learning and rule-based methods in modeling flexibility and generalization to new data. They are capable of learning representations from surface text alone without additional linguistic prior knowledge, which makes them both useful and portable between language domains. They are a clear asset for the task of NSD and open up many creative directions for future work.

2.2.4 Semantic Models

Formal logic methods have also been applied to the problem of NSD. Instead of building a semantics on top of syntactic analysis as the other methods do by identifying scope from words and word features, these methods first reformulate the problem in a purely semantic, logical domain, and then identify scope. This has both benefits and drawbacks.

Packard [29] is a strong proponent of solving NSD ("a semantic problem") with a semantic solution. They claim that linguistic operators, quantifiers, and scope are reasoned about in the logical domain and should thus be solved in this domain as well. Both Packard and Li [25] propose similar models which explore this idea. Packard first induces a semantic parse of a sentence in Minimal Recursion Semantics (MRS) [7], and then crawls the parse to look for negated predicates and arguments. This is a process of backtracking to identify which words of the sentence are in scope of the negation operator. It is able to determine most words inside the negation scope, though the semantic parsing process throws away "semantically vacuous" words like "that," which means Packard must deploy several heuristics to recover them. This is not a

major concern other than for task performance on *SEM (Packard's solution by itself does not achieve a competitive score in part because of this), but it does raise a question about this direction of research. It is interesting that the task can be solved in the logical domain, but doing so effectively offloads the major challenge of the problem onto the semantic parser. At this point is this a purely "semantic solution," since negation and other aspects of semantics are structured by the semantic parser, which leans heavily on other sources of information to do so, like syntax?

Fancellu [12] also developed a logical approach to NSD. This approach is an extension to the UDepLambda system of formal logic [33] which adds negation coverage. In doing so, the first order logic expressions produced by the system also express the scope of negation operators. However, similarly to Packard, Fancellu's approach also throws away "semantically vacuous" tokens in the source text but does not present a plan for mapping back to it. This makes it difficult to assess performance, and further the paper does not provide its own numerical results for evaluation.

The research into semantic solutions provides a fresh perspective on the problem and exposes some questions about the division of labor done to process negation semantics. Negation reasoning may happen in the logical domain, but the *scope resolution* which enables this reasoning seems to be determined during the construction of semantic forms. It seems reasonable then that while negation scope is identifiable in the semantic form, it is solvable in an earlier processing phase which makes semantic backtracking unnecessary.

2.3 Other Related Work

Paulus [30] addresses the task of sentiment analysis using a method of interest to this project. They present a new model called the Global Belief Tree Recursive Neural Network (GB-TRNN) which uses explicit syntax trees to reason about the connections between different constituents, and provides fine-grained sentiment scores for every constituent in the tree. The model first performs an upward pass which recursively combines information state vectors from leaves to root. Conditioned on the global state of the root constituent, a downward pass is then performed from root to leaves, which recursively decomposes the information state following the syntax tree. The authors report state-of-the-art results for their sentiment analysis task which is very encouraging. This mechanism is highly relevant to the project goals because of the way in which it structures tasks directly over syntax trees, incorporates many streams

of data from all words of a sentence at once, and makes fine-grained decisions for each leaf in the tree factoring in the "global" sentence information. Although sentiment analysis is a different task from negation scope detection, the problem framing and motivation are very similar, so this kind of model is of great interest. This model is used as a starting point and will be discussed in more detail, as well as model extensions, in chapter 3.

Chapter 3

Description of Work

3.1 Reframing the Problem

Many of the methods described in the Background chapter consume input sentences as a series of words and features and process them in a linear fashion. Syntactic features are also sometimes used but the input is still treated as a sequence of words. As described, syntax plays an important role and theoretically provides much information necessary for negation scope processing. This project re-frames NSD from a sequence labeling task into a tree-to-sequence task. Using a sentence parse tree and recursively compositional processing techniques, the model developed in this project will process input sentences as trees, not sequences, to produce scope judgments for words and syntactic constituents. This will demonstrate the key role that syntax and structure plays in scope processing.

The primary work completed in this project is a supervised machine learning model for the task of negation scope detection. The machine learning pipeline consists of several steps, for which a detailed description is now presented.

3.2 Data Preprocessing

The *SEM2012 NSD training, development, and test corpora come lightly preprocessed. Each corpus contains sentences from Arthur Conan Doyle's writing, annotated with features for each word. These include a part-of-speech tag, word lemma, and features expressing if the word is a negation cue and which negation scope(s) the word belongs to (sometimes a sentence will have two cues, and thus two scopes). Additionally, a Context Free Grammar (CFG) parse is provided for each sentence.

The first step in the project pipeline is further preprocessing. All sentences are ingested and parsed using the EasyCCG parser [24] to produce new constituency parses. CCG is preferred over CFG parses because they come with a guarantee of binary structure useful for the TRNN model, and for other reasons discussed in chapter 2.1.2. In some cases a constituent node may have only one child — since they are not directly combinatory, during preprocessing these nodes are simply removed to restore binary structure. Sentence tokens are lowercased and a vocabulary of the unique words in the training corpus is produced (some model variants will use this vocabulary for comparison, but the main model will not). The words in each corpus are replaced by unique integer tokens according to this training vocabulary — any out-of-vocabulary word observed in the development and test sets are replaced with the special "UNK" token. This step is performed after CCG parsing so that the parser sees all input words and has the best chance of generating a correct parse tree. Since the *SEM2012 dataset provides a CFG parse for all sentences this decision does not augment the provided dataset, but merely replaces one standard feature.

3.3 The Graph Convolutional Neural Network

This section serves to motivate the use of a Graph Convolutional (Neural) Network (GCN) in this project, and to briefly describe the experimental setups used to test it. The GCN was not successful on the task of negation scope detection so description will be kept brief but informative.

3.3.1 Motivation

A Graph Convolutional Neural Network is a model architecture developed by Kipf and Welling [21]. It was proposed as a method of solving partially supervised problems on graphs, represented as a set of nodes and a set of edges that connect them. The GCN consists of a simple transformation with nonlinearity on each node representation, followed by a convolution operation. This operation is applied once for each node n in the graph and combines (sums) the representations of n's neighbors into a new representation of n, thereby sharing information in local neighborhoods. One GCN layer propagates information one "hop" away, and more layers may be applied in sequence to spread information further.

This sounds superficially ideal to test the project hypothesis of learning negation

scope by structure. A syntactic parse tree may be treated as a graph with constituents as nodes and parent/child relationships as edges. The network may be assessed and trained based on the results it provides for "leaf" nodes (sentence words for which *SEM training labels exist). Applying a GCN with sufficient number of layers to this construction sets up the problem in a way which may provide an answer to the hypothesis. Can the scope of negation be learned by passing information along syntactic paths only?

A GCN is a simple network, and this makes it an ideal test. The basic model by Kipf handles only undirected graphs and has only two methods of updating information states: the nonlinear transformation and the convolution (sum of neighbors). By making few assumptions about the data it is very flexible and suited for testing the project hypothesis.

3.3.2 Experiments and Iteration

The initial outcome was disappointing. The network becomes stuck in a strong optima in which it simply classifies all constituents (graph nodes) as in-scope. The GCN was extended in several ways to address its simplifying assumptions about the data in an attempt to overcome this. First, directionality was added to graph edges by identifying parent relationships and child relationships. Recognizing that convolution (summing) loses ordering information about relationships if there are many neighbors, the CFG parses were swapped with CCG parses. This simplified the trees as the CCG parses are strictly binary trees. After no change was observed in model behavior after both extensions, the convolution operation was replaced with a self-attention mechanism, similar to a Graph Attention Network [43], to enable nodes to selectively attend to specific neighbor information. Still, the experiment yielded the same results. Without positive results no explanation about why the architecture failed may be presented with certainty. However, having accounted for the major assumptions made by the basic GCN, there remains one major difference between this model and other successful ones on the task. It may be conjectured that when processing scope through a constituent parse, it must be processed by serially propagating information in a tree format, and not in parallel as in a graph.

3.4 The Tree Recursive Neural Network

This section serves to motivate the use of the Tree Recursive Neural Network in this project, and describe its technical construction and process of learning from data. This model is successful at the task of negation scope detection and is further analyzed in the Evaluation chapter.

