Negation Scope Detection

Selin Acikel 2637714
Murat Ertas 2775481
Guo Ningxuan 2818054
Csenge Szabó 2803781

Abstract

This study explores the task of negation scope detection, which plays an important role in several Natural Language Processing tasks, including sentiment analysis, question answering and relation extraction. The research employs the *SEM CD-SCO corpus to carry out negation scope detection using various machine learning approaches. The first part of the paper presents feature-based experiments conducted using the Conditional Random Fields algorithm with different set of linguistic features extracted from the data. In the second part of the study, transformer-based classification is performed using a fine-tuned DistilBERT model for negation scope detection. The performance of the transformer-based model exceeded that of the feature-based one, showing a notable improvement with a 0.18 increase in token-level F1-score and 0.35 increase in span-level F1-score on the test data.

Github repository: `https://github.com/asa-sel/Group-B-applied-tm.git`

1 Introduction

Automatic negation detection is essential in Natural Language Processing (NLP) for downstream tasks such as relation extraction, data mining, sentiment analysis, and many more. The task involves marking the negation cue and determining the negation scope. A negation cue can be a single word, multi-words, or an affix of a word. Words affected by the negation cue constitute the negation scope of a negated sentence. In this paper, we use the *SEM CD-SCO corpus to develop models to perform negation scope detection (Morante and Blanco, 2012). Prior to training the systems, we annotated a sample of the development data, and measured the inter-annotator agreement to assess the level of complexity of the annotation task.

This study compares a feature-based classifier and a transformer-based classifier by examining different aspects of both methods. In the first part, eighteen linguistic features were from the dataset, which belong to the categories of token-specific, constituency-based and dependency-based features. We utilized a Conditional Random Fields (CRF) classifier, which has frequently been used due to its effectiveness in modeling sequential relationships and context within textual data, demonstrating its capability in accurately capturing the complex patterns of negation scope in English sentences (Morante and Blanco, 2012). Inspired by the study of Khandelwal and Sawant (2020) on NegBERT, we fine-tuned the DistilBERT-base-uncased model to become applicable to negation scope resolution.

The BERT-based model achieved an F1-score of 0.925 on the token-level and 0.760 on the span-level, outperforming the CRF model by 0.358 on span-level and 0.182 on token-level. This significant improvement confirms the advantage of employing transformer-based models for negation scope detection.

The sections of this report are organized as follows: Section 2 provides a detailed task description, Section 3 discusses related work, Section 4 describes the dataset and pre-processing, Section 5 presents the experiments, Section 6 reveals the results, Section 7 performs an error analysis, and Section 8 concludes the study.

2 Task Description

The *SEM 2012 Shared Task described in Morante and Blanco (2012) is composed of three steps, which entails identifying the *negation cues*, the *negation scope* and the *negated event or property*. Negation could be defined as a grammatical construct that employs various mechanisms to alter the truth value of statements. Generally, the term cue expresses negation, scope refers to the portion of a statement that is being affected by the negation, whereas focus encapsulates the most prominently or explicitly negated element of the scope (Pullum and Huddleston, 2002). Identifying negation is an integral foundation for other NLP tasks, such as information extraction or sentiment analysis (Morante and Blanco, 2012).

The sentences below are examples, where the negation cues are marked in bold letters, the scopes are enclosed in square brackets and the negated events are underlined. The examples were derived from the annotated CD-SCO training corpus.

(1) [Some people] **without** [possessing genius] have a remarkable power of stimulating it.
(2) He declares that [he] heard cries but [is] **un**[able to state from what direction they came].
(3) [There is] **nothing** [more stimulating than a case where everything goes against you].

According to the annotation guidelines, the objective of negation scope annotation is to establish which words within a negated sentence are influenced by the change in polarity, irrelevant words should not be marked for scope. The annotators should aim to identify the longest relevant scope of the particular negation cue. Moreover, it should be noted that the scope may be discontinuous across the sentence. The authors suggest utilizing the paraphrase test *"It is not the case that ..."* on the sentence when marking the scope of negation cues (Morante and Daelemans, 2012).

Abu-Jbara and Radev (2012) provide insight into the importance of negation scope detection by explaining the significance of this task in case of English texts for various NLP applications, such as data mining, relation extraction, question answering, and sentiment analysis. They emphasize that ignoring negation may result in incorrect responses in question answering or incorrect predictions of sentiment in sentiment analysis systems. This highlights the importance of accurately identifying and understanding negation in texts, as it directly affects the performance and reliability of various NLP systems and applications.

2.1 Data annotation

The data annotation procedure of the *SEM CD-SCO corpus involves three steps: identifying negation cues, determining negation scopes, and specifying the negated event or property by following the annotation guidelines (Morante et al., 2011). The guidelines are presented in a 42-page document that provides detailed and complex instructions with examples for marking negation cues, negation scopes and negated events or properties. The authors of this study conducted negation scope annotation on a selected sample of sentences from the development data. The annotations were created in collaboration, following the rules outlined in the annotation guidelines (Morante et al., 2011).

Inter-annotator agreement (IAA) was assessed using two methods: token-based and exact span-based pairwise comparison between the annotations and gold labels as described in section 6.1. The results indicated a relatively high level of agreement, particularly at the token-level, with an F1-score of 0.895, and an F1-score of 0.877 at the span-level. The error analysis highlighted the specific challenges we encountered during the annotation process. Annotation errors were categorized into three main types: issues with discourse-level modifiers like 'yet', problems with elliptical structures, and complexities arising from intricate clause structures. Finally, we established that this particular task is more suited for expert annotators with background in linguistics than crowd annotators due its complexity and nuanced nature.

3 Related Work

3.1 Feature-based classification approaches

Several studies have addressed the challenge of detecting the scope of negation using supervised machine learning methods. Notable among these studies are those conducted by Morante and Blanco (2012), Abu-

Jbara and Radev (2012), Chowdhury (2012) and Lapponi et al. (2012). Each of these studies employs different sets of features, and this section will illustrate both common and distinctive characteristics of feature sets that have significantly contributed to the task.

Common features for this task include the part-of-speech (POS) tags and lemmas. POS tags are crucial in comprehending the grammatical role of words and the structure of a sentence, as well as in determining the parts of the sentence that may be affected by negation. Using lemmas, which represent the base form of words, helps standardize word variations and enables the recognition of patterns in negated sentences.

Token analysis is another feature commonly associated with this task. It involves examining every word or punctuation mark in a sentence, which can provide an understanding of the immediate lexical environment where negation cues exist. Specific words, such as coordinating conjunctions, or punctuation marks may signal the beginning or end of a negation scope.

The work of Abu-Jbara and Radev (2012) is unique in incorporating features, such as negation prefixes and suffixes, types of negation cues, and syntactic distance, to solve the task at hand. These features delve deeper into the language structure, identify specific elements indicative of negation, and compute the syntactic relationships between text elements. They can be particularly useful in identifying complex negation structures where the scope is not immediately apparent.

The work of Chowdhury (2012) considers phrasal and contextual cues and the interaction of negation cues with other parts of speech, such as noun phrases and verb phrases. This approach is necessary to place negation within a broader context, as its scope often extends beyond lexical indications to cover larger syntactic structures.

The work of Lapponi et al. (2012) deviates from previously introduced approaches by using dependency relations to explore the underlying syntactic structure that determines the scope of negation. The method analyzes the way words in a sentence interrelate with one another to determine how negation affects or is affected by their connection.

Thus, while each article includes common features such as the POS, tokens, and lemmas, most include unique features that might yield better model results. The combination of these features might, in turn, strengthen the model's prediction of the negation scope and increase the accuracy of Natural Language Processing. Table 1 below summarizes the implemented features and illustrates the different approaches in the feature selection.

	Abu-Jbara & Radev	**Chowdhury**	**Lapponi et al.**
POS Tags	X	X	X
Lemmas	X	X	X
Token Analysis	X	X	X
Negation Prefixes/Suffixes	X		
Syntactic Distance	X		
Phrasal/Contextual Clues		X	
Dependency Relations			X

Table 1: Features Used in Negation Scope Detection Studies.

