Applied Text Mining

Carol Rameder
Student number: 2700374

ABSTRACT

Negation cue detection is an essential component of natural language processing tasks, as it allows for an accurate understanding of the intended meaning of the text. This study presents the documentation of the negation cue detection task and evaluates the performance of SVM and CRF models on publicly available datasets from the *SEM 2012 Shared Task. The study analyzed 5520 phrases from stories of Conan Doyle, split into train, development, and test sets.

The results indicate that while the CRF model outperformed the baseline model, it still encountered difficulties in capturing compound negations and question phrases, leading to a high number of false negatives and false positives. Incorporating n-grams and a base word counter could improve the model's ability to capture these nuances. However, the study acknowledges that this method has limited performance potential compared to transformer-based models.

Overall, this study provides valuable insights into the challenges of negation cue detection and the potential for improving the accuracy of natural language processing tasks.

1 INTRODUCTION

The phenomenon of negation is a crucial element of language that pervades all known languages. Its semantic function is to invert the meaning of an affirmative statement, thereby serving as a fundamental mechanism of communication [27] [12]. The automatic identification of negation and its scope is a pervasive challenge in various natural language processing (NLP) applications, such as question-answering and sentiment analysis. Initially, negation detection emerged from the medical domain's need for reliable interpretation and indexing of clinical reports and discharge summaries [5]. To this end, the BioScope corpus represented a pioneering effort in the annotation of negation and hedge cues and their respective scopes [10]. Negation detection is crucial since disregarding it may yield incorrect results, such as providing incorrect answers in question-answering systems or predicting the opposite sentiment in sentiment analysis tasks [26]. Therefore, accurate identification and handling of negation are essential in developing robust and reliable natural language processing systems [2]. Furthermore, negation cue detection has practical implications in several areas such as medicine and law, where a precise understanding of negation is essential in making informed decisions. For instance, in the medical field, negation cue detection can assist in making accurate diagnoses, while in the legal domain, it can help in the accurate interpretation of contracts and laws [11].

In the First Joint Conference on Lexical and Computational Semantics - *SEM 2012 - a Shared Task on negation detection was released. It involves three distinct sub-tasks, namely cue identification, scope resolution, and event detection. Additionally, it is divided into two tracks: a closed track, where only the data provided by the organizers (such as word form, lemma, PoS-tag, and syntactic constituent for each token) can be utilized, and an open track, where participants are allowed to utilize any supplementary tools or resources.

The first task is identifying negation cues. These are words that directly express negation and can belong to multiple categories, which will be presented in Section 3.1. The second is determining the scope of the negation. The objective of this subtask is to identify the tokens in a sentence that are impacted by the negation cue. Scope refers to a sequence of tokens that may not be continuous. The third one is identifying the negated event or property, if it is explicitly mentioned. The scope is part of the meaning that is negated and the focus is part of the scope that is most prominently or explicitly negated. It is important to mention that the negated event or property is always within the scope of a cue but the negation cues are never part of the scope [16]. To make it clearer, the example below can be observed. There, the bold word represents the negation cue, the square parenthesis mark the scope of the negation and the underlined word is the negated event.

Example: [Sam had] **never** [<u>run</u> a full marathon].

In this paper, we document our process for performing the first task - negation cue detection. Our system would have aligned to the open track since we extracted the features with tools of our own choice. To achieve this we first selected two algorithms that assign to each token one of the three possible labels: I (Inside the negation cue), O (Outside the negation cue), and B (Beginning of the negation cue).

In Section 2 we provide an overview of the relevant literature pertaining to our task. Next, we explain the steps of performing an annotation task ourselves in Section 3. Subsequently, we will delve into the details of our data in Section 4, including the characteristics of our dataset, preprocessing techniques, and an analysis of the data. In the methodology Section 5, we will provide a comprehensive discussion of the model employed in our study. The results of our analysis will be presented and critically evaluated in Section 6. The error Analysis of both algorithms is presented in Section 7. Finally, we will conclude and present our overall findings in the conclusion Section 8.

2 RELATED WORK

In order to have a complete opinion and knowledge about the model we are going to build, we have to read previous papers. Furthermore, we need to come up with a strategy that fits our main purpose and idea. Reading through previous papers we understand that they are based mainly in the biomedical domain.

[2] proposed an algorithm called NegEx. The NegEx algorithm relies on four types of lexical cues: negation triggers that indicate a negation, pseudo-negation triggers that contain negation triggers but do not negate the clinical condition, and termination terms that stop the scope of the negation trigger.

[21] developed a based system called Negfinder that could recognize negation patterns in biomedical text. This system has 2 parts,

a lexicon scanner, and a parser. [24] used a machine learning approach in which negation is encoded using several features. By beginning, analyzing the features, one feature checks whether a negation expression occurs in a fixed window of four words preceding the polar expression. Another feature accounts for a polar predicate having a negated subject. Furthermore, they have disambiguation features to handle negation words that do not function as negation cues in certain contexts.

Morante proposed a supervised approach for detecting negation cues and their scopes in biomedical text [22]. The system once again has 2 parts. Both parts are memory-based engines. The first part recognizes a token in a sentence and decides if that token is a negation signal. The other part finds the full scope of these negation signals. [14] proposed a rule-based method to determine the oppositeness of sentiments when there are circumstances that one or more examples of a simple negation term appears in a sentence.

2.1 Previously used features

The set of extracted features considered for the learning process plays an important role in the performance of the NLP system. Thus, we analysed a set of papers to get a better overview of the possible features that we can use. We focus on the features used for the negation cue detection, as this is the task addressed in this paper.

The features used in this paper [7] show a clear targeting of the most common negation cues. To achieve this, the authors create binary features specific to the negation cues and thus facilitate to model to learn a clear correlation between a negative cue and a positive value for the targeting feature. For example, the authors implemented “hasNegPrefix”, “hasNegSuffix” and “matchesNegExp” features to indicate to the model that some tokens are highly likely to be negation cues. The “hasNegPrefix” takes the value “1” if a given word begins with one of the prefixes specific to a negation cue such as “un-”, “dis-” or “im-” and otherwise “0”. Authors of [2] selected these two features as well for the negation cue detection. In addition, the feature “matchesNegExp” indicates if a token belongs to a preset list of words that usually are classified as negation cues including, for example, “nor, “neither” and “without”. Therefore, we considered these features when proceeding with the ablation study.