3.4.1 Motivation

To recap, Steedman [39] describes a theory of logical polarity in which word and phrasal polarity can be determined by calculating the effects of polarity flipping lexical items. These calculations involve both propagating polarity upward in a CCG parse and also downward. Such a system is complex to describe in rules but should be learnable by example. The Global Belief Tree Recursive Neural Network (GB-TRNN) of Paulus [30] is philosophically aligned with this theory, and serves as a starting point for this project. The GB-TRNN operates on a sentence and its syntax tree, sequencing together both an upward and downward recursive pass. The upward pass recursively combines constituents in the tree from leaves to root, building up to a single, global information state vector. The downward pass, which is conditioned on this state, then recursively unfolds the global vector following the parse tree from the root to the leaves. Although similar, the model developed in this project is architecturally distinct and so will be referenced differently and for concision simply as the TRNN.

This architecture is designed to gather local information across the sentence and propagate relevant parts of that information to other local points in the sentence for per-word decision-making. This is an ideal framework for the task because it directly models information flow between constituents using sentence structure, testing the research hypothesis.

3.4.2 TRNN Learned Parameters

As a neural network, the TRNN processes inputs in a “forward computation” through the layers of the network to produce scope judgments. During training, a “backward computation” propagates numerical gradients back through the network based on an error function. This gradient signal is automatically calculated using PyTorch [3], and is used to update specific parameters of the model which over time converge to values which minimize the error.

The TRNN model principally learns one embedding matrix for CCG syntactic constituents $E_{CCG} \in \mathbb{R}^{|V_{CCG}| \times s}$ (where V_{CCG} is the CCG tag vocabulary and s is 50, the syntactic hidden unit size used). The TRNN also learns five transformations used at different points in the network which are further explained in the sections below. In this project the hidden size h is chosen to be 200. These parameters are:

1. The matrix $H \in \mathbb{R}^{(s+c) \times h}$ (with bias term $\mathbf{b}_H \in \mathbb{R}^h$) which transforms leaves into hidden states (c is a binary feature expressing if the word is a negation cue)
2. The upward recursive matrix $W^{\uparrow} \in \mathbb{R}^{(2h+s) \times h}$ (with bias term $\mathbf{b}^{\uparrow} \in \mathbb{R}^h$)
3. The global information reversal matrix $G \in \mathbb{R}^{h \times h}$ (with bias term $\mathbf{b}_G \in \mathbb{R}^h$)
4. The downward recursive matrix $W^{\downarrow} \in \mathbb{R}^{(2h+2s) \times 2h}$ (with bias term $\mathbf{b}^{\downarrow} \in \mathbb{R}^{2h}$)
5. The in-scope/out-of-scope classifier matrix $C \in \mathbb{R}^{2h \times 2}$ (with bias term $\mathbf{b}_C \in \mathbb{R}^2$)

3.4.3 The Upward Pass

The first step in the TRNN is an upward pass through the syntax tree, illustrated in Figure 3.1. It is important to note that the main TRNN described in this project does not use actual words, only syntactic constituents. Words are added to the figure for the purpose of readability only.

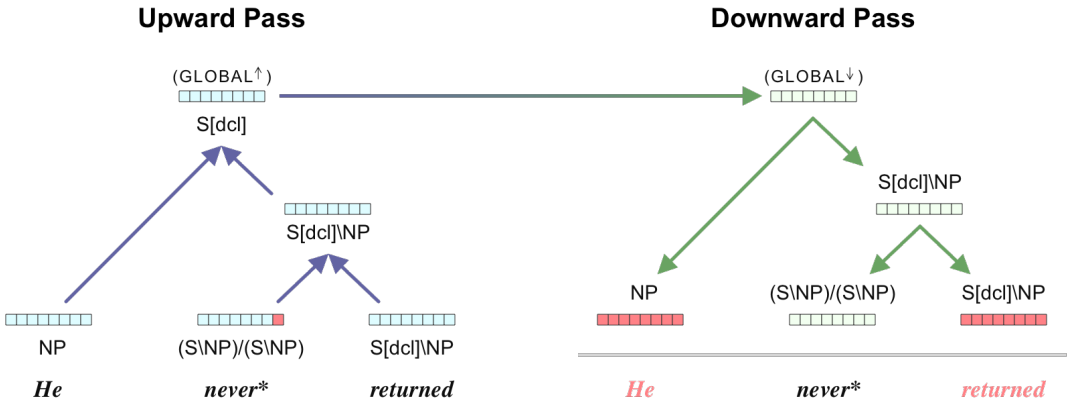


Figure 3.1: (Left) The upward pass: recursive composition of constituent states from leaves to root. (Right) The downward pass: recursive decomposition of constituent states from root to leaves. The arrows show the direction of composition/decomposition, and point to syntax labels used as target input. Neg cue starred* and scope in red .

The upward pass of the TRNN recursively combines the information states of two child constituents to produce a resulting state for the parent constituent. The GB-TRNN of Paulus consumes two recurrent connections: the child states $\mathbf{u}_{left}$ and $\mathbf{u}_{right}$. In the base case of the recursion (parse tree leaves) the information state of the node used for input is the result of the transformation H. First the syntactic embedding $\mathbf{s}_{leaf}$ is concatenated with the binary cue feature $\mathbf{c}$, then this result is transformed using H into a $\mathbf{u}$ vector fit for use in recurrent connections. Importantly, word embeddings are not used as input here, only syntax embeddings. Notably, the recursive transformation in this project takes one additional input $\mathbf{s}_{parent}$, the embedding of the target parent's syntactic constituent, which guides TRNN scope computation. These are concatenated together and multiplied by the upward matrix $W^{\uparrow}$. A bias term $\mathbf{b}^{\uparrow}$ is added, and the hyperbolic tangent nonlinearity is applied. This produces a target output state $\mathbf{u}_{parent}$ of the parent constituent which will be used as a recurrent connection in future computations.

To illustrate this the result of the upward pass, $\texttt{GLOBAL}^{\uparrow}$ information state, is now formulated for the example in Figure 3.1. This is also the resulting state for `S[dcl]`, the top constituent in the tree.

$$\mathbf{u}_{\texttt{NP}} = \mathbf{s}_{\texttt{NP}} H + \mathbf{b}_H$$
$$\mathbf{x} = [\ \mathbf{u}_{\texttt{NP}}\ ;\ \mathbf{u}_{\texttt{S[dcl]\NP}}\ ;\ \mathbf{s}_{\texttt{S[dcl]}}\ ]$$
$$\texttt{GLOBAL}^{\uparrow} = \mathbf{u}_{\texttt{S[dcl]}} = tanh(\mathbf{x}W^{\uparrow} + \mathbf{b}^{\uparrow})$$

3.4.4 The Downward Pass

The second step in the TRNN begins by transforming the $\texttt{GLOBAL}^{\uparrow}$ upward information state using G into the $\texttt{GLOBAL}^{\downarrow}$ downward information state following Paulus.

The $\texttt{GLOBAL}^{\downarrow}$ state is recursively unfolded down the tree using the downward recursive cell with weights $W^{\downarrow}$, bias term $\mathbf{b}^{\downarrow}$, and hyperbolic tangent nonlinearity. This produces a globally-informed, local information state $\mathbf{d}$ for each constituent in the syntax tree. The recursive cell consumes two recurrent inputs following Paulus: the downward parent state $\mathbf{d}_{parent}$ and upward state $\mathbf{u}_{parent}$. It produces a double-wide state vector which is split into left- and right-child state vectors. Similarly to the upward pass, the downward recursive cell in this network additionally consumes an embedding for the target syntactic constituents of both children to better inform scope computation down the tree. In total, at each step this recursive operation consumes four inputs and produces two outputs.

To illustrate this, the **d** information states for NP (the leaf node for "He") and S[dcl]\NP are now formulated for the example shown in Figure 3.1.

$$\mathbf{x} = [\ \mathbf{u}_{\texttt{S[dcl]}}\ ;\ \mathbf{d}_{\texttt{S[dcl]}}\ ;\ \mathbf{s}_{\texttt{NP}}\ ;\ \mathbf{s}_{\texttt{S[dcl]\NP}}\ ]$$

$$[\ \mathbf{d}_{\texttt{NP}}\ ;\ \mathbf{d}_{\texttt{S[dcl]\NP}}\ ] = tanh(\mathbf{x}W^{\downarrow} + \mathbf{b}^{\downarrow})$$

3.4.5 Classification

Finally, after the upward and downward passes of the TRNN the classification transformation C is applied to constituents to make scope judgments using the outputs from both passes.

To produce a judgment for a constituent, both the upward and downward context vectors **u** and **d** for the constituent are concatenated, then transformed with C to produce the classification of in-scope or out-of-scope. Figure 3.2 shows this concatenation of upward and downward states.