3.2 Transformer-based classification approaches

In the work of Fancellu et al. (2018), neural models are developed and trained on cross-lingual corpus for negation detection tasks. Neural models have shown a great capability in negation detection works, even in cross-lingual circumstances. Their ensemble model, the BiLSTM + D-LSTM, performs with a recall score of 0.89, showing that an ensemble of neural networks mainly relies on POS tags, syntactic features and punctuation marks, which offers inspiration for pre-processing for negation detection.

The work of Britto and Khandelwal (2020) explores the application of transformer-based models, including BERT, XLNet and RoBERTa. The experiment is carried out on the BioScope Corpus and SFU Review Corpus. The task is split into two subtasks, namely cue detection and scope resolution. The tasks are performed on both negation detection and speculation detection. They used a joint model, XLNet and RoBERTa, together with BERT to carry out the task, and the former reported state-of-the-art results. The scores for the model per word of the input sentences are calculated. Two methods are utilized to make

a prediction for each word: averaging the output vectors of each token and taking an argmax over the result vector, or only considering the vector of the first token and retrieving the label by an argmax over the vector. For cue detection, the averaging method is applied, while for scope detection both methods are utilized. They obtained a gain of 3.61 F1 points on BF, 0.06 F1 points on BA, and 0.3 F1 points on SFU. Expanding the work of Khandelwal and Sawant (2020), the usefulness of transformer-based models is demonstrated in the field of speculation and negation detection and scope resolution, although it is domain-specific.

In the recent work of Truong et al. (2022), a new strategy of negation-focused pre-training is proposed to better incorporate negation detection tasks across domains. NegBERT from the work of Khandelwal and Sawant (2020) is applied as baseline and by extensive experiments, its transfer-learning capability is proved. A novel negation-specific masking strategy is utilized in their experiment. First, they applied the standard random word masking strategy in model training, then the negation cues were masked to incorporate the information about negation patterns in the model. Below is an example illustrating their sentence tokenization strategy:

No serious complications such as hypertension, diabetes.

***[CUE]** serious complications such as **[MASK]**, diabetes.*

Under this masking scheme, they introduced a new type of token, **[CUE]**. To reconstruct the sentence, the model needed to predict correctly both the **[CUE]** to be *No*, and the **[MASK]** to be *hypertension*. By replacing BERT with RoBERTa and applying the whole-word-masking strategy, the model's ability to distinguish between different types of tokens is enhanced. Three benchmarks for negation detection are all used, namely the BioScope, the Sherlock database, and the SFU Review Corpus. To introduce more data, the shorter, more informal subset of VetCompass UK was also used (Cheng et al., 2017). Gains are observed across all datasets and are more noticeable over the biomedical datasets, showing that the negation-focused pre-training strategy is effective for enhancing the performance of models utilizing transfer-learning on negation detection tasks.

4 Dataset Description

The *SEM CD-SCO annotated corpus for negation scope detection is composed of stories by Conan Doyle. The training data is composed of chapters 1-14 from *The Hound of the Baskervilles*, the development data contains *The Adventure of Wisteria Lodge*, and the test data combines *The Adventure of the Red Circle* and *The Adventure of the Cardboard Box*. The annotations indicate negation cues, negation scope, and the negated events or properties (Morante and Blanco, 2012).

Chapter	Sentence	Token	Word	Lemma	POS	Parse Tree	Cue	Scope	N. Event/Property
wisteria01	60	0	Our	Our	PRP$	(S(NP*	_	_	_
wisteria01	60	1	client	client	NN	*)	_	_	_
wisteria01	60	2	looked	look	VBD	(VP*	_	_	_
wisteria01	60	3	down	down	RB	(ADVP*)	_	_	_
wisteria01	60	4	with	with	IN	(PP*	_	_	_
wisteria01	60	5	a	a	DT	(NP(NP*	_	_	_
wisteria01	60	6	rueful	rueful	JJ	*	_	_	_
wisteria01	60	7	face	face	NN	*)	_	_	_
wisteria01	60	8	at	at	IN	(PP*	_	_	_
wisteria01	60	9	his	his	PRP$	(NP*	_	his	_
wisteria01	60	10	own	own	JJ	*	_	own	_
wisteria01	60	11	unconventional	unconventional	JJ	*	un	conventional	conventional
wisteria01	60	12	appearance	appearance	NN	*))))))))	_	appearance	_
wisteria01	60	13	.	.	.	*)	_	_	_

Table 2: Example Sentence from the *SEM CD-SCO Development Set

The data are provided using the CoNLL format, whereby a new line is used to introduce each token separately in a given sentence. Empty lines indicate the end of a sentence. Each row in the data contains a minimum of 8 columns providing information about chapter name (1), sentence number (2), token number (3), word (4), lemma (5), part-of-speech (6), parse tree information (7) and negation cues if any (8). For non-negated sentences, the final column (8) contains the symbol '***' and there are no additional columns. If the sentence is negated, however, the information about the negation is organized

into three columns indicating the negation cue, negation scope (9), and negated events or properties (10). In case of two or more negations within the same sentence, an additional three columns per negation are added to provide details about the negations. Negation cues are excluded from the negation scope except for cases when negation is expressed by affixes (e.g. *'im-'*, *'-less'*). Moreover, both negation cue and negation scope can be discontinuous within the sentence (Morante and Blanco, 2012). Table 2 illustrates the data structure. For our experiments, the models were trained on the training set, model fine-tuning was carried out on the development set, and lastly, the best-performing models were evaluated on the test set.

4.1 Pre-processing the corpus

Pre-processing aimed to transform the original data into a better-structured and manageable format. Moreover, our objective was to extract and organize information related to negation cues, scopes, and focuses and prepare the documents for feature extraction.

During pre-processing the original CoNLL formatted TXT files were reformatted into TSV (Tab-Separated Values) files. In case a sentence contains multiple negations, each negation was treated individually by duplicating the sentence according to the number of negations. This means that if a sentence entailed two negations, it resulted in two copies of the sentence in the processed data. Each duplicate provides information about the respective negation cue, negation scope, and negation focus within the sentence. The sentence IDs in column 2 were also adjusted to reflect if the negated sentence is the first, second, or third duplicate. In the next step, the number of columns in each row was also reduced to 10. Column 9 indicating which tokens are in scope, was overwritten by introducing two labels: 'OS' for out-of-scope tokens and 'IS' for in-scope tokens. Column 7, which contains information about the syntactic constituency for each token, was also adjusted: the asterisks were replaced by the corresponding tokens and a preceding blank space. This step was required to later parse the constituency trees using the NLTK library and extract constituency features. We utilised SpaCy's dependency parser[1] with the 'en-core-web-sm'[2] pipeline in order to extract sentence dependency features. In case of our feature-based model, we utilised eighteen features, eleven of these were represented in additional columns containing feature values for each token.

The original CD-SCO corpus included two separate files for test data. To streamline the analysis and ensure consistency, these files were combined into a single TSV file. Statistical information about the corpus before and after pre-processing is in Table 3 (Morante and Blanco, 2012). Pre-processing resulted in an increase in the number of tokens and negated sentences due to the duplication sentences with multiple negation cues. This also led to an increase in the percentage of negated sentences in the data. Notably, the number of negation cues exceeds the number of negation scopes, the reason for this is that in such sentences the clause containing the cue usually comprises only of the cue itself. An example for this is the sentence *"**No**, sir; I believe you mean well by me.*[3]*"* without a scope in the development set.

	Training	**Dev.**	**Test**	**P. Training**	**P. Dev.**	**P. Test**
# Tokens	65,450	13,566	19,216	69,218	14,272	19,915
# Sentences	3644	787	1089	3780	816	1118
# Negated Sentences	848	144	235	984	173	264
% Negated Sentences	23.27%	18.29%	21.57%	26.03%	21.20%	23.61%
# Negation Cues	984	173	264	1003	179	274
# Unique Cues	30	20	20	30	20	20
# Scopes	848	144	235	887	168	249

Table 3: CD-SCO Corpus Statistics before (Columns 2-4) and after pre-processing (Columns 5-7).

[1] https://spacy.io/api/dependencyparser

[2] https://spacy.io/models/en

[3] sentence ID: 'wisteria02_155_1'. Other examples in the development set: 'wisteria01_223_1', 'wisteria01_295_1'.