To make the use of the first two features more efficient and accurate, authors of [9] make some additions to the previously described methods. The authors extract character n-grams from the start and end of the base word that an affix is attached to, such as possi, poss, pos,... and sible, ible, ble,... for a word like "impossible". This is done in conjunction with details about the affix (im) and the part of speech of the token ("JJ"). Another innovative addition is to count the number of occurrences of the base word the affix attaches to. This indicates to the model how likely it is that the base word has meaning itself. For example, "possible" has a high number of occurrences and thus the model learns that "im" from "impossible" is a negation cue. In contrast, "portant" will have 0 occurrences, thus "im" from "important" will be less likely to be labelled as a negation cue.

Finally, the common features of an NLP system like lemma, part of speech (POS), and lemmatized POS were considered by the authors of [9] [2] [7].

3 ANNOTATION TASK

Before making any progress in the coding section and the model building, our task includes an annotation of 12 different texts by each member of the group in order to fully understand the concept of negation. For this task, we use the "e-Host" application through Java comparing the annotations we make with each other. These annotations include negation cues we searched manually through the texts. After hitting an acceptable matching score with each member of the group we can proceed to the model building with a full understanding of the main attribute of the task.

3.1 Types of negation

[22] offers a very simple categorization of negation cues. In this article, the authors mention four types of negation:

- single words (e.g., never)
- multiwords (e.g., no longer, by no means)
- affixes(e.g. im-, -less)
- discontinuous, e.g., neither [...] nor.

In addition to that, [9] adds a further categorization of affixes into:

- prefixes: "im" in "impatience"
- infixes: "less" in "carelessness"
- postfixes: "un" in "unstable"

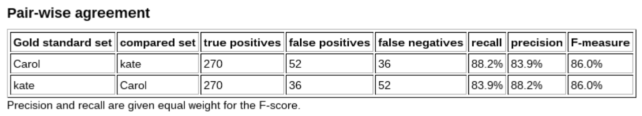
However, the annotation guidelines [18] offer a full categorization of the negation cues for each part of speech.

- adverb: "never"
- affixes: attached to an adjective (previously mentioned); attached to a verb: "n’t"
- determiners: "no" in "no idea"
- interjection: "No" in "No, what did you do?"
- pronouns: "nothing"
- preposition: "without"
- verbs: "fail"

3.2 Our previous work

In order to get a better understanding of Negation Cue Detection and Corpus Annotation, we performed a data annotation experiment involving all members of the group. After every member performed the annotation step, we conducted Inter-Annotator Agreement Analysis and Error Analysis. The latter brought the theoretical knowledge needed to evaluate Machine Learning models performing a similar task on a different dataset.

To annotate the data, we followed the guidelines presented by [18]. This article uses two stories by Conan Doyle as context and indicates the possible negation cues for each part of speech and offers examples at the sentence level on how to identify negations and their scope. There were examples offered regarding non-continuous scopes and special constructions like questions and imperatives. Finally, particular constructions with false negations and non-existing scope were provided to avoid expected mistakes.

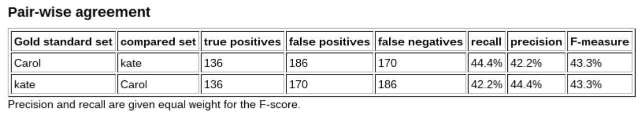


Pair-wise agreement

Gold standard set	compared set	true positives	false positives	false negatives	recall	precision	F-measure
Carol	kate	270	52	36	88.2%	83.9%	86.0%
kate	Carol	270	36	52	83.9%	88.2%	86.0%

Precision and recall are given equal weight for the F-score.

Figure 1: Results - Overlapping spans



Pair-wise agreement

Gold standard set	compared set	true positives	false positives	false negatives	recall	precision	F-measure
Carol	kate	136	186	170	44.4%	42.2%	43.3%
kate	Carol	136	170	186	42.2%	44.4%	43.3%

Precision and recall are given equal weight for the F-score.

Figure 2: Results - Exact matching spans

The corpus we worked with contains texts from official sources such as the National Health Service UK, but also independent publications like the National Vaccine Information Center or Natural News, thus showing both pro and anti-vaccination opinions. We annotated negation cues in texts related to the vaccination debate using the e-Host annotation tool [1]. All annotations were added to our GitHub Repository.

Inter-annotator agreement (IAA) study brings insights into the annotation process by comparing, analyzing, and quantifying the annotations of different people. In our case, each member independently got familiar with the guidelines and annotated the same 12 texts found in the fifth batch. In order to perform the IAA study, everyone shared their annotated files, which were compared between them pairwise. Due to the span mismatches caused by the different operating systems, the team members were working on, we could only select two annotators to perform our analysis. These will be referred to as A and B further on. To obtain the IAA scores we generated the Annotator Performance Reports with Ehost. We took into consideration overlapping spans and obtained the results from Figure 1 and Figure 2:

3.2.1 Error Analysis. An error analysis was conducted to identify and explain systematically sources of disagreement among annotators.

The number of true positives halves from 270, when selecting overlapping spans, to 136 when considering exact spans.

The first insight brought by the Error Analysis between A and B is that there was a high amount of common negations found, but the span selected was different. To assess that, the first three files were selected as a random sample. In this, we found 27 exact span matches and 28 overlapping span matches. All the exact span matches were single word cues like "not, no, nothing", whilst non-matching spans included suffixes like “n’t”, prefixes - in, un and constructions like “do not” or “no idea”. Annotator A always selected the smaller group of words such as “n’t” in “won’t”, “don’t” or “un” in “unvaccinated” and even “no” in “no idea” whilst B selected the whole construction.

This is caused by a different interpretation of the annotation task. The annotation guidelines indicated clearly that the negation cue is only the affix in the previous examples. Thus, the most probable explanation is that B misinterpreted the guidelines, whilst A followed them strictly.

[1]https://github.com/chrisleng/ehost

Another important aspect indicated by this analysis is the irregularity of the negations found in false positives and false negatives. All categories of negations including single-word, multi-word and prefixes were missed by both annotators. This indicates that human error is the most probable cause. However, B always identified as negation the constructions "anti-" and "unless", whilst A omitted them. This is caused by a different interpretation of the meaning of these words, which do not directly show a negation. Moreover, these were not suggested as possible cues in the guidelines. This is the only identified case of annotator bias.

More time allocated to this task would allow the annotators to better inspect the guidelines and do double verification of their work. In our case, this would have increased the inter-annotator agreement.

4 DATA

4.1 Dataset Description

The dataset used in this task is provided by the organizers of the Shared Task [16] and is focused on determining the scope and focus of negation within two stories by Conan Doyle: "The Hound of the Baskervilles" and "The Adventures of Wisteria Lodge." The datasets initially offered for this assignment were text files. The task was to preprocess this data and extract features for our model.