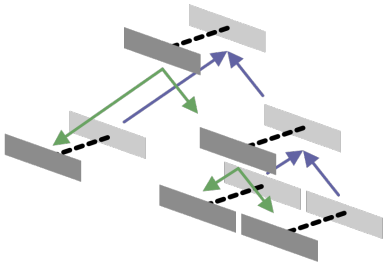


Figure 3.2: The scope classifier reads both the upward state and downward state (concatenated on the dotted lines) to classify a constituent.

3.5 TRNN Model Variations

The primary TRNN model evaluated in this project relies only on the syntactic parse tree of input sentences. To justify disregarding the identities of the words themselves, two additional model variants were developed and tested: model L using jointly-learned word embeddings, and model B using pretrained BERT embeddings [9] via the Huggingface implementation [18].

The word embeddings for model L are stored in E_{words}, an additional matrix of size $\mathbb{R}^{|V| \times h_{words}}$. V is the training vocabulary including the "UNK" word for use in testing as described in Data Preprocessing (in total 2526 unique tokens), and h_{words} was chosen to be 200 dimensions following Fancellu [10]. The embedding weights in model L are randomly initialized and learned jointly during training.

Word embeddings in model B are handled differently due to BERT implementation details. Because BERT provides contextual word embeddings specific to each sentence, embeddings are collected for each word in every sentence, totaling 21,006 embeddings of size 768 each. In many cases BERT splits words into subword units

and provides embeddings for each unit. However, embeddings must align with the leaves of each syntax tree (they must represent whole words), so subword embeddings are summed to form a single embedding for each word. Further, since BERT comes as a prepackaged model already trained on outside data, embeddings are derived for all words in the development and test sets even if they do not appear in the training set.

The word embeddings are derived differently between the two models but are used in the same way. The transformation H, which converts the input representation of a leaf node into a hidden state vector, is modified to accommodate the extra word embedding as input features. Thus two H transformations are used: $H_L \in \mathbb{R}^{(s+w_L+c)\times h}$ and $H_B \in \mathbb{R}^{(s+w_B+c)\times h}$. Both transformations retain the same bias term $\mathbf{b}_H \in \mathbb{R}^h$. Here $w_L = h_{words} = 200$ as stated above, and $w_B = 768$ as per the output size of BERT.

After leaf representations have been transformed into hidden states both experiments proceed as normal. Results of model L and model B are discussed in the Evaluation chapter.

3.6 Optimization

The TRNN parameters are learned through a process of optimization using gradient backpropagation. The optimization program is described below with hyperparameter settings used in this experiment.

3.6.1 Optimization and Objective Function

The Adam optimizer [20] was used with a learning rate of 0.001, and a decay scheduler which drops the learning rate by one order of magnitude if 10 epochs pass with no loss improvement on the development set. Early stopping was also applied if no development set loss improvement was observed after 20 epochs. Further, training examples were randomly shuffled each epoch and divided into minibatches of size 30. After each minibatch the optimization step updates the model parameters.

The cross entropy loss of model output probabilities with true labels was used as the objective function to optimize. Since optimization was stochastic minimizing this function is equivalent to maximizing the expected model probability of correct classifications. As described further in Evaluation, the model was trained with this objective but performance was measured using the F_1 score of the in-scope class. This difference is significant when considering that some example sentences contain negation cues but

no scopes (for instance, an interjection like "no!"), which have F_1 scores of 0 since they contain no true positive scope labels. These cases make it clear that training to predict the not-in-scope class is just as important as training to predict the in-scope class. Further discussion on performance measurement continues in the Evaluation chapter.

3.6.2 Regularization

The training set is small, consisting of fewer than 1,000 sentences. Because of this regularization is an important consideration to help alleviate the problem of overfitting to sparse data.

To this end dropout was heavily applied to connections between data transformations within the TRNN. Dropout is the process of randomly "destroying" connections between neural network layers during a forward computation [37]. By zeroing out random data values the network is forced to learn more generalizable parameters that are less dependent on co-occurrences in the training data, boosting test performance on unseen data. Gal [13] provides a framework for applying dropout within recurrent neural network modules which suggests using a predetermined dropout mask for recurrent connections, fixed at the start of each forward pass over a minibatch. This prevents the recurrent output from being virtually completely zeroed out if instead a newly randomized dropout mask is applied after each recurrence. In this case large data samples require more recurrent connections which would result in more data being dropped. Gal's framework avoids this problem by fixing the dropout mask ahead of time so only a fixed number of connections are dropped independent of variations in the data. In addition to these recurrent masks, other dropout masks are applied at points between transformations. As per Gal, recurrent connection masks have a probability of zeroing out state values of 0.2 and non-recurrent masks have a probability of 0.5.

Chapter 4

Evaluation

4.1 Performance of the Negation Scope Detector Learned with Structure

The negation scope detection problem as described earlier in this report is formulated as a binary classification problem on the words within a sentence. A detector takes as input a sentence (and associated metadata, including an annotation of the negation cue) and computes an output binary label for each word, corresponding to the predicted scope of negation for the cue.

The *SEM2012 competition provides a suite of labeled datasets for this task: a training corpus of 848 negated sentences (due to double negation, this is 984 negative scopes), a development corpus of 144 negated sentences (173 scopes), and a testing corpus of 235 negated sentences (264 scopes). These consist of sentences where each word is annotated with a gold label of "in scope" or "out of scope," and the negation cue (if present) is also annotated. Further, a CFG parse is also provided for each sentence. As described in Methods, the model trained in this project learned only from the training data and treated the development dataset as held-out data. This was used during training to guide the learning rate decay only, and the test set was used to evaluate final model performance.

To measure performance, it is first useful to understand why a simple measure like prediction accuracy may be misleading for this problem. The scopes licensed by negation cues are commonly unbalanced compared to the number of out-of-scope tokens in the same sentence. The *SEM2012 dataset, in fact, has approximately a 2:1 ratio of out-of-scope tokens to in-scope tokens, given the sentence contains a negation

cue (the ratio would be much more drastic if all sentences in the dataset are considered, even ones with no negations). Thus a model can easily achieve 67% accuracy by simply predicting all words to be out of scope of any cue.

There are a few ways to measure system performance on this task and in fact the original *SEM2012 competition used several metrics, including measures of complete scope detection and partial scope detection of individual scope tokens. The primary metric that has since been used on this task and dataset is the F_1 measure of individual scope tokens predicted as in-scope of their respective cues. This is the harmonic mean of precision and recall calculated for the "in-scope" label class. Precision measures the percentage of tokens predicted as in-scope which are correct, and recall measures the percentage of truly in-scope tokens which are predicted. Taking the harmonic mean gives a single metric we can use to compare models, but precision and recall are both useful themselves for comparison, as they portray how conservative the model behaves. Precision greater than recall indicates the model prefers "sure bets," while recall greater than precision indicates a willingness to predict based on less evidence.

The F_1 metric is calculated across the entire *SEM test corpus instead of per-sentence. This at first seems unintuitive because the task defines output per-sentence, and it might be reasonable to calculate a score such as the average F_1 score (or accuracy) per sentence. Unfortunately, scoring over the whole corpus means that for future models to compare with current ones they must test with the *SEM test corpus to calculate a comparable F_1 score. However, this metric does provide some normalization over sentence variation such as length and complexity, since it aggregates over the entire corpus. Additionally, given that an F_1 measure is used for comparison, calculating the score over the whole dataset is helpful to account for special cases in which the F_1 score should legitimately be low, such as when a cue has no scope as in a conversational interjection. If predicted correctly this kind of sentence would individually have an F_1 score of 0 due to having all true negative predictions (and zero true positives). Scoring over the whole corpus again normalizes for this phenomenon since the zero counts will be effectively ignored.

Table 4.1 shows performance results on the evaluation dataset for the Tree Rescursive Neural Network and key comparison models. Not shown are results for Fancellu's Dependency-LSTM or GCN, since they have no published results on this dataset.

Model	Precision	Recall	F_1 Score
BiLSTM + PoS + Word2Vec Emb (Fancellu)	92.62	85.13	**88.72**
MRS *Crawler* (Packard)	85.8	68.4	76.1
MRS $Crawler_P$ (Packard)	86.1	90.4	88.2
UiO_1 (Read, *SEM winner)	81.99	88.81	85.26
TRNN (No Word Emb)	91.04	85.60	**88.24**
TRNN + BERT Emb	89.36	86.54	87.93
TRNN + ST Emb	92.56	83.38	87.73
TRNN + BERT Emb (No Regularization)	86.46	82.05	84.20
Baseline (all scope tokens negated)	37.23	100.00	54.26

Table 4.1: Table of Scope Tokens results for the *SEM2012 Negation Scope Detection test dataset of Conan Doyle stories. “PoS” is added part-of-speech tags. “ST Embs” is self-trained embeddings learned jointly with the task. “BERT Embs” is pretrained embeddings from BERT, held frozen during training.