5 Experiments

5.1 Feature-based classification experiments

Choice of model

Negation scope detection is a sequence labeling task since the objective is to decide if each token within a given sentence belongs to the negation scope or not and the relationship between those tokens is essential for this task. Several machine learning models have been proposed for negation scope detection, among which Conditional Random Fields (CRFs) have been frequently implemented (Morante and Blanco (2012), Abu-Jbara and Radev (2012), Lapponi et al. (2012)). For this task, we decided to employ CRFs as the main model due to their effectiveness at modeling sequential dependencies.

CRFs are a type of probabilistic graphical model capable of capturing the sequential characteristics of data. They have been frequently used for sequence labeling NLP tasks, such as part-of-speech tagging and Named Entity Recognition (Lafferty et al., 2001). In this particular context, CRFs can efficiently model the connections between negation cues and scopes, while taking contextual information from other words in the sentence also in consideration. Consequently, the model can decide if negation cues affect only a few surrounding words or the complete sentence clause. Due to their ability to capture long-range dependencies between tokens and handle a wide range of features, CRFs are suitable for negation scope detection.

Selected Features

The selected features for the negation scope detection task include token-specific, constituency-related, and dependency-related features. The following section will briefly describe the selected features and their relevance to this particular task.

Token-specific features	Constituency-based features	Dependency-based features
Token	Distance to first cue	Dependency relation
Lemma	Same clause as cue (binary)	Dependency head
POS window (previous, current, next)	Same phrase as cue (binary)	Distance to the root
Lexicalized POS		Distance to the cue
Negation cue information		
Token is punctuation (binary)		
Token is a negation cue (binary)		
Token position (beginning / in / end)		
Physical distance to nearest cue		

Table 4: Overview of Features for CRF Models.

a. Token-specific features:

These features include the token, lemma, previous and following POS, lexicalized POS, negation cue, binary feature for punctuation, binary feature for being a negation cue, the token's position within the sentence (start / inside / end), and the token's physical distance to the nearest cue. Including the token itself might be useful because it allows the model to capture raw lexical information. The use of lemma as a feature is motivated by the need to generalize across different inflections and word forms, making it easier to identify negation scope regardless of inflectional changes. The choice of lexicalized POS as a feature is motivated by its ability to capture the grammatical role of a specific token (Lapponi et al., 2012). The grammatical role added with the token might give more contextual information. For instance, the token '*saw*' can act as a noun and verb on its own, this information can be encoded with the added POS tag.

The binary feature for marking punctuation tokens is included because punctuation marks often serve as indicators for sentence boundaries or clause divisions. Identifying punctuation helps in understanding the structure of the sentence, which is crucial for negation scope detection (Abu-Jbara and Radev, 2012). This study also applied a binary feature to indicate if the token is part of the negation cue (Abu-Jbara

and Radev, 2012). This may impact related features and thereby allow the model to identify the correct scope range. Certain kinds of negation cues, such as '*im-*' and '*un-*' prefixes, only have a small negation scope surrounding the cue. The token's position within the sentence can be useful for the model since tokens at the start and end of the sentence are often not part of the scope. Lastly, the token distance to the cue measures the distance of a particular token to the nearest negation cue (Abu-Jbara and Radev, 2012). Since each sentence containing multiple negations is duplicated, the tokens in the scope are correctly aligned with the referred negation cues. Measuring this distance is essential for determining the scope of negation because tokens closer to the cue are more likely to be within the scope, while those farther away are less likely.

b. Constituency-based features:
These features were extracted using the parse tree information provided in the data in column 7. The NLTK library was utilized to reconstruct the parse tree for each sentence. The features include constituency distance between the token and the first negation cue, a binary feature to validate if the token is in the same clause as the first negation cue, and another binary feature to validate if the token is in the same phrase as the first negation cue (Abu-Jbara and Radev, 2012). To extract these features, first, the position of the negation cue and the token within the tree is determined. Then, the lowest 'S' ancestor, in other words, the nearest sentence clause ('S') for a given token and the negation cue is determined. Lastly, finding the immediate parent (phrase) of the token and the negation cue can determine whether they are within the same phrase or clause. We assume these features will be useful because constituency distance provides information about how tokens are connected within the sentence's syntactic structure. Tokens with a shorter syntactic distance to the negation cue are more likely to be within the scope. The clause and phrase membership features can help identify tokens that are syntactically related to the cue, which is crucial information because tokens within the same phrase or clause as the negation cue are more likely to share a scope.

c. Dependency-based features:
These features include the type of dependency relation, the dependency head of the token, the dependency distance to the root, and that to the cue (Lapponi et al., 2012). Both constituency and dependency features give us information about the structural aspects of a sentence, while dependency features offer a particularly more detailed perspective on the relationships between individual words within the structure. These features were extracted using the spaCy library, supported with custom functions for processing. A key to our method was the handling of hyphenated words present in the data, which we addressed with a custom spaCy pipeline component named 'merge_hyphenated_compounds'. This component merges hyphenated compounds into single tokens, contributing to accurate analysis. We identified hyphenated words using the regular expression pattern within this function. Before analyzing sentences with spaCy, we also implemented additional pre-processing of the input text, which includes normalizing quotation marks and handling special tokens to streamline processing.

During the extraction of dependency features, we incorporated NetworkX to create a graph to represent token dependencies, which enables us to calculate the shortest path between tokens and provide the model an insight into the inter-token relationships. This method was implemented to determine the dependency distance to the root of the sentence and to the first negation cue, if present. The model can utilize this information to establish which tokens are closer to the core of the sentence and to the negation cue and, therefore more likely to be part of the scope. Usually, a shorter dependency distance suggests a stronger syntactic connection, which can be informative in determining the scope.

Alongside these distances, we also extracted the dependency relation and the head of each token. Certain dependency relations can indicate the scope of the negation cue clearly to the model, for instance, this feature can indicate if the cue modifies a verb, an adjective, or an adverb. The dependency head of a token indicates hierarchical relationships in the dependency tree, which is crucial for understanding the extent of the negation cue's impact on different levels of the sentence. In case of sentences with multiple clauses for example, the dependency head feature can help the model accurately assign the scope to the group of impacted tokens.

Vectorization
In contrast to models that require dense vector-based representations like word embeddings, the CRF model utilized its final features in their original, categorical form. This method is referred to as sparse vectorization. Rather than forming dense vectors, each feature is presented in its dimension within a high-dimensional space, which indicates the presence or absence of that feature. This approach is particularly well-suited for CRF models because they are proficient at handling and capturing the relationships between a wide range of discrete features in sequence labeling tasks. Sparse vectorization enables the model to directly leverage the interpretability and specificity of each feature, which is especially valuable for nuanced tasks such as negation scope detection.

Feature Ablation Study
In order to understand the individual contribution of each feature to the CRFs model's performance, we conducted feature ablation on the best-performing model. Initially, we run the model with all features to establish a baseline token-level F1-score. In the next step, we iteratively removed a single feature at a time, observing the change in the resulting F1-score. We then removed the feature with the largest negative impact on the F1-score, which became a new baseline. This process was repeated until all remaining features exhibited positive contributions to the F1-score, retaining only those features that enhanced the overall performance. In total, three iterations were carried out, resulting in the removal of two features: 'is_negation_cue' and 'same_phrase', while keeping the remaining sixteen features. Table 5 below provides a breakdown of this process.

The feature 'is_negation_cue' assigns a binary value to each token indicating whether it constitutes a negation cue or not. This feature may be redundant, as the 'negation_cue' feature already captures this information. The feature 'same_phrase_as_cue' is a constituency-based binary feature that assigns a value to each token based on its membership in the same phrase as the first negation cue, if present. While employing these sixteen features resulted in a higher token-level F1-score of 0.74, it also led to a decline in the span-based F1-score to 0.36. Therefore, for the final evaluation on the test set, we opted to utilize all eighteen features and remove none. Additionally, the results highlight that the features 'next_token_pos', 'negation_cue', 'token_distance_to_nearest_cue' and 'dependency_head' strongly contribute to the model's comparatively high performance.