Initially, the files contained 5 columns. The first column is the chapter name, the second is the sentence id, the third is the token id based on each sentence the fourth is the word by itself which contains the tokens, as identified by the author of the file. The last column denotes the negation of the word and represents the label. It can have three values: ’O’ for tokens outside the negation cue, B-NEG for the first word of a negation cue and I-NEG for a word inside a negation cue (’than’ in the case of ’rather than’, which is a compound negation cue).

The goal of this assignment is to preprocess the data and extract the features in a suitable format for the machine-learning model. In our case, we analyzed the previous work and the available literature and decided to store the features in a suitable format for the Conditional Random Field (CRF) model as described in Section 5.

The training and development datasets were provided on the course page and had the names listed below. We train the model on the training set and evaluate it on the development set.

- SEM-2012-SharedTask-CD-SCO-dev-simple.v2.txt
- SEM-2012-SharedTask-CD-SCO-training-simple.v2.txt

The two following files were merged into a single table, to enlarge the number of testing instances, as there were the same columns in both. For experiments, they will be tested separately as well.

- SEM-2012-SharedTask-CD-SCO-test-cardboard.txt
- SEM-2012-SharedTask-CD-SCO-test-circle.txt

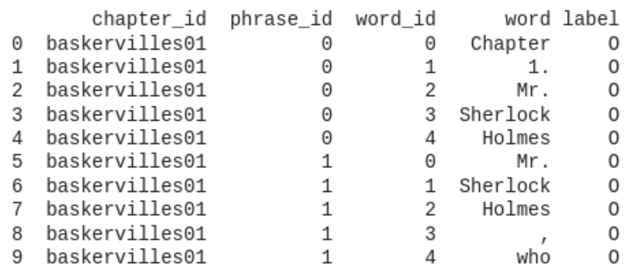
An overview of the dataset can be observed in Table 1. The number of instances of each type of token found in training, development and testing datasets accordingly can be seen in Table 2. Given that our task is to identify the negation cues, the unbalanced proportion of the training set indicates the need to undersample the non-negation cues when testing the model.

set	number of chapters	chapters name	phrases per chapter	tokens per phrase	number of tokens
train	14	"baskervilles01" (...) "baskervilles14"	from 134 to 399	min: 2 max: 83 avg: 17.96	65451
dev	2	"wisteria01" and "wisteria02"	347, 340	min: 2 max: 63 avg: 17.23	13567
test	3	"cardboard", "circle01", "circle02"	222, 371, 496	min: 2 max: 68 avg: 17.65	19216

Table 1: Dataset description

no. of instances	training	dev	test
B-NEG cues	987	176	269
I-NEG cues	16	3	5
non-negation cues	64448	13388	18942

Table 2: Instances count overview



```
      chapter_id  phrase_id  word_id      word label
0  baskervilles01          0        0   Chapter     O
1  baskervilles01          0        1        1.     O
2  baskervilles01          0        2       Mr.     O
3  baskervilles01          0        3  Sherlock     O
4  baskervilles01          0        4    Holmes     O
5  baskervilles01          1        0       Mr.     O
6  baskervilles01          1        1  Sherlock     O
7  baskervilles01          1        2    Holmes     O
8  baskervilles01          1        3         ,     O
9  baskervilles01          1        4       who     O
```

Figure 3: Input data

5 METHODOLOGY

5.1 Data Preprocessing

The first step of data preprocessing was merging the two datasets accordingly for training and testing using Pandas[2]. This did not require other additional processing steps, as the tables had the same columns already, which were further renamed as shown in Figure 1.

The next step was to merge from the data frame the words into sentences, as most of the features of a word depend on the context in which they appear. This was implemented, by filtering the table. We used 'chapter id' and 'phrase id' to uniquely identify a phrase and then merged the values of the 'word' attribute to get its string form. To preprocess the text data, we used the Spacy library[3], which is dedicated to Natural language Processing tasks. It was used to extract the tokens from the previously obtained phrase. Using these tokens we extracted the features from the text as described in the next section.

We removed the punctuation and stop words simultaneously with extracting tokens from the phrase. We used the 'is stop' and 'is punct' token properties available in Spacy. More precisely, we did not add the features of tokens with these properties to the training data, whilst their labels were removed accordingly.

However, when removing the stopwords with a naive initial approach, the number of negation cues from the training set decreased significantly from 987 to 166. We definitely did not want to remove them, as this would affect the performance. This is caused by the fact that some negations belong to the list of stop words [4] of Spacy. Therefore, a further check was done on the tokens and if they belonged to the preset list, they were kept in the final dataset. The list contained the words found in the intersection between negation cues and stop words that would have been eliminated otherwise. Some other words with the 'I-NEG' label were also kept to help the model also learn this under-represented label. This list was inspired by the work of [7] and was checked using the list of words in the annotation guidelines [17]. It included the following words (and their uppercase form): *nor, neither, without, nobody, none, nothing, never, not, no, nowhere, non, n't, rather, than, for, the.*

5.1.1 *Features format.*

The training set consists of a list of phrases. Each phrase consists of a list of dictionaries (as shown below), each element representing the features of one token. The key-value pairs consist of a feature and its value. Thus, the training and test data both are lists of dictionaries. The label data for both training and testing consists of a list of either 'B-NEG', 'I-NEG' or 'O'. The data was pre-processed to fit the CRF model, which can learn from attributes with categorical values, thus the data was not transformed into a numerical format. An example of one such data point can be seen in Figure 4.

5.1.2 *Issues encountered.*

The experiment presented certain challenges in the pre-processing step, specifically in regard to the decision-making process. One of the difficulties encountered was determining the optimal pre-processing approach that would work best for each one of the chosen models. Additionally, aligning the input to the requirements of both models proved to be problematic, as the input data was in the form of a list of dictionaries, which made it difficult to manipulate and process. Another issue encountered was keeping the labels and word list at the same length when removing stop words or punctuation. Moreover, matching the length of the labels list with the input turned out to be even more problematic. The number of tokens extracted using Spacy was usually not equal to the length of the list of labels accordingly. These challenges highlight the importance of carefully considering the input data and pre-processing steps when working with machine learning models, as these steps can greatly impact the performance of the model. Overcoming these challenges requires a thorough understanding of the input data and the requirements of the model being used, as well as a willingness to experiment with different pre-processing techniques until the optimal approach is found.

5.2 Features

In this section, we will discuss all the features which we are using in our project and will shed light on the use case of those features.