4.1.1 TRNN Self-Comparison with Ablation Studies

During this project several features were added to the TRNN in an attempt to improve performance. Ablation studies were performed to test these features in isolation. In each test, one feature was deactivated to highlight individual contributions to model performance. The results can be seen in Table 4.1, and are now discussed.

First and most crucially it’s clear that explicit word representations are not necessary for high performance. Using only small, learned representations of syntactic constituents, a simple cue feature, and the recursively compositional network the model is able to effectively perform on the task at an equal or possibly better level than with added word embeddings. Interestingly, both self-trained word embeddings and pretrained BERT embeddings provided little to no benefit for the system in terms of F_1 score. In many applications pretrained embeddings provide a performance boost, especially with a small training dataset, but it seems that even high-quality embeddings do not provide a distinguishable benefit to performance on this task.

Additionally, it is important to discuss the contribution of regularization. The training dataset contains fewer than 1,000 example sentences and a vocabulary size of 2,525 unique words, both of which are quite small. This could provide a significant challenge if the model were heavily reliant on the vocabulary since the test dataset contains many

words previously unseen to the model. Before adding regularization, training performance consistently rated 5 or more F_1 points ahead of test, signaling overfitting to the training data. Adding dropout in several places throughout the model as a regularization technique provided a 3-4 F_1 point boost on test performance. It is likely that such a large boost was achieved because there is less possible variation in sentence structure than there is in vocabulary usage (a simple example of this is that in any sentence you can switch a word with a synonym to produce a new sentence without changing the syntactic structure of the sentence). Because the model relies on patterns in structure to predict scope and not on specific word choice, it is shown empirically that the small dataset provides enough examples of structure to generalize well to the test set.

4.1.2 Comparison of the TRNN to Other Models

The Tree Recursive Neural Network outperforms the winner of the *SEM2012 competition, UiO_1. This demonstrates the effectiveness of a neural network approach over a classical machine learning technique even when similar information (syntactic paths) is used in the learning process. Because of the flexibility in neural network modeling the TRNN directly propagates syntactic information flowing upward and also downward through the parse tree, from all words in the sentence at the same time. Conversely, in order to fit the restrictions of the Support Vector Machine used in scope detection, UiO_1 must identify a singular path in the tree for the negation cue and make decisions about its scope in isolation of other syntactic information in the tree, specifically from other words and syntactic structures. This poses an opportunity to lose information and may explain the difference in results.

The TRNN also performs within half a point of the highest-scoring model to date, Fancellu's Bidirectional LSTM Network. This is interesting because these two models process sentences very differently. The BiLSTM is dependent on word representations, processing words in sequence and thus in direct context with their neighbors. However, the TRNN instead processes only syntactic constituents, and does so in a purely compositional way. Words that are neighbors in a sentence may be very distant in the parse tree yet the structure is enough to inform scope decisions, even without identifying individual words. Thus, the TRNN is able to compete with the BiLSTM while completely discarding the vocabulary, which means using far fewer parameters.

Finally, it is useful to compare the results to Packard's Minimal Recursion Semantics Crawler. The purely semantic *Crawler* by itself has an F_1 score of only 76.1, far

behind other methods. The "improved" model $Crawler_P$ falls back to UiO_1 (a syntactic model which already outperforms *Crawler*) in cases of low confidence. Packard claims that a combination of syntactic and semantic analyses provide the optimal strategy for a model of NSD. However, even with this improvement the TRNN scores the same, if not better, than $Crawler_P$, demonstrating that explicit semantic analysis may not actually be necessary to achieve high performance on this task.

4.1.3 A Note on TRNN Performance Testing

It should be noted that the time limit for this project placed restrictions on the kinds and amounts of work done. The scores shown in Table 4.1 were not the result of a systematic hyperparameter search, which would take considerable time to complete yet might improve scores. Instead optimization heuristics and "general practices" were applied in the model tests. This is itself helpful in that it shows the TRNN's success is achievable with conventional, "off the shelf" settings.

4.2 Analysis of Model Predictions

4.2.1 Tree Reporting Notation

As discussed in Methods, the training supervision signal consists of in-scope/out-of-scope labels for words only. However, model predictions are produced for every constituent in the parse tree, using the same learned classifier as for word prediction. This may provide some insight into the kinds of rules learned by the model. To fully demonstrate results, examples will be printed as parse trees using the following scheme:

1. A sentence is displayed with its CCG parse. The training and testing data are modified to omit constituents with only one child (and are thus omitted from these printouts).

2. For readability, word-level CCG supertags are replaced with their actual lexical item from the sentence. It is important to reiterate that the model processes only CCG tags and otherwise cannot identify individual words.

3. Cue words are appended with a '*' character.

4. The model's in-scope predictions are colored red .

5. The *SEM2012 gold scope labels are underlined.

4.2.2 Overall Performance

Model performance on the test set [1] of 263 sentences is shown in Figure 4.1, which displays a histogram of the accuracy scores for all sentences. The model is able to classify all tokens in 149 sentences perfectly, and classifies tokens in 93.9% of sentences better than chance.

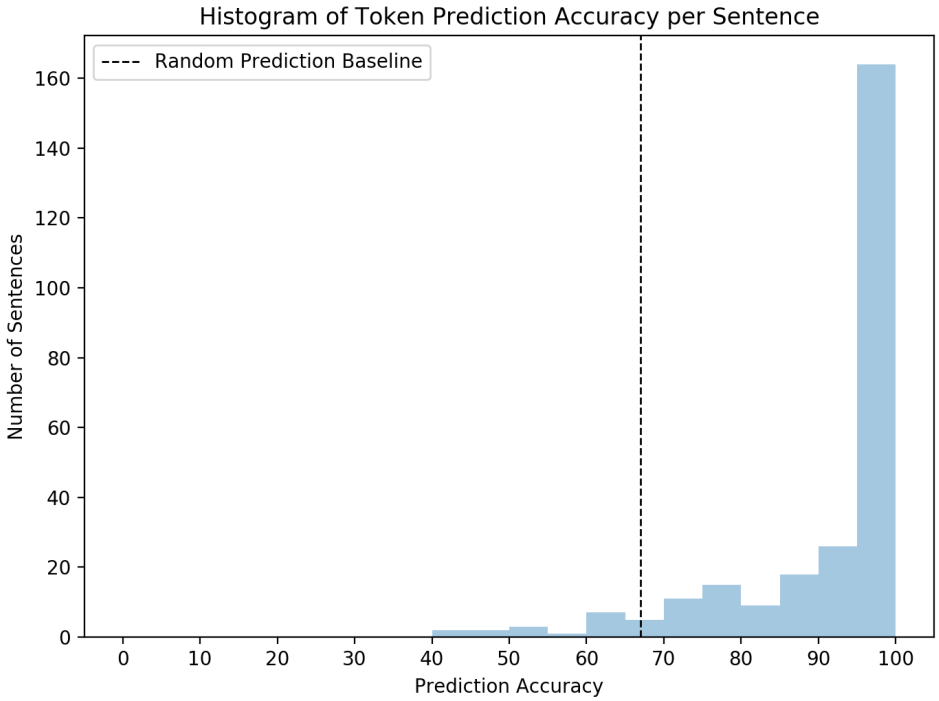


Figure 4.1: Distribution of individual sentence scores. Each sentence was scored for accuracy (percentage of correct token classifications) and categorized into bins. 149 of 264 sentences were predicted with 100% accuracy. The plotted baseline represents the accuracy of random guessing respecting the prior distribution of classes in the dataset (about 66%).

Figure 4.2 shows a simple and well-performing example which scored 100% accuracy. The scope of the negation cue covers most tokens except "but," the period, and the cue itself, and this is correctly predicted by the model. It is also interesting to note that the model predicts the entire phrase "very far" as in-scope according to the parent constituent combining the two tokens.

[1] After this analysis the author recognized that using the (smaller) development dataset would be the ideal dataset to use here in order to fully respect the held-out test data. Given enough time this analysis would be done on development set data.