Feature	**Baseline F1: 0.73**	**Baseline F1: 0.74**	**Baseline F1: 0.74**
token	-0.01	-0.001	0.011
lemma	-0.01	-0.003	0.006
pos	0.003	-0.001	0.008
lexicalized_pos	-0.002	-0.006	0.006
previous_token_pos	0.008	0.006	0.013
next_token_pos	0.041	0.043	0.055
negation_cue	0.055	0.065	0.084
constituency_distance_to_cue	0.014	0.008	0.009
same_clause_as_cue	0.012	0.009	0.016
same_phrase_as_cue	-0.002	-0.009	removed
is_punctuation	0.005	0.007	0.006
token's_sentence_position	-0.004	0.001	0.009
is_negation_cue	-0.004	removed	removed
token_distance_to_nearest_cue	0.038	0.047	0.053
dependency_type	0.004	0.011	0.006
dependency_head	0.046	0.043	0.041
dependency_distance_to_root	0.005	0.0	0.007
dependency_distance_to_cue	0.01	0.004	0.021

Table 5: Feature Ablation Results for Negation Scope Detection with All Features.

Hyper-parameter tuning
To optimize L1 and L2 regularization, this study employed a grid search method to find the ideal hyper-parameters 'c1' and 'c2' for the third CRF model utilizing eighteen features. This involved

systematically varying the values of 'c1' and 'c2' over a predefined parameter grid[4] and evaluating the model's performance for each combination of these parameters. The aim was to strike a good balance between precision and recall, and this was measured using the F1-Score on the validation set. The study resulted in slightly adjusted hyper-parameter settings to maximize this score. The best hyper-parameters identified were c1 = 0.1 and c2 = 0.01, achieving a token-level F1-Score of 0.735 and a span-level F1-score of 0.384. These scores indicate a slightly higher level of model performance under these specific parameter settings.

Problems and Conclusions

During the feature extraction process, the dependency parsing features brought up certain challenges. Since SpaCy works better with sentences as input, the tokens were initially transformed into doc files, including the sentences of the training dataset. However, after analyzing the output, it was observed that SpaCy retokenizes the sentences in a different manner than the original dataset. The main differences include the hyphenation, ordinal numbers, quotation marks, and punctuation. The original dataset takes one token for hyphenated words ('*heart-rug*'), and SpaCy separates these into three different tokens ('*heart*', '-', '*rug*'). The same goes for ordinal numbers. Numbers in the original file are indicated as one token, such as '*1.*', but the SpaCy tokenizer splits these in two. Furthermore, the curly quotation marks had to be straightened, but even then, random punctuation marks threw the progress off the track.

As a remedy to these problems, we decided to implement a different approach to extracting dependency-related features. To address the issue of SpaCy's tokenization methods, which are different from the dataset, a custom SpaCy pipeline component was created to merge hyphenated compounds into single tokens. This function allows us to align SpaCy's tokenization closer to the dataset. We preprocessed the data to normalize non-standard punctuation marks and handle the special cases in the input, such as *1., '86, No.*, which hindered the parsing process. For each sentence, a network graph of token dependencies was created using NetworkX, which was then used to extract the dependency features. To manage the large size of the datasets and prevent memory loss errors, the script processes sentences individually, which allows for a reduction in memory usage and allows for the identification of problematic patterns causing blockages. The processed data, including the lexical, constituency-based, and dependency-based features was transformed into a TSV format using pandas. Empty lines were dropped after feature extraction since the unique sentence IDs and sentence position features indicate which tokens belong to the same sentence. Based on our experiments, we may conclude that the CRF model reaches the most optimal performance with the combination of lexical, constituency-based, and dependency-based features.

5.2 Transformer-based Classification Experiment

BERT model Description

This report considered the use of DistilBERT-base-uncased for the task of negation scope resolution. We were interested in testing whether a feature-based or transformer-based model could perform the task with higher results. According to Sanh et al. (2019), DistilBERT is simply a lighter version of the BERT model proposed to increase efficiency in NLP tasks. DistilBERT was developed by applying a technique known as *knowledge distillation* to the BERT model, which involves training a smaller model to replicate the behavior of a larger, previously trained model. The term "base" in DistilBERT-base indicates the model's size. DistilBERT has six transformer layers, which is half of the number of layers in the BERT-base model. "Uncased" suggests that the model does not differentiate between lower- and uppercase tokens. This innovation is a solution to the significant computational requirements of the original BERT model by Devlin et al. (2018), which allows more accessible applications for various downstream tasks.

DistilBERT was pre-trained on the same corpus as BERT, which is a combination of English Wikipedia and Toronto Book Corpus. However, it underwent pretraining in a self-supervised manner. Since this approach does not use human-annotated labels, a wide variation and large volume of publicly

[4]parameter grid = 'c1': [0.01, 0.1, 1, 10], 'c2': [0.01, 0.1, 1, 10]

available data can be utilized. In this process, input and labels are automatically created by employing the BERT base model. To mimic the BERT base model, DistilBERT was trained with the Masked Language Modeling (MLM) objective, where some words are hidden in the input sentences, and the model predicts them. Furthermore, it focused on making its internal language understanding very close to the original BERT model by training the model to return the same probabilities and generate hidden states similar to the BERT base model. Even though DistilBERT is 40% smaller than BERT, making it faster and lighter, it still understands language almost as well as BERT (Sanh et al., 2019).

Pipeline Description

In our approach, we marked negation cues in the input sentences before feeding them to the fine-tuned DistilBERT model, drawing inspiration from the augmentation technique introduced in Khandelwal and Sawant (2020). Based on our observations of the CRF model, we assume that information about the negation cue itself can be useful for the model since certain negation cues tend to have a certain range of negation scopes. The cue *without* for example, usually takes a narrower negation scope compared to the cue *not*, which might negate an entire clause or sentence. Therefore, we decided to preserve the original negation cues and add a special token immediately before the cue, thus ensuring that the model had access to the cue itself. By encoding the cue marking tokens, we aim to guide the model's attention towards the words and phrases expressing negation, which can enhance the model's performance in predicting the scope. This can be especially useful in case of duplicated sentences containing multiple negation cues, allowing the model to learn the mappings between the specific cue and its scope. The various negation cue types were handled differently in this step:

1. If the sentence only contains a single-word negation cue, e.g. *not, without*, the cue token will be replaced with CUE_0 + token.
2. If the sentence only contains an affixal negation cue, e.g. *im-, un-, dis-, -less*, the cue token will be replaced with CUE_1 + token.
3. If the sentence contains multi-word negation cues, e.g. *by no means, neither ... nor*, the cue tokens will be replaced with CUE_2 + token.

The BERT template we used was designed for the task of Named Entity Recognition (NER), which does not require such modifications. In NER, there is no need to signify a particular token specially in contrast to our model. This additional modification helped the system identify scopes more effectively and required its post-processing accordingly.

Fine-tuning Adaptations

We adapted a BERT fine-tuning template designed for NER for our Negation Scope Detection task. Both tasks use data presented in CoNLL format and aim to detect sequential patterns. The NER task identifies Named Entities with BIO tags, while the negation scope identifies a sequence of tokens affected by the negation cue. We applied certain modifications to adapt the code to our task. To begin with, the provided code template for BERT fine-tuning was initially designed to load its dataset directly from the HuggingFace hub. However, in our case, the data was not readily available in the repository. As a result, we needed to make certain modifications to the code template to accommodate our locally stored data in TSV format.

Loading datasets from HuggingFace with the datasets module automatically represents the dataset in a particular format. To represent our data in the same 'DatasetDict' format, which includes training, development, and test data as sub-dictionaries, we initially transformed our data into pandas data frames. This was useful to represent them later with the 'Datasets' module utilizing the 'from_pandas' method. Before calling the data in 'Datasets', we needed to represent the specific content in data in the appropriate dataset formatting, particularly defining the columns as features in a sequential format. Additionally, we defined our negation scope label set, which corresponds to the NER labels in the original template. It is used as the main reference point for our predictions and evaluations.