[2]https://pandas.pydata.org/
[3]https://spacy.io/
[4]https://machinelearningknowledge.ai/tutorial-for-stopwords-in-spacy/

5.2.1 Lemma. In Natural Language Processing, lemma refers to the uninflected form of a word, which is often the word's dictionary form [15]. Lemmatization is the process of identifying the lemma of a word, and it is commonly used in various NLP applications including text classification, information retrieval, and text generation. An example of a lemma is "run" which is the base form for "running", "ran" and "runs". In our case, the lemma was extracted from a token by using the NLTK library [5], which allows the POS tag as an argument as well. This increases the accuracy, as the lemma of a token can be different depending on the meaning of the word in the current phrase. For example, the lemma of "books" is "book" if the word is a verb and "books" if it is substantive. To implement this, we used the ".lemma" attribute from NLTK.

5.2.2 Part of speech tagging. In NLP, POS tags are labels that are assigned to words in a text to specify their grammatical role, including nouns, verbs, adjectives, adverbs, and so on. The process of POS tagging involves automatically assigning these labels to words in a text [13].

Fine-grained POS tagging utilizes a comprehensive set of POS tags that categorize words into distinct grammatical forms and subtypes [25]. For example, a fine-grained POS tagger may differentiate between different subcategories of nouns like common, proper, singular, and plural, and different forms of verbs such as transitive, intransitive, regular, and irregular.

Coarse-grained POS tagging, in contrast, employs a limited set of POS tags that classify words into broader groups [25]. For example, a coarse-grained POS tagger may only distinguish between major categories such as nouns, verbs, adjectives, and adverbs.

In our case, the ".pos" and ".tag" attributes of tokens from the Spacy library were used.

5.2.3 Dependency. (Reformulate and add/remove reference dep)

Dependency refers to the grammatical connections between words in a sentence. The approach of dependency grammar describes these connections by analyzing the dependencies between words in a sentence. Dependency parsing is the process of identifying the dependencies between words in a sentence. This can be used in NLP tasks such as information extraction or when you are understanding the language.

In this step, the phrase is organized in a tree structure using a directed acyclic graph as shown in Figure 2 and Figure 3. The phrase used for the example is *Holmes was sitting with his back to me, and I had given him no sign of my occupation.* Usually, the predicate is the root and each token is connected to a parent, on which it depends. The text of the parent token represents the 'head' attribute. Each branch in the tree represents the type of dependency that each node has on its parent. This represents the 'dependency' attribute in our model. Another dependency feature considered is the length of the path to the root attribute, which was implemented using the breadth-first search on the phrase tree and incrementing the distance by 1 after each step.

5.2.4 Other features. There are specific suffixes and prefixes found mainly but not exclusively in negation cues. Thus, indicating directly to the model that a token contains one of these letters group

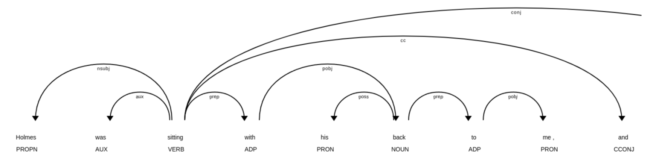


Figure 4: Left side of tree

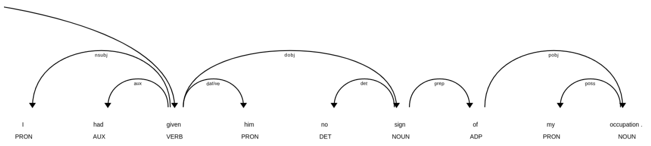


Figure 5: Right side of tree

```
text  :  back
lemma  :  back
fine_pos  :  NOUN
coarse_pos  :  NN
dependency  :  pobj
head  :  with
suffix  :  0
prefix  :  0
len_path_root  :  2
```

Figure 6: Features example

might facilitate the learning process and help it better identify our targets. The list using the model from [7] contains the following prefixes: *un, dis, im, in, non, ir, no* and the suffix *less.* In our case, each token is checked, and if one of these prefixes or the suffix is found, the according attribute is set to 1, otherwise, it is 0.

Figure 4 shows an example of the extracted features from the word 'back' belonging to the same phrase described in Section 5.2.3.

5.2.5 Feature Selection CRF. Feature selection provides an effective way to solve this problem by removing irrelevant and redundant data, which can reduce computation time, improve learning accuracy, and facilitate a better understanding of the learning model when working with high-dimensional data. The goal of this procedure is to obtain a subset from an original feature set according to certain feature selection criteria, which selects the relevant features of the dataset. [4]. By doing this, we reduce the variance of the model and thus diminish the chances of overfitting development data. Backward deletion - which we used during experiments - is a supervised feature selection method in which irrelevant attributes are removed consequently, starting from the initial set of all possible features.

In our case, we prioritized finding as many negative cues as possible. Thus the best indicator of the performance of our model was Recall, which we chose as the feature selection criterion. The development data was used throughout this procedure. The initial set of features can be seen in Figure 6, for which we got a 0.858 Recall. In the first step, we eliminated the features 'lemma' and 'text', as they directly indicate the form of the word. Without these

[5]https://www.nltk.org/

two, the Recall drops significantly to 0.435 so we added them back. Then, the figure slightly dropped to 0.835 when removing the 'fine POS' and 'coarse POS' features. In the next step, we removed the features describing the affixes - 'suffix' and 'prefix'. This lowered the Recall from 0.835 to 0.812. Further, we eliminated the dependency features - 'length of the path to root', 'dependency' and 'head'. This increased the Recall to 0.818. Finally, the 'lemma' attribute was removed as well and the performance of the model using only the 'text' attribute was 0.699. However, when we evaluated the model without the dependency features the performance dropped below the initial threshold. Through trial and error, we obtained the best setup when excluding the "len path root" and "head" attributes with 0.869 Recall.

When looking at the performance figures' evolution, we notice the high importance of 'text' and 'lemma' attributes. As expected, the model uses mostly the text form of a token to decide if it is a negative cue or not. The information brought by the dependency features removed does not help the model to better predict the labels, thus they can be omitted for further experiments. The most likely explanation for this is that the information is too abstract or high-level for the model to be used in this context.

5.2.6 Feature ablation of SVM. Feature ablation is a systematic method to evaluate the contribution of individual features to the overall performance of a machine learning model. The process involves either removing or reducing the impact of each feature and observing the change in performance. This information can be utilized to optimize the feature set, enhance the performance of the model, and make informed decisions regarding feature engineering and selection.

In our study, we conducted a feature ablation experiment in which we removed each feature, including tokens, POS tags, and lemmas, and evaluated the accuracy of our model. The model achieved an accuracy of 0.997% when all features were included. Our results indicated that both the 'Tokens' and 'Lemma' features had an impact on the overall accuracy, with observed changes in accuracy upon their removal. However, the 'Previous token' and 'next Token' features had a negative impact on our task, with a modest increase of 0.0009% in accuracy noted after their removal. Although this increase was not substantial, it was still noteworthy in our study.