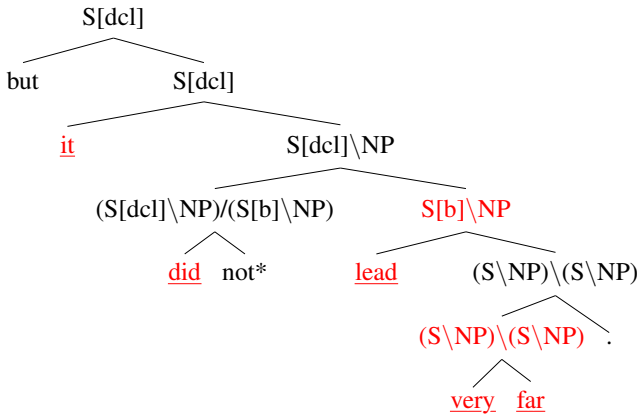


Figure 4.2: A correct sample output from model predictions on the test set.

Figure 4.3 shows another well-performing example with a more complicated syntactic structure. Correctly, only the local subject "I," the verb, and the modal auxiliary verb are predicted as in-scope. The syntactic tree clearly distinguishes this clause from the rest of the sentence, providing important evidence to the model about the projection of negation scope. This is frequently the case seen in test sentences: syntactic subtrees are strongly correlated with correct scope projections and usually provide enough information to make predictions without needing further lexical information. More examples are shown in Appendix A at the end of this report.

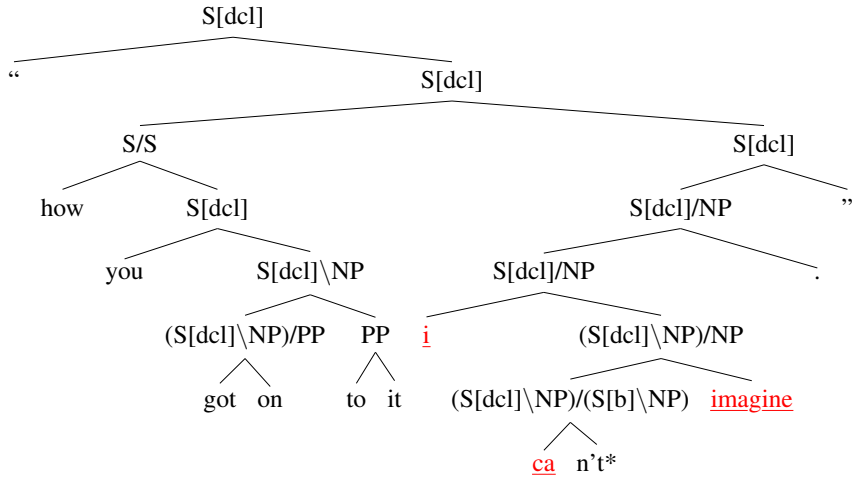


Figure 4.3: A correct sample output from model predictions on the test set.

Beyond what is predicted as in-scope, it is interesting to note what is correctly *not* predicted. For instance, Figure 4.3 illustrates that punctuation marks are excluded even when they are directly attached to a syntactic subtree which contains a complete scope. Recalling that the actual training instances replace the sentence words with their CCG

supertags helps explain this. CCG tags are sufficiently descriptive of their constituents that the model can broadly identify punctuation tokens and other indicators of scope boundaries to make predictions.

Further, several sentences in the dataset contain two negation cues and thus license two (sometimes overlapping) scopes. Figures 4.4 and 4.5 show one case where a negation scope is nested inside another scope. The model receives the sentence as two separate training examples, each with one cue annotated. Notably, the model learns that the cue nested inside the larger scope is itself a scope token — in all other cases cues are universally not scope tokens themselves according to *SEM annotation guidelines.

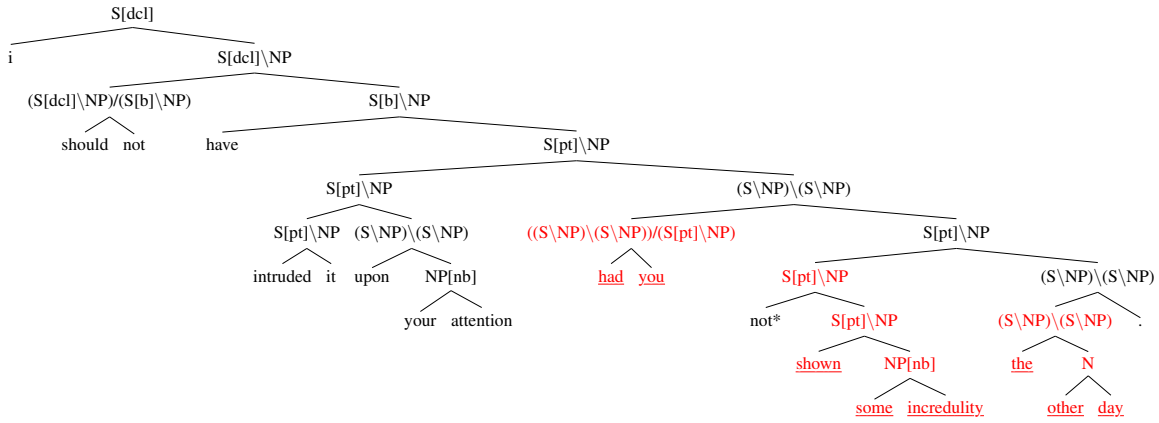


Figure 4.4: A correct sample output from model predictions on the test set. This sentence contains two negative cues: Figure 4.5 shows the other cue and scope.

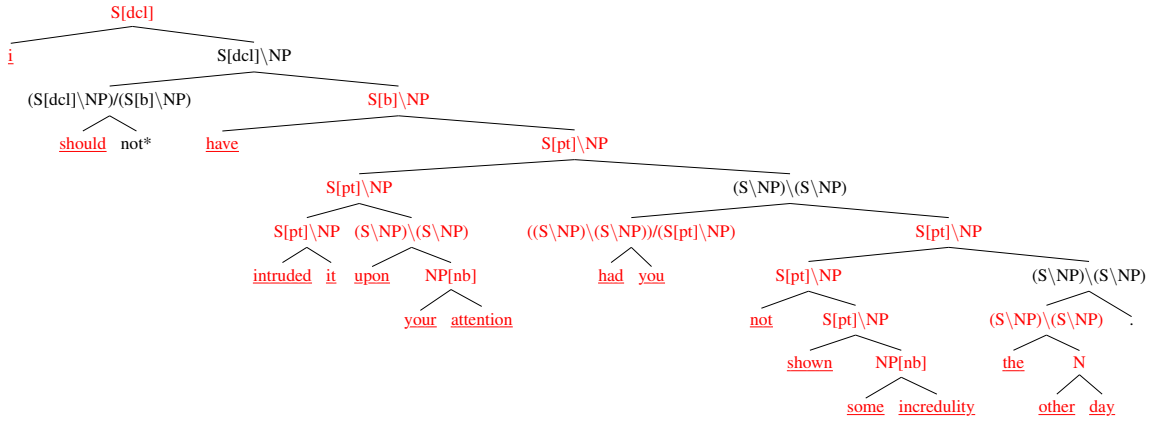


Figure 4.5: A correct sample output from model predictions on the test set. This sentence contains two negative cues: Figure 4.4 shows the other cue and scope.

The tree format of the data is crucial to TRNN operation. In an analysis of model capability, several Pearson correlation coefficients were calculated to determine to

what extent data characteristics or the format may affect performance.

1. A Pearson *r* of 0.020 was calculated ($p = 0.74$) for sentence accuracy vs. sentence length, indicating no correlation with model performance.

2. A Pearson *r* of -0.006 was calculated ($p = 0.92$) for sentence accuracy vs. maximum parse tree depth of the sentence, indicating no correlation with model performance.

3. A Pearson *r* of 0.023 was calculated ($p = 0.71$) for sentence accuracy vs. parse tree depth of the cue, indicating no correlation with model performance.

4. Parse (binary) tree balance was approximated using the formula `(max tree depth) / (sentence length)`, where smaller values indicate more even balance. A Pearson *r* of -0.003 was calculated ($p = 0.96$) for sentence accuracy vs. parse tree balance, indicating no correlation with model performance.

These calculations suggest that the model is quite robust with respect to variation in data length and tree shape.

4.2.3 Error Analysis

A detailed analysis shows 16 test sentences (6.1% of the test corpus) had a token classification accuracy at or below random chance guessing (approximately 66% accuracy respecting the class imbalance). These poor-performing test cases may indicate weaknesses in the system and opportunities for improvement, and are now considered.

Analysis shows that of these 16 sentences the largest group of similar errors consisted of 7 CCG parsing errors, and it is likely these errors significantly influenced TRNN scope processing. For instance, in Figure 4.6 a left-branching subtree ("his mother's heart") should be differently constructed and attached directly to the verb "break," as it is the verb's object. The true scope ends after this phrase and it is likely that a better parse would make this more clear syntactically. The clause at the end of the sentence ("that appears to be irrelevant.") is improperly treated as a syntactic sibling of this object, and thus the model predicts incorrectly that it is also in scope.