The fine-tuning process required adjusting certain parameters to work best for the convergence of the model. We empirically assessed the relationship between batch size and learning rate parameters as seen in Table 6. Batch size sets the amount of input that we feed at once, and the learning rate determines the step size at which the weights are updated during the training process. If it is too large, which means the model takes bigger steps each time, it may overshoot the minimum of the loss function, whereas if it is too small, it would either take too much time to converge or it would settle in an undesired minimum (Jurafsky and Martin, 2022). We experimented with various learning rates ranging from 1e-6 to 1e-3, batch sizes of 8, 16, 24, and 32, and were able to get the best generalization from batch size of 16 and learning rate of 1e-4. Most of these experiments, which are around 25 runs, were done with 2 epochs due to material limitations. Our results further improved when we finally ran our best-performing batch size - learning rate match for 4 epochs. The results of these batch-size versus learning rate experiments as displayed in Table 6.

Learning R. ↓ \ Batch S. →	**8**	**16**	**24**	**Metric**
2e-5	0.871	0.873	0.861	token-f1
	0.663	0.651	0.545	span-f1
1e-4	0.889	0.903	0.877	token-f1
	0.678	0.728	0.623	span-f1
2e-4	0.867	0.869	0.891	token-f1
	0.690	0.631	0.682	span-f1

Table 6: Learning Rate - Batch Size Matrix using 2 epochs

The format of the input and its specific representation tailored for BERT can be seen below. Since our data is in CoNLL format, the model can easily represent each sentence separately as a list of words. The tokenizer parses these words into a tokenized version using a subword tokenization technique. The function for aligning the labels with tokens is modified in a way that it calculates the longest input sentence in the dataset and adds padding to each input sentence accordingly. These pad tokens are not included in the attention set, nor assigned with any label, therefore they are excluded as input for the training of the model. The last format of the input used in training consists of the tokenized sentence representation with token IDs, attention masks, and corresponding labels per token. In our task, attention values are 1 for each token because we want to process the whole sentence at once to identify a sequential pattern for negation scope. The system output consists of assigned probabilities per each token acquired via applying softmax function. Each subarray represents the probability of labels for their corresponding token.

Tokens of the first sentence in the development data:

```
['1.', 'The', 'Singular', 'Experience', 'of', 'Mr.', 'John', 'Scott', 'Eccles']
```

Sentence processed by the tokenizer:

```
['[CLS]', '1', '.', 'the', 'singular', 'experience', 'of', 'mr', '.', 'john',
'scott', 'ec', '##cles', '[SEP]', '[PAD]', '[PAD]', '[PAD]', '[PAD]', '[PAD]',
'[PAD]']
```

Tokenized sentence with token IDs and corresponding labels:

```
'input_ids':  [[101, 1015, 1012, 1996, 13048, 3325, 1997, 2720, 1012, 2198, 3660,
14925, 18954, 102, 0, 0, 0, 0, 0, 0]], 'attention_mask':  [[1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0]], 'labels':  [[-100, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, -100, -100, -100, -100, -100, -100, -100]]
```

DistilBERT output:

```
[[2.400e-03, 9.976e-01], [2.000e-03, 9.980e-01], [6.000e-04, 9.994e-01],
[3.000e-04, 9.997e-01], [3.000e-04, 9.997e-01], [3.000e-04, 9.997e-01],
[3.000e-04, 9.997e-01], [4.000e-04, 9.996e-01], [3.000e-04, 9.997e-01], [4.000e-04,
9.996e-01],....
```

Tokenization and Post-processing

"PreTrainedTokenizerFast" was implemented for the pre-processing steps. This tokenizer allowed tokens to be numerically represented with their unique input IDs, including special CLS and SEP tokens.

Notably, this tokenizer also split some words into multiple subtokens by making use of WordPiece subtokenizer[5].

The tokenized examples were aligned with their respective labels, making them ready for the fine-tuning process. After training our model, our results were exported into a prediction file which enabled us to calculate our span-based and token-based metrics in particular ways that we defined earlier. However, to accurately assess the model's performance, we needed to undertake extensive post-processing steps due to the subword tokenization mechanism employed by the large language model.
During post-processing we had to solve the following issues among others:

1. The added tokens *CUE_0, CUE_1, CUE_2* indicating negation cues had to be removed from the output while retaining the original negation cue tokens within the sentence.
2. Subtokens, such as *super, ##sti, ##tion* had to be rejoined into a single token.
3. Verb contractions, such as *'ve, 'm, 're, 'll*, were divided and had to be rejoined into a single token.
4. Common abbreviations, such as *Mr., Dr., Mrs., Co.*, were divided and had to be rejoined into a single token.
5. Name initials, such as *J.P.*, were divided and had to be rejoined into a single token.
6. Tokens with internal quotation marks, such as *n't, o'clock* were divided and had to be rejoined into a single token.
7. Hyphenated compound words, such as *fair-sized, grass-grown, tete-a-tete* were divided and had to be rejoined into a single token.
8. Serial numbers, such as *1.*, were divided and had to be rejoined into a single token.
9. Apostrophe-number splits, such as *'86*, were divided, and the apostrophe needed to be rejoined with the digits.
10. Double quotation marks and double dashes were divided into single ones and had to be rejoined into a single token.

To address these issues, our post-processing implementation involved a series of conditional checks and manipulations on the affected tokens. The outcome of post-processing was a new TSV file with accurate representation of tokens in each sentence, restoring the original sentence structure as present in the input file for BERT. This rigorous implementation was essential to be able to accurately compare the performance of the fine-tuned BERT model with our feature-based results.

Problems and Conclusions
Initially, we assumed that only post-processing requirement would be handling subword tokenization instances such as 'en ##li ##ven'. However, upon examining the output file, we discovered that this issue also applied to a wider range of cases as listed above. It would have been impractical to compare the input and output files sentence by sentence, thus we employed a method that compared the number of tokens present for each pair of sentences using the unique sentence IDs. Based on inspecting the cases where the number of tokens was more or less than expected in the sentence, we established generic rules to address the specific token separations that required restoration. This is proved as a useful tool to address the subword tokenization problem.

In conclusion, this study's exploration of DistilBERT for negation scope resolution offered some insightful outcomes. Despite its smaller size, DistilBERT performed with impressive metrics. It outperformed the CRF model on the specific parameters this report illustrated, see section 6.3. Our custom fine-tuning approach attempted to find the best parameters, which led to a preferred batch size of 16 and a learning rate of 1e-4 on four epochs. This resulted in a noticeable boost in accuracy, bringing higher results than the CRF model in key metrics. However, it must be noted that these parameters work the best for this particular dataset, so more metric insights into the test set will confirm or deny the generalization capacity of this model.

6 Results

6.1 Evaluation Metrics

In this section, we perform quantitative analyses to compare the feature-based and transformer-based models when applied to the task of negation scope detection. The models will be evaluated by comparing

[5]https://huggingface.co/docs/transformers/main_classes/tokenizer

their output of negation scope labels ('IS' in scope or 'OS' out of scope) against gold labels for the development set in sections 6.2 and 6.3 and for the test set in section 6.4. The evaluation will be carried out using two different matrices: token-based and exact span-based pairwise comparison.
For token-based evaluation, the following rules were followed:

- TP: Both gold and predicted label are inside the negation scope 'IS'.
- TN: Both gold and predicted label are outside any negation scope 'OS'.
- FP: System incorrectly predicted the token as 'IS' while the gold label is 'OS'.
- FN: System incorrectly predicted the token as 'OS' while the gold label is 'IS'.

For exact span-based evaluation, the following rules were followed:

- TP: The sentence has a negation scope in gold and the entire negation scope (all tokens) identified by the system matches the gold data.
- TN: The sentence is not negated and the system labels and gold labels agree on the absence of scope.
- FP: System incorrectly identified a negation scope but the gold has no scope at all.
- FN: (1) Gold has a scope but the system predicted no scope at all. (2) Partial matches where the predicted span is included in the gold span.
- FN and FP: Incorrect matches where some labels predicted by the system are correct and some are not.

These rules were adapted from the evaluation guidelines described in the paper introducing the *SEM 2012 Shard Task (Morante and Blanco, 2012). Some might say that the way false positives and false negatives were defined might penalize incorrect matches too strongly, however, we decided to adhere to these guidelines. We applied the same token-level and span-level rules described above for all the feature-based and transformer-based models.