5.3 Algorithms

5.3.1 Baseline Model. A baseline model is essentially a simple model that acts as a reference or a starting point in a machine-learning project. Its main function is to contextualize the results of trained models. Baseline models usually lack complexity and may have little predictive power. They serve as benchmarks for trained models and guide the author in evaluating the improvements brought by the systems [19].

In our case, we benchmark the CRF and SVM models using the baseline in order to assess if the learning process worked properly and if the systems gained predictive power. The baseline chosen for our experiment is a simple rule-based model. It classifies a token as "B-NEG" if it belongs to a preset list of words – presented at the end of Section 5.1 – or if it contains negation cue specific affixes. The second rule was implemented by checking if the current token has a non-null value for the "prefix" and "suffix" features described in Section 5.2.4.

As the results from Figure ?! show, the model achieves a high Recall but low Precision score. This outcome was expected as the system allows a large number of tokens to be classified as negation cues. For example, all tokens starting with "im" will have a non-null value for the "prefix" feature and therefore be classified as negation cues, without necessarily being the case. Our aim for the chosen models is to increase the precision score while maintaining the Recall high as well.

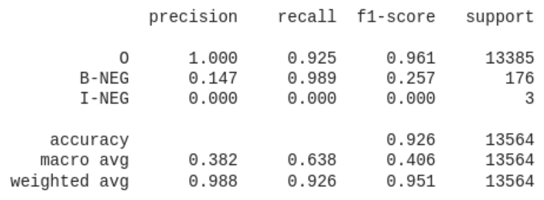

```
              precision    recall  f1-score   support

           O      1.000     0.925     0.961     13385
       B-NEG      0.147     0.989     0.257       176
       I-NEG      0.000     0.000     0.000         3

    accuracy                          0.926     13564
   macro avg      0.382     0.638     0.406     13564
weighted avg      0.988     0.926     0.951     13564
```

Figure 7: Results of the Baseline Model

5.3.2 CRF model. Conditional Random Fields (CRF) is a type of Machine Learning model optimized for making predictions where the surrounding context or neighbouring states impact the current prediction outcome. CRF belong to the Discriminative Classifiers class, and as such, they model the decision boundary between the different classes by applying Logistic Regression on sequential data. The model takes into account the context or dependencies between the observations, making it well-suited for tasks where neighbouring observations have an impact on the prediction outcome. CRFs are often trained using the maximum likelihood estimation principle and the learned parameters define the conditional probabilities of the labels given the input features. The prediction process involves finding the most probable sequence of labels given the input features. Due to their capacity to model sequential data, CRFs are commonly utilized in Natural Language Processing and possess a variety of applications within that field [6]. Therefore, this algorithm was considered a suitable method for our task.

$$\hat{y} = \text{argmax}_y P(y \mid x)$$

In our case, the sequence is a sentence from one of the datasets. The dependencies are represented through the features of each separate token. We aim to predict if a token is a Negation Cue or is located inside a Negation Cue. This is done by assigning one of the two labels ($\hat{y}$ is either B-NEG or I-NEG) accordingly based on the input features described previously in Section 5.2. Thus, a procedure similar to Logistic Regression is used to determine a categorical feature.

Each token considered an input for the CRF models needs to be described by a feature function determined by its features, its label, its position in the sentence and the features and labels of the previous token. In our case, we did not specifically extract this large set of features for each token. Instead, we leveraged the dependency features, described in Section 5.2.3, to let the model learn the sequences. By doing this, an input token is determined using fewer features and without decreasing the performance of the model.

The CRF model assigns a set of weights for each feature function and learns them by minimizing the Negative Log cost function using a method specified as a parameter of the algorithm. In our case, we tested Gradient descent using the L-BFGS method (LBFGS) and Stochastic Gradient Descent with the L2 regularization term (L2SGD).

The implementation of the CRF model was done using the Scikit-learn library [6] and its documentation and tutorial [7] for this algorithm.

5.3.3 SVM model. The Support Vector Machines (SVM) algorithm is a widely recognized machine learning approach for classification problems [23]. The SVM algorithm aims to find the optimal hyperplane that separates the data into two distinct classes. The hyperplane is determined by maximizing the margin between the closest data points from each class, referred to as support vectors [23]. Predictive outcomes are produced through the calculation of the dot product between the feature representation of each data point and a set of weights. Additionally, the SVM algorithm is equipped with the ability to handle non-linearly separable data through the implementation of kernel functions. The versatility of SVM, demonstrated through its application in various domains including text classification, image recognition, and bioinformatics, has solidified its position as a highly-regarded machine learning algorithm [20].

The reason behind utilizing Support Vector Machines (SVM) in negation cue detection lies in its effectiveness and efficiency for binary classification problems [20]. Negation cue detection, being a binary classification task, can benefit from the capabilities of SVM. The algorithm can easily learn to identify patterns in text that suggest negation cues. Furthermore, SVM has demonstrated exceptional performance in NLP tasks, making it a suitable choice for negation cue detection[20].

The objective of this study is to evaluate the effectiveness of Support Vector Machines (SVM) in detecting negation cues within a corpus of text. The labelling scheme employed in the training data is BIO-tagging, assigning each token a label of either "B-NEG," "I-NEG," or "O" depending on whether it is a negation cue or not. The training dataset was comprised of tokens, chapter identification, token position, and the corresponding gold label.

To train the SVM model, we utilized tokens as well as supplementary dependency features, such as Part-of-Speech (POS) tags, the lemma of each word, and the preceding and following words. The inclusion of these features aimed to provide the SVM model with additional context and pattern recognition capabilities in detecting negation cues. This, in turn, aimed to improve the accuracy of the SVM model in making classifications.

5.4 Experimental setup

5.4.1 CRF. The three experimental setups described below (A, B, C) were tested using the development set. The goal of this step is to get the best configuration of the model that would be tested once with the test set. The final run was done using the hyper-tuned model, and thus the best version of the model we could build with the training data available. In the last run, we used the test data for the first time, in order to not overfit the model. The features resulting from the ablation study (Section 5.2.5) were considered for these experiments.