It is notable that of these 7 example sentences, 5 contain an em-dash and 1 contains a semicolon (all with bad parse decisions involving these tokens). It is unclear what actually caused the EasyCCG parser to output these trees, but it may be possible that the parser was not perfectly trained for this writing style or genre (One of the

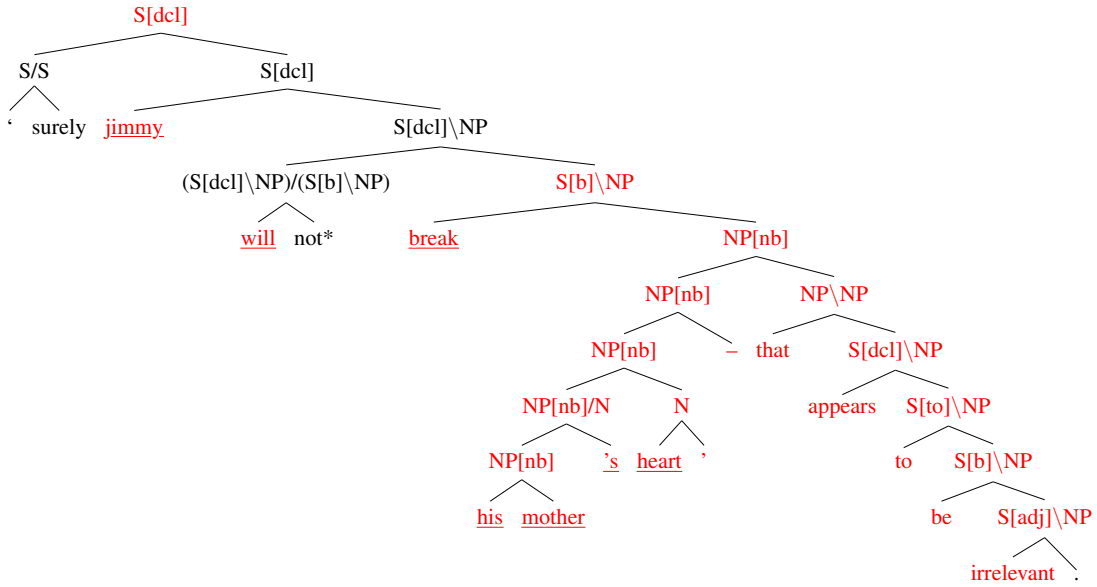


Figure 4.6: An incorrect sample output from model predictions on the test set.

training texts, *The Hound of the Baskervilles*, was originally published in 1902). It is also important to recognize the possible impact of decisions made about data preprocessing. The *SEM2012 corpora come lightly preprocessed with some tokenization, and although some additional preprocessing steps were applied the data may still contain disadvantageous features. An instance of this can be seen in Figure 4.6 where “mother” is split apart from the possessive “’s” attachment. This preprocessing may have impacted CCG parse decisions, resulting in consequences for the TRNN.

Packard and Fancellu note inconsistencies in the *SEM2012 gold standard negation scopes. Noted here is an instance which might also be an annotation inconsistency. Figure 4.7 shows a sentence constructed very similarly to that in Figure 4.3 yet with a different scope annotation. Both sentences are inverted in that they place a dependent clause ahead of an independent clause, with a negation cue in the latter clause in both examples. Yet the gold scope in 4.3 does not cover the dependent clause, whereas in 4.7, it does. If the independent clause weren’t in scope it would otherwise be perfectly labeled by the model. Four cases of possible inconsistencies were observed out of the 16 failure cases, and a more detailed analysis might show other inconsistencies significant to scoring.

Finally, it is worth considering the limited experience gained from the small training dataset. While the correctness of the parse in Figure 4.8 may be argued (the parsing of the quotation marks at least is questionable), it may also be argued that this is an edge-case in other ways. It is one of the smallest test cases and contains two two-word

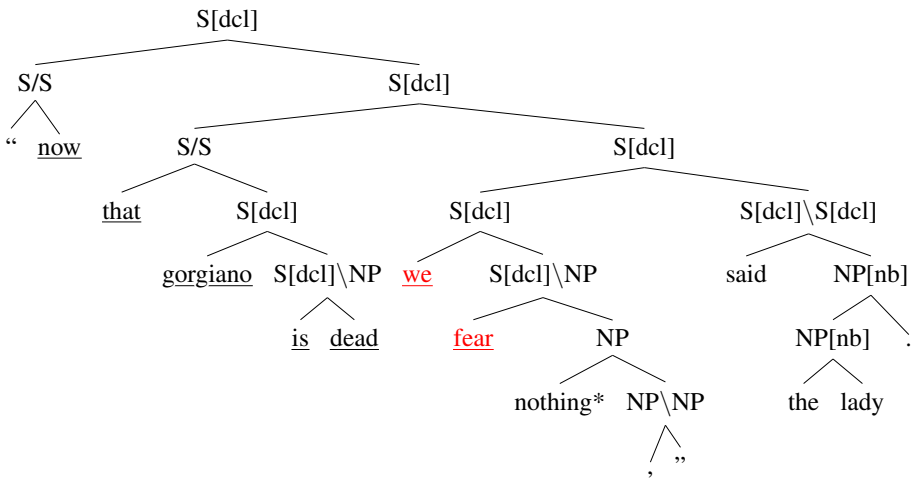


Figure 4.7: An incorrect sample output from model predictions on the test set.

clauses. There is also added complexity because one is a question and not a statement. Checking the training corpus revealed no training examples similar to “why not?” so it is possible that the failure is due to lack of experience generalizable to this example.

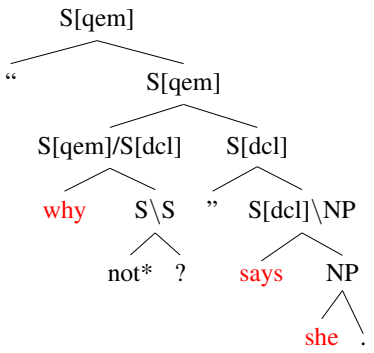


Figure 4.8: An incorrect sample output from model predictions on the test set.

4.3 Qualitative Inferences about Scope Colorings

The TRNN model predicts negation scope at all levels of the syntax tree using one learned classifier. A training signal is only provided for the leaves of each tree (which represent the words), so it’s undetermined what kind of rules or patterns the model may learn at higher levels of the tree. CCG parse trees allow for some interpretation of the model’s coloring choices at higher levels in the tree because by definition tags describe the grammatical roles of constituents at all levels. The full tree coloring results on the *SEM2012 test set will now be interrogated to look for patterns in model decisions.

4.3.1 Is Scope Processed in Distinguishable Grammatical Units?

Negation scopes frequently cover larger grammatical units than words, such as phrases and clauses. Does the model recognize this? This question can be addressed by examining local coloring decisions in each parse tree. In cases where a parent constituent has two child constituents classified as in-scope, the model might be expected to also classify the parent as in-scope. In fact, in 89.9% of these cases the model does recognize that the larger grammatical constituent is negated and colors it as in-scope. This compares to the average chance of being colored given no guarantee on the scope colorings of a constituent's children at 24.1%. The model has clearly learned a strong preference towards representing scope in larger grammatical units.

4.3.2 When is Scope Propagation Blocked?

Similar to the above scenario, when a parent constituent and only one child constituent are negated, this could indicate that scope information is blocked from flowing to the other child. Is the model learning meaningful rules for blocking scope? To test this, the most frequently blocked constituents are gathered and reported here.

Constituent	Count
.	89
,	35
(S\NP)\(S\NP)	30
NP\NP	30
S[dcl]\NP	28

Figure 4.9: Counts for the most blocked constituents. Blocked means not itself negated but both parent and sibling are negated.

The two most frequently blocked constituents are periods and commas, indicating the model has learned to strongly dissociate punctuation marks from semantic scope. The next highest constituent is functions which modify verb phrases: (S\NP)\(S\NP). Some examples from *SEM are "either." and "for so short a time." This makes some sense, as these phrases do not contribute to the *SEM definition of being a negated event or one of its direct arguments. Therefore they should not be included in the scope of a negation cue, even if they are surrounded by the scope.