6.2 Feature-based Classification Results on Development Set

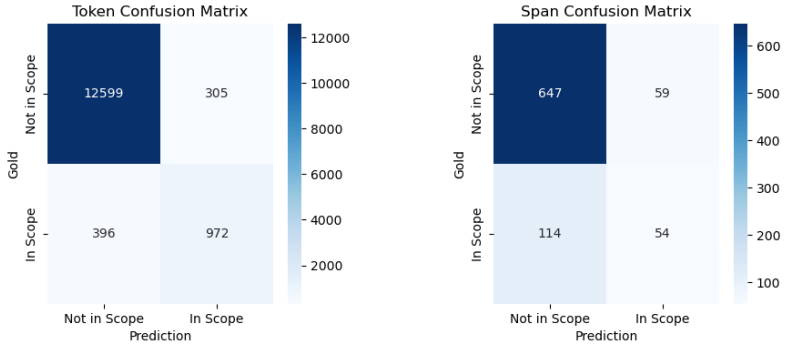
The evaluation results in Table 7 indicate the feature-based CRF models' performance in terms of precision, recall and F1-score in identifying negation scopes. The first model solely relies on the combination of token-specific and constituency-based features, the second model utilizes the combination of token-specific and dependency-based features, while the third model utilizes features of all three kinds. The results in the table illustrate that the model combining eighteen features of all three categories (Model 3) with optimized hyper-parameters[6] reaches the highest results across all metrics with a token-level F1-score of 0.735 and a span-level F1-score of 0.384. It can be observed that the scores are comparatively lower for span-based evaluation than for token-based evaluation. This difference is caused by the stricter requirements for identifying the complete negation span correctly in case of the span-level approach. In case of both types of evaluation, the model reached a higher precision than recall, the precision indicating that the majority of the labels produced by the model were correct. The lower recall suggests, however, that the model failed to correctly identify a large number of tokens.

	Model 1		**Model 2**		**Model 3**	
	Token-level	**Span-level**	**Token-level**	**Span-level**	**Token-level**	**Span-level**
Precision	0.746	0.451	0.749	0.391	0.761	0.478
Recall	0.621	0.274	0.700	0.256	0.711	0.321
F1-score	0.678	0.341	0.724	0.309	0.735	0.384

Table 7: Evaluation Metrics for CRF Models on Development Set.

[6] c1 = 0.1, c2 = 0.01

We created two confusion matrices to understand the degree of agreement between the gold labels and the predicted labels by Model 3. The high number of false positives and false negatives highlights further need for improvement in feature engineering, particularly to reduce the number of missed negation scopes and partial or incorrect scope matches.



(a) Token-level (b) Span-level

Figure 1: Confusion Matrices using CRF Model 3 on Development Set.

6.3 Transformer-based Classification Results on Development Set

In comparison with the best-performing CRF model utilizing all eighteen features, the BERT model with optimized parameter settings[7] reaches higher token-level and span-level F1-scores as illustrated in Table 8. To be more specific, the token-level F1-score shows an increase of 0.17, and the span-level F1-score a larger increase of 0.35. It is worth noting that in the case of the feature-based model, there is a difference of 0.35 between token-level and span-level F1-scores, whereas for the transformer-based this value difference is about half as large, only 0.17. The considerable performance improvement observed in the fine-tuned BERT model may be attributed to the transformer architecture, which allows for higher generalization capacity over sequential data in unseen text. On the other hand, the CRF model relies on a set of selected features resulting from feature engineering efforts.

It is important to acknowledge that experimenting with a larger selection of features and adjusting hyper-parameters is likely to have a direct impact on these metrics. Moreover, when utilizing transformer-based models for NLP tasks, one should also consider the factor of random weight initialization. This means that when using the same dataset, identical parameters, and hardware, the outcomes might still exhibit variations. We did not set a specific random seed for reproducibility in our experiments, thus it should be acknowledged that the results may vary in each run due to the inherent randomness.

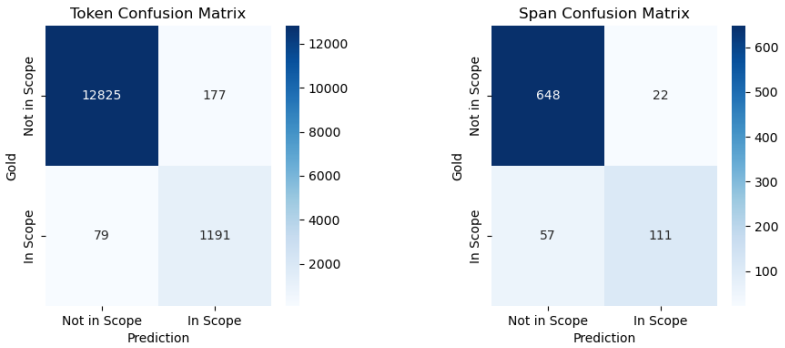
	CRF		**BERT**	
	Token-level	**Span-level**	**Token-level**	**Span-level**
Precision	0.761	0.478	0.871	0.835
Recall	0.711	0.321	0.938	0.661
F1-score	0.735	0.384	0.903	0.738

Table 8: Comparison of CRF Model 3 and DistilBERT model on Development Set.

We created confusion matrices to further inspect the performance of the BERT-based model for the task of negation scope detection. When comparing the token-level matrices of the feature-based and transformer-based model, we notice a significant decrease in the number of false negatives and false positives in the predicted labels. For example, compared to the CRF model, which failed to identify 396 tokens as in scope, the BERT model only missed 79 tokens. The span-based confusion matrices also

[7]Batch size: 16, Learning rate: 1e-4, Epochs: 4

show us that the number of cases where the model missed to identify a span has significantly decreased, and the cases where the model correctly identified the spans have increased to 111.



(a) Token-level (b) Span-level

Figure 2: Confusion Matrices using the Fine-tuned DistilBERT model on Development Set.

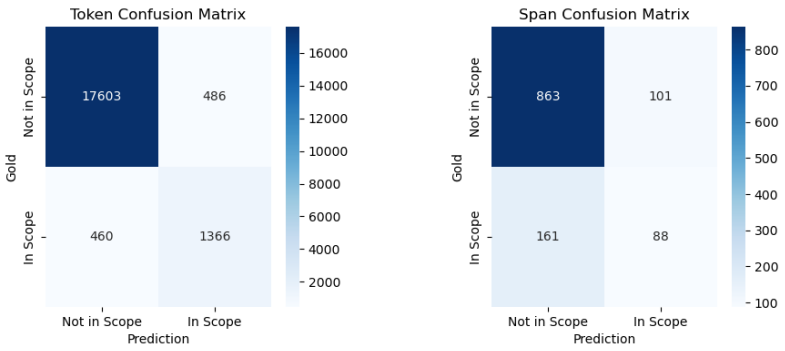
6.4 Results on Test Set

This section aims to compare the performance of the optimized CRF model with the fine-tuned DistilBERT model, focusing on their performance on the combined test data. Similarly to the results observed in the development data, the transformer-based model brings higher results than the CRF model considering both the token-level and the span-level evaluation as shown in Table 9. It exhibits an increase of 0.183 points at the token-level and a 0.358 points at span-level F1-score score, showcasing state-of-the-art results for the task of negation scope detection. A noteworthy observation is that both models perform slightly better on the test set compared to the development set, which could be due to closer similarity between training and test set. Although these results are promising, one should note that the training, development and test data belong to the same genre and style. We could only draw generic conclusions if the models were also tested on a more diverse array of annotated texts to validate these findings.

	CRF		**BERT**	
	Token-level	**Span-level**	**Token-level**	**Span-level**
Precision	0.738	0.466	0.895	0.822
Recall	0.748	0.353	0.957	0.707
F1-score	0.743	0.402	0.925	0.760

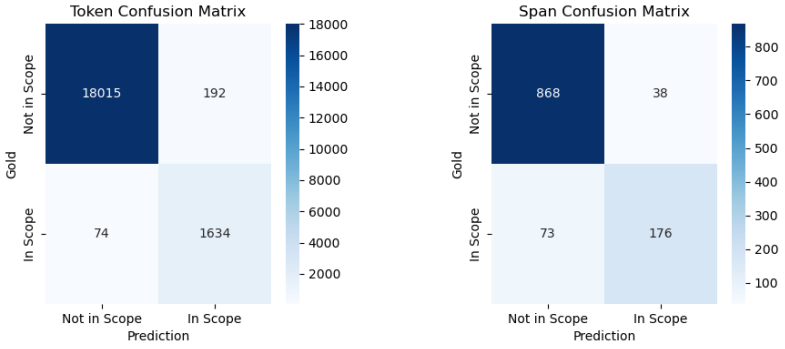
Table 9: Comparison of CRF Model 3 and DistilBERT model on Test Set.