First experimental setup (A):

- Optimization method: L2SGD
- L2 regularization coefficient: 0.1
- Maximum number of iterations: 1000

Second experimental setup (B):

- Optimization method: LBFGS
- L2 regularization coefficient (c1): 0.1
- L1 regularization coefficient (c2): 0.1
- Maximum number of iterations: 100

For the third experimental setup (C), we used hyperparameter tuning for the LBFGS algorithm, as it yielded a higher F1 score on the first run, even though fewer iterations were allowed. The dataset used is the development set. The method used was RandomizedSearchCV [8] and its parameters were:

- The search space for c1 and c2: Exponential continuous random variables with the scales 0.5 and 0.005, from which the coefficients were sampled at each iteration
- The number of Cross-Validation folds tested at each iteration (CV): 5
- Number of iterations: 100
- Comparison measure: F1 Score

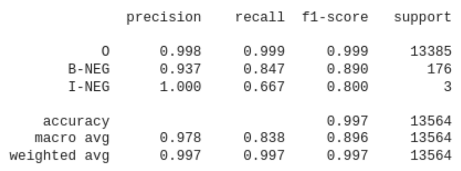
The Hyperparameter tuning phase yielded a c1 score of 0.084 and a c2 score of 0.013, which were further used to evaluate the model on test data.

5.4.2 Experimental setup SVM.

- Data Collection: a dataset of tokens, chapters, token positions and gold labels is collected for training, development and testing the SVM model.
- Feature Extraction: Dependency features such as Part-of-Speech (POS) tags, lemma and the previous and next words are extracted from the tokens.
- Model Training: The SVM model is trained on the dataset, using the extracted features as input.
- Evaluation: The performance of the trained SVM model is evaluated on development and tested on the test dataset.

6 RESULTS

6.1 Results CRF

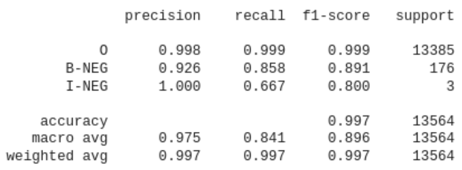


```
              precision    recall  f1-score   support

           O      0.998     0.999     0.999     13385
       B-NEG      0.937     0.847     0.890       176
       I-NEG      1.000     0.667     0.800         3

    accuracy                          0.997     13564
   macro avg      0.978     0.838     0.896     13564
weighted avg      0.997     0.997     0.997     13564
```

Figure 8: Results of the experimental setup - A

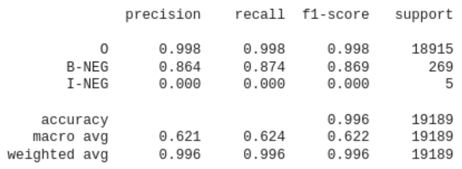
As seen in the figures, the CRF model worked well on development data but underperformed on testing data. It yielded a weighted

[6]https://scikit-learn.org/stable/
[7]https://sklearn-crfsuite.readthedocs.io/en/latest/tutorial.html
[8]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html



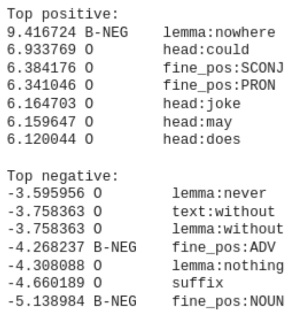
```
              precision    recall  f1-score   support

           O      0.998     0.999     0.999     13385
       B-NEG      0.926     0.858     0.891       176
       I-NEG      1.000     0.667     0.800         3

    accuracy                          0.997     13564
   macro avg      0.975     0.841     0.896     13564
weighted avg      0.997     0.997     0.997     13564
```

Figure 9: Results of the experimental setup - B



```
              precision    recall  f1-score   support

           O      0.998     0.998     0.998     18915
       B-NEG      0.864     0.874     0.869       269
       I-NEG      0.000     0.000     0.000         5

    accuracy                          0.996     19189
   macro avg      0.621     0.624     0.622     19189
weighted avg      0.996     0.996     0.996     19189
```

Figure 10: Results on the test set



```
Top positive:
9.416724 B-NEG    lemma:nowhere
6.933769 O        head:could
6.384176 O        fine_pos:SCONJ
6.341046 O        fine_pos:PRON
6.164703 O        head:joke
6.159647 O        head:may
6.120044 O        head:does

Top negative:
-3.595956 O        lemma:never
-3.758363 O        text:without
-3.758363 O        lemma:without
-4.268237 B-NEG    fine_pos:ADV
-4.308088 O        lemma:nothing
-4.660189 O        suffix
-5.138984 B-NEG    fine_pos:NOUN
```

Figure 11: Feature Analysis

F1 score of 0.622 for the testing set dropping from 0.896 on the dev set. Replacing the standard feature functions of the CRF model with dependency features yielded satisfactory results for the task of negation cue detection as shown in Figure 10, as the model yields a high F1-score for the "B-NEG" category. There are two possible explanations for that. The first one could be that the model learned from the sequential data using only the dependency features or the negation cues can be easily detected without using too much sequential information. Figure 11 confirms the results and findings of Section 5.2.5 - the dependency features are not important for this classification task. These features did not have any significant impact and thus, they do not appear in the feature values for which an instance is very likely to be in one of the classes.

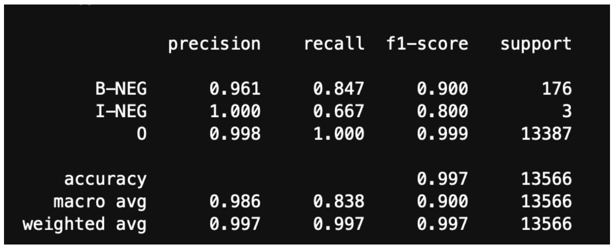
As seen in Figure 9, the word 'nowhere' is a clear indication that we encounter a negation cue, as expected. In addition, having a positive value for the suffix is strongly correlated with a token being a negation cue (not having the label 'O' and thus being outside of a negation cue).

A problem encountered is that, initially, we got only a 40% weighted F1 score when stop words were removed. Thus, for all the experiments mentioned above, the stop words were added back to the training data.

6.2 Results SVM

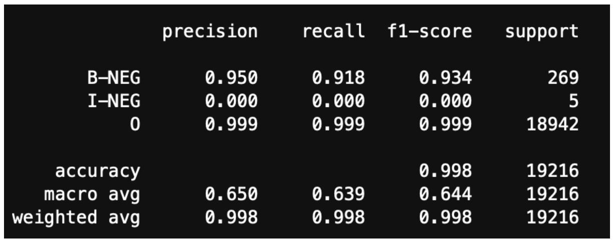
The SVM model performed exceptionally well on the development dataset and test set, achieving a weighted average f1-score of 0.997 and 0.998 respectively (Figure 12, Figure 13). The f1-score is a commonly used evaluation metric in machine learning that balances precision and recall and is a measure of a model's accuracy. An f1-score of 0.998 indicates that the SVM model was able to make highly accurate predictions in a binary classification task.