Chapter 5

Conclusion

5.1 Conclusions from this Work

This report first describes and analyzes trends in the history of negation scope detection models. The hypothesis of this project poses a theory stemming from but opposed to this history: patterns in syntactic structure alone are sufficient to determine negation scope. This theory is tested using a Tree Recursive Neural Network architecture to directly model information flow between constituencies in a CCG parse of a sentence, without using the identities of words or other features derived from them.

The performance evaluation shows that the TRNN model developed in this project effectively determines negation scope through explicit processing over grammatical structure. Near state-of-the-art performance is achieved without identifying individual words or extracting word features, confirming the project hypothesis.

The significance of this result is highlighted with a comparison to the key models discussed in the Evaluation chapter. The TRNN model performs about the same as the BiLSTM model of Fancellu [10], which relies on word embeddings. Learning a vocabulary of embeddings is itself a demanding task and is shown to capture many aspects of language usage including syntax and semantics [26]. The TRNN method shows high performance is achievable without this overhead using only syntactic information. Further, using a neural network to directly model syntactic information flow is shown to significantly outperform models like UiO_1 which emphasize syntax but are limited by model architecture, such as an SVM, which require feature engineering. Finally, these results demonstrate that scope can be resolved without needing to derive a full semantic parse and then perform reasoning as Packard [29] claims is necessary.

This project successfully demonstrates a single, simple idea that diverges from

common practice in many of the effective and opinionated models, which frequently undervalue the contribution of syntax. The project results empirically support the underlying theory [39] that negation scope semantics are built on syntactic information irrelevant to the specific content of the sentence.

5.2 Directions for Future Work

The results of this research suggest followup experiments which may help analyze and extend understanding of the phenomena discussed.

First, a followup research question emerges when considering some of the background research. Negative polarity items are known to help human readers discern the existence and coverage of negation scopes, since they (mainly) appear in negative contexts. However, the model used in this project ignores the identities of these words and still performs well. Is this a coincidence? It could be that the *SEM dataset underrepresents NPIs and thus the model does not appear to struggle on this dataset, but might not perform well on other datasets. But it is also possible that the information provided by NPIs is subsumed by that of grammatical structure, and is not complementary. What kind of benefit, if any, could they provide for a system like the TRNN?

Second, other important questions remain about model performance. It's possible that the TRNN model of this project might be modified and improved through further research. Additionally, many tree-structured neural network models have been developed for tasks in sentence processing, like the Child-Sum Tree LSTM of Tai [42] and the Tree Attention Network of Ahmed [5]. These models may provide insight and further direction for improving the structure-processing architecture of the TRNN.

Many comparison models on this task make use of syntax data along with word data, like word embeddings. It may be interesting to reproduce some of these models and ablate them by removing the word data to examine how they perform. If performance does not suffer it may be inferred that they are learning from the syntax using a different architecture than the TRNN. However if they do perform worse, then the TRNN architecture may be credited as a particularly useful model of information processing with structure.

Finally, it is important to train and test the TRNN on text from other genres to study its extensibility. Conan Doyle's writing represents a very specific genre, writing style, and language, and different textual domains may expose new strengths or weaknesses in the TRNN system.

Bibliography

[1] *SEM2012: First joint conference on lexical and computational semantics.

[2] *SEM2012 shared task. https://www.clips.uantwerpen.be/sem2012-st-neg/index.html.

[3] Pytorch. Pytorch.org, 2019.

[4] Amjad Abu-Jbara and Dragomir Radev. Umichigan: A conditional random field model for resolving the scope of negation. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, SemEval '12, pages 328–334, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[5] Mahtab Ahmed, Muhammad Rifayat Samee, and Robert E. Mercer. Improving tree-lstm with tree attention. *CoRR*, abs/1901.00066, 2019.

[6] Emilia Apostolova, Noriko Tomuro, and Dina Demner-Fushman. Automatic extraction of lexico-syntactic patterns for detection of negation and speculation scopes. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 283–287, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[7] Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A Sag. Minimal recursion semantics: An introduction. *Research on language and computation*, 3(2-3):281–332, 2005.

[8] William D Davies and Stanley Dubinsky. *The Grammar of Raising and Control: A Course in Syntactic Argumentation*. Wiley, 2004.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[10] Federico Fancellu, Adam Lopez, and Bonnie Webber. Neural networks for negation scope detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 495–504, 2016.

[11] Federico Fancellu, Adam Lopez, Bonnie Webber, and Hangfeng He. Detecting negation scope is easy, except when it isn't. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 58–63, 2017.

[12] Federico Fancellu, Siva Reddy, Adam Lopez, and Bonnie Webber. Universal dependencies to logical forms with negation scope. *arXiv preprint arXiv:1702.03305*, 2017.

[13] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027, 2016.

[14] Anastasia Giannakidou and Hedde Zeijlstra. *The Landscape of Negative Dependencies: Negative Concord and N-Words*, pages 1–38. American Cancer Society, 2017.

[15] Laurence Horn. *A Natural History of Negation*. University of Chicago Press, 1989.

[16] Laurence R. Horn and Heinrich Wansing. Negation. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2017 edition, 2017.

[17] Rodney Huddleston and Geoffrey K. Pullum. *A Student's Introduction to English Grammar*. Cambridge University Press, 2005.

[18] Huggingface. Pytorch transformers. Huggingface.co, Jul 2019.

[19] Jin-Dong Kim, Tomoko Ohta, and Jun'ichi Tsujii. Corpus annotation for mining biomedical events from literature. *BMC bioinformatics*, 9(1):10, 2008.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2017.

[22] Natalia Konstantinova, Sheila CM De Sousa, Noa P Cruz Díaz, Manuel J Mana López, Maite Taboada, and Ruslan Mitkov. A review corpus annotated for negation, speculation and their scope. In *Lrec*, pages 3190–3195, 2012.

[23] Emanuele Lapponi, Erik Velldal, Lilja , and Jonathon Read. Uio2: Sequence-labeling negation using dependency features. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, SemEval '12, pages 319–327, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[24] Mike Lewis and Mark Steedman. A* ccg parsing with a supertag-factored model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 990–1000, 2014.

[25] Junhui Li, Guodong Zhou, Hongling Wang, and Qiaoming Zhu. Learning the scope of negation via shallow semantic parsing. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 671–679, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[26] Tomas Mikolov, Scott Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013)*. Association for Computational Linguistics, May 2013.

[27] Roser Morante and Eduardo Blanco. Description of *SEM2012 shared task: Resolving the scope and focus of negation.

[28] Roser Morante and Eduardo Blanco. *SEM2012 shared task: Resolving the scope and focus of negation. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International*

Workshop on Semantic Evaluation, pages 265–274. Association for Computational Linguistics, 2012.

[29] Woodley Packard, Emily M Bender, Jonathon Read, Stephan Oepen, and Rebecca Dridan. Simple negation scope resolution through deep parsing: A semantic solution to a semantic problem. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 69–78, 2014.

[30] Romain Paulus, Richard Socher, and Christopher D Manning. Global belief recursive neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2888–2896. Curran Associates, Inc., 2014.

[31] Zhong Qian, Peifeng Li, Qiaoming Zhu, Guodong Zhou, Zhunchen Luo, and Wei Luo. Speculation and negation scope detection via convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 815–825, 2016.

[32] Jonathon Read, Erik Velldal, Lilja , and Stephan Oepen. Uio1: Constituent-based discriminative ranking for negation resolution. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, SemEval '12, pages 310–318, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[33] Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140, 2016.

[34] Sabine Rosenberg. Negation triggers and their scope. Master's thesis, Concordia University, 2013.

[35] Walter Daelemans Roser Morante, Sarah Schrauwen. Annotation of negation cues and their scope guidelines v1.0., 2011.

[36] Beatrice Santorini. Node relations. https://www.ling.upenn.edu/∼beatrice/syntax-textbook/box-nodes.html.

[37] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[38] Mark Steedman. A very short introduction to ccg. https://www.inf.ed.ac.uk/teaching/courses/nlg/readings/ccgintro.pdf, 1996.

[39] Mark Steedman. Negation and polarity. In *Taking Scope*, pages 175–208. The MIT Press, nov 2011.

[40] Mark Steedman, Stephen Clark, and Julia Hockenmaier. Parsing with ccg. August 2002.