The confusion matrices, as depicted in Figure 3 and Figure 4 showcase the results of the two best-performing models on the test set. The token-level analysis highlights that the CRF model produces a considerably higher combined total of false positives and false negatives (946 instances) compared to the DistilBERT model (266 instances). At the same time, the number of true positives increased by 268 instances with the DistilBERT model. Observing the span-level evaluation results, the CRF model's effectiveness is modest, correctly identifying 88 spans where negation scopes were present. On the other hand, the transformer-based model correctly identifies each token as in scope or out of scope in 176 negated instances. The subsequent section we further analyse the identifiable error patterns in the fine-tuned DistilBERT model, offering insights about possible further improvements.



(a) Token-level (b) Span-level

Figure 3: Confusion Matrices using CRF Model 3 on Test Set.



(a) Token-level (b) Span-level

Figure 4: Confusion Matrices using the Fine-tuned DistilBERT Model on Test Set.

7 Error Analysis

This section analyses error patterns identified in the negation scope detection task using the DistilBERT-base-uncased transformer on the test set based on the span-level evaluation. We chose this model as it has yielded the best results, and we were interested in finding out if the errors in the predictions were systematic or random. In total, 74 errors were made: 37 of type both FN-FP, 36 partial match FNs, and one FP. The analysis reveals notable patterns as shown in Table 10. Most commonly the machine includes phrases that are out of the scope according to the gold labels. The sentences often include more than one error category, therefore, the total amount of errors might not match the sums in the table. This section will describe common patterns, provide examples, offer linguistic explanations, and suggest possible improvements to avoid them in future work. In the examples, negation cues are bold, gold annotations are underlined, and model predictions are in curly brackets. Sentence IDs are given in parentheses.

a. Inclusion of Out of Scope Phrase

One of the most frequently recurring error pattern in our analysis involves the inappropriate inclusion of phrases that belong outside the negation scope. In most cases, the gold annotations do not include the verb in the negation scope. The model however, does include the entire verb phrase.

1) I {knew <u>you could</u>} **not** {<u>think of Beecher without thinking of that also</u>}.(cardboard_50_2)

b. Exclusion of In Scope Phrase

In contrast to the previous pattern, the model often fails to include phrases within the negation scope. This issue is particularly crucial as it can change the intended meaning of the sentence. This mostly

occurs in long, complex sentences with multiple clauses. When the model detects subordinate clauses, often separated with commas, it does not interpret the clause as an important aspect to the negation scope.

2) This morning {he had} **not** {gone ten paces down the road when two men came up behind him}, threw a coat over his head, and bundled him into a cab that was beside the curb.(circle01_207_1)

c. Adjectives/Adverbs

Another notable error pattern is the incorrect prediction of entire phrases as being within the scope when they should not be. This pattern is particularly evident when the negation cue involves an adjective or adverb with an affix, leading to incorrect annotations. The model seems to struggle with understanding the different roles that adjectives and adverbs play in the context of negation. The model fails to learn that adjectives and adverbs should be labeled differently based on their syntactic functions.

Morante et al. (2011) mention in the annotation guidelines that when an affix attached to an adjective modifies a noun, only the noun phrase should be included in the scope. In the example below, the system fails to learn this pattern, and includes the clause to be in scope instead.

3) [...] and {we had ceaseless rows about nothing}. (cardboard_413_1)

Furthermore, the guidelines mention that when an affix attached to an adverb modifies a verb, the entire sentence should be included in the scope. In the example sentence below, the adverb *undoubtedly* does not affect any verb, so the annotators only included the cue in the scope. Again, the system takes the sentence inside the scope.

4) {He would undoubtedly have done so to her old address}. (cardboard_322_1)

d. Exclusion of Random Tokens

Another recurring pattern this report identified is the inconsistent exclusion of random tokens that are within the scope of negation. This phenomenon is illustrated with the following example, where the system correctly predicts the scope except for the word *Civil*. There appears to be no clear reason as to why this pattern may happen.

5) I was well aware that {you could} **not** {do this without thinking of the mission which he undertook on behalf of the North at the time of the} Civil {war} [...] (cardboard_49_2)

e. Inclusion of Random Tokens

Similar to the previous error, this error pattern includes random tokens often at distance from the negation cue, with no apparent relation to the scope or cue, as exemplified below. The system predicts *Browner* to be inside the scope, while it has no relation to the scope. It may have misinterpreted the token *Browner* to be a comparative adjective that modifies *an unsuccessful lover*.

6) {An **un**successful lover} might have killed Mr. and Mrs. {Browner}.(cardboard_331_1)

f. Conjunctions

Furthermore, the model consistently fails to include conjunctions within the negation scope when it is contextually appropriate to do so. This oversight can significantly alter the intended meaning of sentences, as conjunctions often play a crucial role in connecting ideas or actions, see illustrated below.

7) Perhaps {she could} **not** or would not {have told you much}. (cardboard_256_1)

g. Punctuation

The following sentence presents a complex structure where the negation is part of a subordinate clause embedded within a larger sentence. This complexity is heightened by the use of commas to demarcate a relative clause. The model's failure to handle such complexity suggests a limitation in understanding the syntactic functions of punctuation in nested or compound structures. However, the issue remains in the fact that the subordinate clause is correctly identified as in scope, only the punctuation mark falls outside the scope.

8) The attack upon Mr. Warren further shows {that the enemy} , {whoever they are}, {are themselves} **not** {aware of the substitution of the female lodger for the male}. (circle01_288_1)

h. Conditional Clauses

The final pattern is the omission of dependent conditional clauses starting with *if*, from the negation scope. This exclusion is problematic as conditional clauses often play a critical role in the context and meaning of negation within a sentence. In the following sentence, the conditional clause is fully removed from the scope by the system prediction.

9) If my friends are not good enough for this house, then {I am} **not** {good enough for it either}. (cardboard_438_2)

Error Type	**#**	**%**	**Description**
Inclusion of Out of Scope Phrase	12	14.6	Machine predicts phrases to be inside the scope, while they should stay out.
Adjectives/Adverbs	10	12.2	Machine incorrectly predicts negation scopes when the negation cues are adjectives or adverbs.
Exclusion of In Scope Phrase	9	11.0	Machine predicts phrases to be out of the scope while they should be included.
Inclusion of Random Tokens	9	11.0	Machine includes random tokens at relative distance from the scope to be included in the scope.
Exclusion of Random tokens	9	11.0	Machine excludes random tokens that should remain inside the scope.
Conjunctions	6	7.3	Machine predicts conjunctions to be outside the scope.
Punctuation	5	6.1	Machine predicts punctuation mark to be outside the scope, even if it is in the middle of the negation scope.
Conditional	4	4.9	Machine predicts conditional clauses to be outside the scope.
Cue confusion	4	4.9	Machine excludes negation cues from scope when there are multiple cues in a sentence and some should be included inside the scope. Or machine includes negation cue in the scope, when it should not be included.
Relative clause	4	4.9	Machine excludes relative pronouns or relative clauses from the scope.
Including subject	3	3.7	Machine predicts the subject of a phrase related to the negation scope to be inside the scope.
Including preposition	2	2.4	Machine incorrectly includes preposition or prepositional phrase inside the scope.
Scope confusion	2	2.4	Machine incorrectly includes or excludes the scope of another negation cue within the same sentence.
Ellipsis	2	2.4	Machine fails to include parts of elliptical structure inside the scope.
Co-reference	1	1.2	Machine includes co-references in the scope, but not the first mentions, which should be included in the scope.

Table 10: Errors of the Fine-tuned "distilbert-base-uncased" Model on the Test Set.