The exceptional performance of the Support Vector Machines (SVM) model on the development and test dataset can be primarily attributed to two key factors: its inherent capability as a classifier and the selection of dependency features.



```
              precision    recall  f1-score   support

       B-NEG      0.961     0.847     0.900       176
       I-NEG      1.000     0.667     0.800         3
           O      0.998     1.000     0.999     13387

    accuracy                          0.997     13566
   macro avg      0.986     0.838     0.900     13566
weighted avg      0.997     0.997     0.997     13566
```

Figure 12: Results of SVM model on development set



```
              precision    recall  f1-score   support

       B-NEG      0.950     0.918     0.934       269
       I-NEG      0.000     0.000     0.000         5
           O      0.999     0.999     0.999     18942

    accuracy                          0.998     19216
   macro avg      0.650     0.639     0.644     19216
weighted avg      0.998     0.998     0.998     19216
```

Figure 13: Results of SVM model on test set

6.3 Comparing results

The goal of generating Table 3 for comparison is to compare the performance of two models. The table includes Precision, Recall and F1 scores of the baseline model, SVM model and CRF model. These metrics are commonly used in machine learning to evaluate the performance of a model. By including these metrics in the comparison table, it is possible to see how the two models perform and make a fair comparison between them.

According to the results presented in the table, a few observations can be made. The baseline model was considerably improved, by making predictions with both chosen models. Thus, we can affirm that both of them learned from the extracted features and are able to make informed decisions. Secondly, the imbalance present in the dataset resulted in the 'I-NEG' class being inadequately predicted by any of the models. Nevertheless, the Support Vector Machine (SVM) model slightly outperforms the Conditional Random Field (CRF) model in terms of precision, recall, and F1 scores on both the dev set and the test set. The reason for this superior performance may be attributed to the inherent strengths of the SVM algorithm,

Table 3: Result comparison of the baseline model, CRF (including the 3 Experimental Setups) and SVM model with Precision, Recall, and F1-scores (macro)

	Precision	Recall	f1-score
Baseline			
Dev set	0.382	0.638	0.406
Test set	0.387	0.632	0.413
CRF			
Exp A	0.979	0.838	0.896
Exp B	0.975	0.841	0.89
Test set	0.621	0.624	0.622
SVM			
Dev set	0.986	0.838	0.9
Test set	0.650	0.639	0.644

which is capable of handling high-dimensional feature spaces and non-linear decision boundaries [23], in contrast to CRF, which is a probabilistic graphical model typically used for sequence labelling tasks.

7 ERROR ANALYSIS

Error analysis is a process in NLP where the performance of an algorithm is evaluated by analyzing its misclassified instances. In the case of a negation cue detection task, the error analysis process involves identifying the instances where the algorithm failed to identify the negation cue correctly. The misclassified samples are then manually examined to understand the reasons for the misclassification and identify patterns in the errors. This information can then be used to improve the performance of the algorithm by updating its training data, feature engineering, or model parameters. The goal of error analysis is to systematically analyze and understand the algorithm's weaknesses and make data-driven improvements to increase its accuracy.

Error Analysis of a Machine Learning model is important for an NLP task as it amplifies the contributions of the research article both in terms of short-term interest and long-term relevance. A good error analysis reasons why the chosen methods are effective or ineffective for the given task. This provides a much richer starting point for further research, allowing us to go beyond just showing the performance of some models while also discovering which are the harder parts of a problem [3]. An important insight we are looking for is recurring linguistic patterns that affect the performance of the model.

7.1 Error Analysis for SVM

The investigation of the errors produced by the SVM model revealed that approximately 25% of the errors were the result of inaccurate gold labels, where the model identified words like "nothing," "not," and "without" as negation cues but were not labelled as such by the gold standard. In this research, the imbalance in the dataset was identified as a challenge to the performance of the model. Specifically, the 'I-NEG' class had only 16 in training and 5 instances of the majority class in the test set. This resulted in the model primarily detecting the majority class and disregarding instances of the 'I-NEG' class. The remaining errors were a result of incorrect predictions by the model and could be attributed to sub-optimal feature selection or inadequate hyperparameter tuning. To address these errors, it may be beneficial to retrain the model using alternative features or to augment the current feature set to improve its performance.

7.2 Error analysis for CRF

To get the results of this section, the outputs of the model were added as a new column attribute in the test data frame with Pandas. This allowed us to easily filter the errors and compare between the prediction and gold labels.

The Error Analysis for the results of the CRF model focuses on the 'B-NEG' class. This is the most representative of our task of detecting negation cues. The 'I-NEG' class is never identified by the CRF model out of the 5 existing cases in the testing set. This is caused mainly caused by the lack of proper modelling and understanding of the language capabilities of our chosen model. Identifying the words inside of a multi-word negation cue also requires more training data - there were only 16 instances of this type out of a total of 65451 tokens. Two examples of this situation are "far from" and "absolutely nothing". In these cases, the model failed to recognize the negation cues, thus they belong to the class of **False Negatives**(FN). This class includes 24 instances out of 67 total errors made by the model. Out of these, only 7 were single-word negations like "not", "no", "nor" and "never" which are considered easy to predict. The last observed pattern of missed negation includes words with specific affixes. The majority of them (8) contain "un", for example, "unsuccessful", "uneasiness" and "unusual", whilst the rest contain "in" and less (4 in total).

The other 43 were **False Positives**(FP). Even though not all of them belong to a certain type of pattern, there were numerous words with negation-like affixes, which wrongly triggered to model to classify them as negation cues. Some examples from this category are "unfortunately", "uneasy", "listless", "ruthless" and "endless". These were initially labelled with the 'O' class, as the annotators considered they have meaning by themselves and do not negate other words or constructions. These could be arguably some annotations errors. The second identified pattern is negation cues from questions. There were 7 instances of "not", 2 of "without" and one "n't" which were misclassified as negation cues by the model, whilst their gold label is 'O'. A third pattern concerns fixed constructions such as "no doubt", "nothing but", "without a doubt", and "not only" in which "not", "nothing" and "no" are not negation cues. The model fails to learn these specific use cases due to the absence of context. Only the features of the current word were considered in our experiment.