[41] György Szarvas, Veronika Vincze, Richárd Farkas, and János Csirik. The bioscope corpus: Annotation for negation, uncertainty and their scope in biomedical texts. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, BioNLP '08, pages 38–45, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

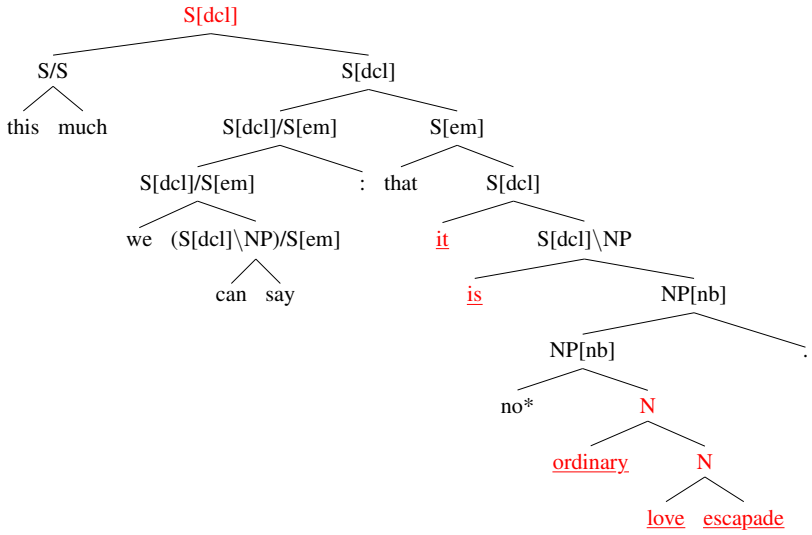
[42] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. *CoRR*, abs/1503.00075, 2015.

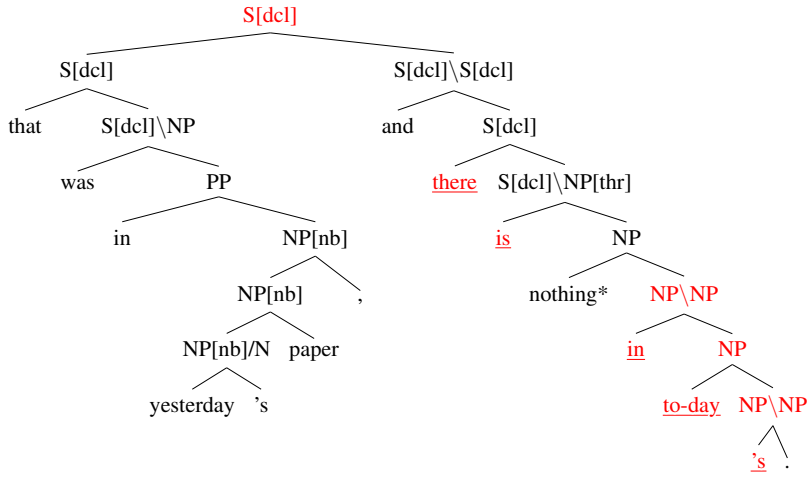
[43] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *arXiv e-prints*, page arXiv:1710.10903, Oct 2017.

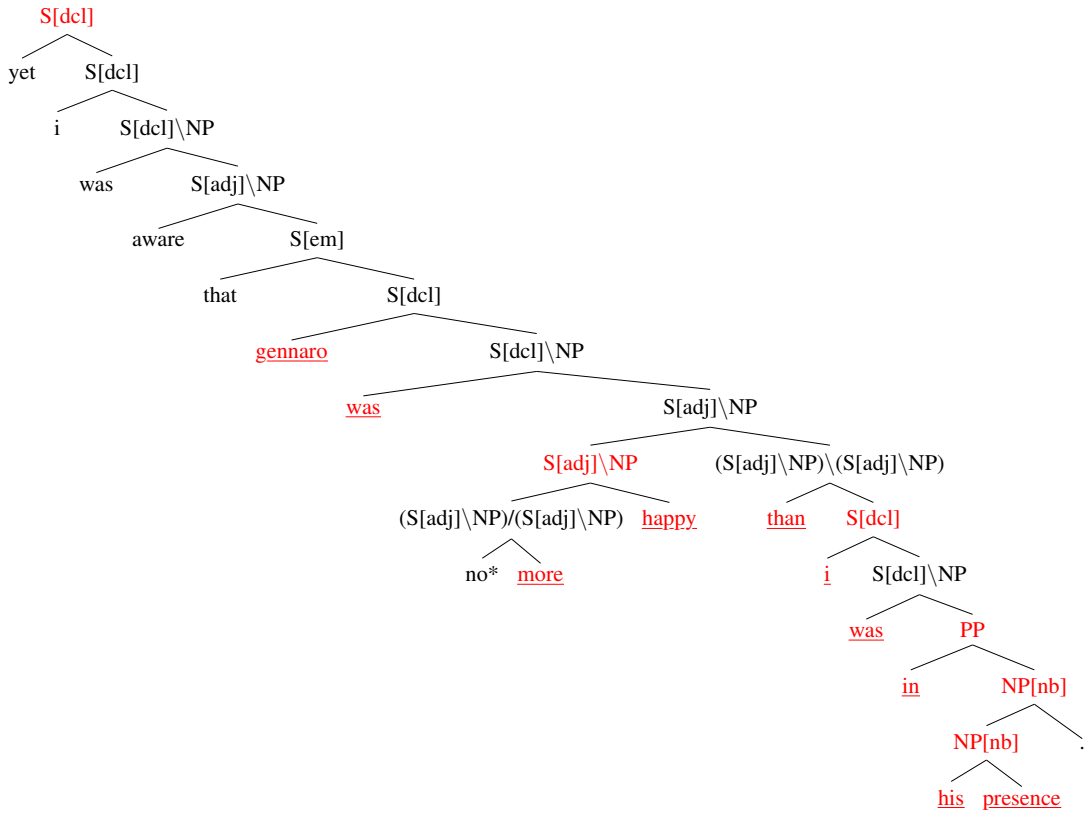
[44] Bowei Zou, Guodong Zhou, and Qiaoming Zhu. Tree kernel-based negation and speculation scope detection with structured syntactic parse features. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 968–976, 2013.

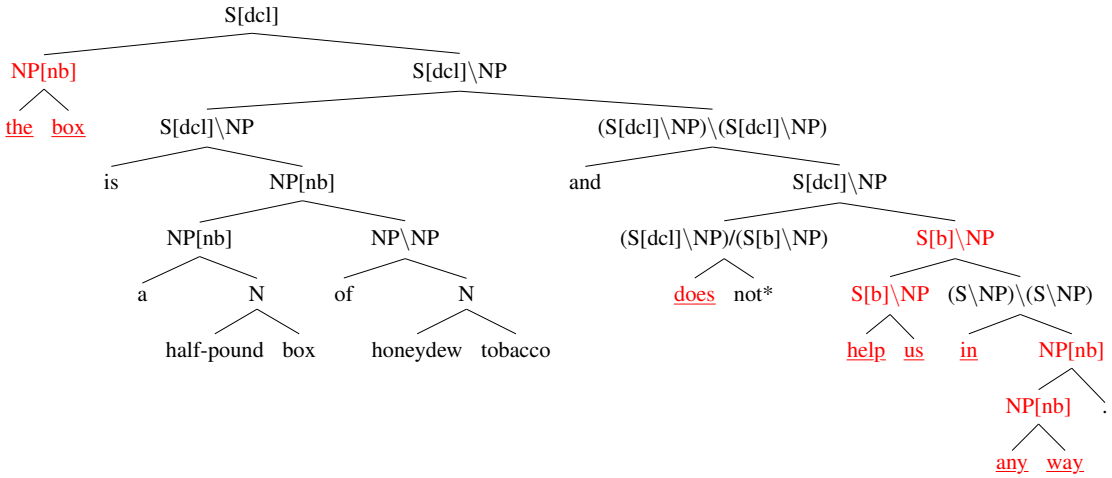
Appendix A

Supplementary Parse Tree Colorings






S[dcl]
yet
S[dcl]
i
S[dcl]\NP
was
S[adj]\NP
aware
S[em]
that
S[dcl]
gennaro
S[dcl]\NP
was
S[adj]\NP
S[adj]\NP
(S[adj]\NP)\(S[adj]\NP)
(S[adj]\NP)/(S[adj]\NP)
happy
than
S[dcl]
no*
more
i
S[dcl]\NP
was
PP
in
NP[nb]
NP[nb]
.
his
presence


S[dcl]
NP[nb]
S[dcl]\NP
the
box
S[dcl]\NP
(S[dcl]\NP)\(S[dcl]\NP)
is
NP[nb]
and
S[dcl]\NP
NP[nb]
NP\NP
(S[dcl]\NP)/(S[b]\NP)
S[b]\NP
a
N
of
N
does
not*
S[b]\NP
(S\NP)\(S\NP)
half-pound
box
honeydew
tobacco
help
us
in
NP[nb]
NP[nb]
.
any
way