The error analysis revealed several areas where the model needs to improve to enhance the model's capability of negation scope detection. One key strategy for this improvement involves broadening the training dataset with examples similar to the complex sentence structures identified in our error patterns. With this expansion, we expect the system to learn these patterns more accurately and improve its predictions in negation scopes, particularly in sentences that involve the use of conjunctions, conditional clauses, and cues that serve as adjectives or adverbs.

However, rectifying errors that involve the random inclusion or exclusion of words is a complex challenge. These types of errors are often unpredictable and do not follow a clear pattern, which makes them difficult to address, even with a larger training set. To include these irregularities, more advanced strategies may be necessary, such as in-depth linguistic analysis or the implementation of different neural network architectures that can better understand this particular aspect of language.

8 Conclusions

We introduced a fine-tuned transformer-based system for identifying the scope of negation in English sentences and compared its performance with that of a feature-based CRF model, which was trained using a set of linguistic features extracted from the labeled *SEM CD-SCO dataset. Our BERT-based model processed the same data, marking negation cues with a special token before inputting them into the system. This approach resulted in a noticeably improved generalization performance for the BERT model, reaching the token-level F1-score of 0.925 and span-level F1-score of 0.76 on the test set. Based on our experiments, the optimal set of features for the CRF model involve token-specific, constituency-based and dependency-based features. The hyper-parameter tuned CRF model reached a token-level F1-score of 0.743 and a span-level F1-score of 0.402 on the test set, staying behind the transformer-based model. Future research could involve additional experimentation with various feature combinations and the application of other neural models for the task of negation scope resolution.

Appendix 1: distribution of work for assignment 1

Selin Acikel Annotating file A. Annotator Agreement report sections: Error analysis, Conclusion. Small input in code writing for confusion matrix.

Murat Ertas Writing code for reformatting data files for negation scopes, evaluation metrics, confusion matrices. Annotating file A. Latex formatting for error analysis, confusion matrix.

Guo Ningxuan Writing introduction with minor editions, writing the report negation scope detection part, latex formatting, annotating file A.

Csenge Szabo Writing code for extracting annotated negation scopes and gold negation scopes; code for error analysis. Writing README and requirements.txt. Annotating file A. Writing the report sections: introduction, negation annotation task, dataset description, quantitative analysis.

Appendix 2: distribution of work for assignment 2

Selin Acikel Reading the paper FBK: Exploiting Phrasal and Contextual Clues for Negation Scope Detection. Writing feature description for negation scope detection.

Murat Ertas Reading the papers: UiO2: Sequence-Labeling Negation Using Dependency Features, BK: Exploiting phrasal and contextual clues for negation scope detection, A conditional random field model for resolving the scope of negation. Writing feature description for negation scope detection.

Guo Ningxuan Reading the paper Umich, A Conditional Random Field Model for Resolving the Scope of Negation, writing feature description for negation scope detection.

Csenge Szabo Reading the paper *SEM 2012 Shared Task: Resolving the Scope and Focus of Negation. Writing task description, and contributing to features for the negation scope detection section.

Appendix 3: distribution of work for assignment 3

Guo Ningxuan Writing code for feature-extractions: isNegationcue, LexicalisedPOS, Dependency path, Dependency distance, Dependency relationship. Writing report: selection of features part, minor editing of other parts

Murat Ertas Writing code for CRF script, working on SpaCy dependency parsing, sentence position features. Writing a small addition to problems and conclusions.

Selin Acikel Writing code for the Spacy dependency parsing, token distance to the cue, and writing code for hyper-parameter tuning. Writing 'selected features', 'fine-tuning', 'vectorization', and 'problems and conclusions'.

Csenge Szabo Writing code for pre-processing the dataset, extracting distributional statistics, extracting constituency and dependency features, feature ablation, updating code for evaluation metrics. Small addition to code for running CRF model. Writing section 4, 4.1, choice of model, feature ablation and evaluation in 5.1. Updating README and requirements.txt.

Appendix 4: distribution of work for assignment 4

Selin Acikel Incorporating feedback from A2. Writing section on 'BERT model description', 'conclusion', and minor addition on 'task description'. Running experiments on batches for fine-tuning.

Murat Ertas Writing code for BERT template implementation, prediction output for evaluation. Running experiments for parameter tuning. Writing fine-tuning adaptations section.

Guo Ningxuan Checking and running code for pre-processing and post-processing, modifying the report according to feedback, writing report part for related works, formatting citation works, language refining

Csenge Szabo Adjustments on code an report sections for Assignment 3. Writing code for pre-processing and post-processing the dataset for BERT input and output, adjusting evaluation script. Report sections: pipeline description, post-processing, quantitative analysis, problems and conclusions. Updating requirements.txt and README.

Appendix 5: distribution of work for assignment 5

Selin Acikel analyzing the errors, and writing the error analysis section.

Murat Ertas Writing/adapting an error filtering/retrieval code. Updating utilized features descriptions. Adapting BERT code to get output probabilities to update BERT output section. Creating a spreadsheet with relevant statistics and classifications of error for manual error assessment. Writing a part of the conclusion.

Guo Ningxuan Write the abstract and introduction, latex formatting, proofreading.

Csenge Szabo Adjustments based on Assignment 3 and 4 feedback on code and report. Updating pre-processing, dataset description sections. Writing sections: results, data annotation, part of the conclusion, introduction. Updating README and requirements.

References

Amjad Abu-Jbara and Dragomir Radev. 2012. Umichigan: A conditional random field model for resolving the scope of negation. In ** SEM 2012: 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 2012)*, pages 328–334.

Benita Kathleen Britto and Aditya Khandelwal. 2020. Resolving the scope of speculation and negation using transformer-based architectures.

Katherine Cheng, Timothy Baldwin, and Karin Verspoor. 2017. Automatic negation and speculation detection in veterinary clinical text. In Jojo Sze-Meng Wong and Gholamreza Haffari, editors, *Proceedings of the Australasian Language Technology Association Workshop 2017*, pages 70–78, Brisbane, Australia, December.

Md. Faisal Mahbub Chowdhury. 2012. FBK: Exploiting phrasal and contextual clues for negation scope detection. In Eneko Agirre, Johan Bos, Mona Diab, Suresh Manandhar, Yuval Marton, and Deniz Yuret, editors, **SEM 2012: 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 2012)*, pages 340–346, Montréal, Canada, 7-8 June. Association for Computational Linguistics.

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

Federico Fancellu, Adam Lopez, and Bonnie Webber. 2018. Neural networks for cross-lingual negation scope detection.

Dan Jurafsky and James H. Martin, 2022. *5.6.2 The Stochastic Gradient Descent Algorithm*. Pearson.

Aditya Khandelwal and Suraj Sawant. 2020. NegBERT: A transfer learning approach for negation detection and scope resolution. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5739–5748, Marseille, France. European Language Resources Association.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Emanuele Lapponi, Erik Velldal, Lilja Øvrelid, and Jonathon Read. 2012. UiO 2: Sequence-labeling negation using dependency features. In Eneko Agirre, Johan Bos, Mona Diab, Suresh Manandhar, Yuval Marton, and Deniz Yuret, editors, **SEM 2012: 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 2012)*, pages 319–327, Montréal, Canada, 7-8 June. Association for Computational Linguistics.

Roser Morante and Eduardo Blanco. 2012. *SEM 2012 shared task: Resolving the scope and focus of negation. In Eneko Agirre, Johan Bos, Mona Diab, Suresh Manandhar, Yuval Marton, and Deniz Yuret, editors, **SEM 2012: 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 2012)*, pages 265–274, Montréal, Canada, 7-8 June. Association for Computational Linguistics.

Roser Morante and Walter Daelemans. 2012. Conandoyle-neg: Annotation of negation cues and their scope in conan doyle stories. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 1563–1568. European Language Resources Association (ELRA).

Roser Morante, Sara Schrauwen, and Walter Daelemans. 2011. Annotation of negation cues and their scope guidelines v1.0. Technical report, CLIPS - University of Antwerp, May.

Geoffrey K. Pullum and Rodney Huddleston, 2002. *Negation*, page 785–850. Cambridge University Press, Cambridge.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Thinh Truong, Timothy Baldwin, Trevor Cohn, and Karin Verspoor. 2022. Improving negation detection with negation-focused pre-training. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4188–4193, Seattle, United States, July. Association for Computational Linguistics.