A possible solution would be targeting these exact scenarios, for example adding a specific boolean feature to indicate if a word belongs to a question or not. Moreover, the n-gram feature could provide a larger context with previous and following words and this could facilitate CRF to learn the cases of fixed word constructions in both FP and FN cases. These additions come with the risk of overfitting, as the number of attributes will largely increase when trying to fix each error pattern manually. Moreover, a possible future work that will improve the performance of the CRF model is using the features described at the end of Section 2.1. These offer more knowledge to the model when it needs to label an affix attached to a base word such as "irritable" or "endless" from FP and the words starting with "un" from FN. For now, the model only knows if the word has a negation cue specific affix, the addition will also indicate the number of occurrences and the POS of the base word.

The main problem for the other occurring errors is that the CRF model lacks language understanding. A method to solve this problem would be using a transformer-based classifier such as BERT [8]. Unlike CRF, this goes beyond specific features like POS tags and "lemma" and understands the natural language. This type of classifier would probably increase the performance on the negation cue detection tasks, as it will better differentiate between true negation cues and words containing negation-like affixes.

Authors of [1] trained a model to identify whether a given sentence was negated or not. This is achieved by representing the negation identification task as a binary classification problem. This task required preparing the input data and formatting it to be suitable for the BERT fine-tuning procedure. They obtained 97.56% Precision and 98.89% Recall, considerably higher performance than both the SVM and CRF used here and on used larger dataset - 2670 phrases compared to our test set of 1089 phrases.

8 CONCLUSIONS

In this paper, we implemented the full concept of negation cues. Starting from researching, annotating and building a model that can recognize negation cues within a text. After processing the data we were given, we brought it into a format that we could use in our model. We then built a simple form of a baseline model to get the first results before building the two models we used.

The results of our experiments showed that both SVM and CRF are effective algorithms for negation cue detection, achieving high F1 scores on the dev set and bringing a significant improvement compared to the baseline. However, both of them drop in performance when tested on a newly seen dataset. Future possible work on this study should include experimenting with the token-level n-gram as a feature alongside a base word counter and a targeting feature for the specific cases of questions.

Appendices

Work splitting

- Carol: Corrected writing errors, and organized paragraphs and sections. Wrote and implemented all (CRF model notebook file from Github) the experiments for the CRF model (5.3.2) including data preprocessing (5.1), feature extraction and selection (5.2), experimental setup (5.4) and error analysis (7.2). Wrote the Error Analysis for the Annotation task (A2) (3, 3.2, 3.2.1) and Dataset description parts (4).
- Rishvik: Worked on implementation of SVM model file (code plus text), wrote results section of SVM in the result section. Wrote introduction segment, in data section wrote and described all the usefull features, worked on tables, indentation and grammatical errors. Worked and wrote on the error analysis part of SVM model as well, and compared the results.
- Apostolos: Worked on the implementation of the Baseline model (notebook file from Github) and the data pre-processing. Wrote and worked on the abstract , the conclusion, the previous work section, the annotation task and the types of negation, and the methodology section including the baseline model. Lastly compared the results .
- Github link: https://github.com/rishvik/textmining2023_aibi

REFERENCES

[1] Ghadeer Althari and Mohammad Alsulmi. "Exploring Transformer-Based Learning for Negation Detection in Biomedical Texts". In: *IEEE Access* 10 (2022), pp. 83813–83825. DOI: 10.1109/ACCESS.2022.3197772.

[2] Dragomir Radev. Amjad Abu-Jbara. "UMichigan: A Conditional Random Field Model for Resolving the Scope of Negation". In: (2012).

[3] Emily M. Bender and Emily M. Bender. *Error analysis in research and writing.* Jan. 2018. URL: http://coling2018.org/error-analysis-in-research-and-writing/.

[4] Jie Cai et al. "Feature selection in machine learning: A new perspective". In: *Neurocomputing* 300 (2018), pp. 70–79. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2017.11.077. URL: https://www.sciencedirect.com/science/article/pii/S0925231218302911.

[5] Wendy W Chapman et al. "A simple algorithm for identifying negated findings and diseases in discharge summaries". In: *Journal of biomedical informatics* 34.5 (2001), pp. 301–310.

[6] Ravish Chawla. *Overview of conditional random fields.* Apr. 2018. URL: https://medium.com/ml2vec/overview-of-conditional-random-fields-68a2a20fa541.

[7] Md. Faisal Mahbub Chowdhury. "FBK: Exploiting Phrasal and Contextual Clues for Negation Scope Detection". 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. Montréal, Canada: Association for Computational Linguistics, 2012, pp. 340–346.

[8] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[9] Lilja Øvrelid Emanuele Lapponi Erik Velldal and Jonathon Read. "iO2: Sequence-Labeling Negation Using Dependency Features." In: (2012).

[10] Richárd Farkas et al. "The CoNLL-2010 shared task: learning to detect hedges and their scope in natural language text". In: *Proceedings of the fourteenth conference on computational natural language learning–Shared task.* 2010, pp. 1–12.

[11] Bonnie Webber2 Hangfeng He Federico Fancellu. "Neural Networks for Negation Cue Detection in Chinese". In: URL: https://aclanthology.org/W17-1809.pdf.

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

[13] Deepika KumawatVinesh JainVinesh Jain. "POS Tagging Approaches: A Comparison". In: URL: https://www.researchgate.net/publication/277907730_POS_Tagging_Approaches_A_Comparison.

[14] Stuart RA Jia L Kaur J. "Mapping of the Saccharomyces cerevisiae Oxa1-mitochondrial ribosome interface and identification of MrpL40, a ribosomal protein in close proximity to Oxa1 and critical for oxidative phosphorylation complex assembly." In: *Research Support, U.S. Gov't, Non-P.H.S.* (2009).

[15] Divya Khyani. "An Interpretation of Lemmatization and Stemming in Natural Language Processing". In: URL: https://www.researchgate.net/publication/348306833_An_Interpretation_of_Lemmatization_and_Stemming_in_Natural_Language_Processing.

[16] Roser Morante and Eduardo Blanco. "*SEM 2012 Shared Task: Resolving the Scope and Focus 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).* Montréal, Canada: Association for Computational Linguistics, July 2012, pp. 265–274. URL: https://aclanthology.org/S12-1035.

[17] Roser Morante and Walter Daelemans. "ConanDoyle-neg: Annotation of negation cues and their scope in Conan Doyle stories". In: *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12).* Istanbul, Turkey: European Language Resources Association (ELRA), May 2012, pp. 1563–1568. URL: http://www.lrec-conf.org/proceedings/lrec2012/pdf/221_Paper.pdf.

[18] Roser Morante, Sarah Schrauwen, and Walter Daelemans. "Annotation of negation cues and their scope: Guidelines v1". In: *Computational linguistics and psycholinguistics technical report series, CTRS-003* (2011), pp. 1–42.

[19] Aashish Nair. *Baseline models: Your guide for model building.* Apr. 2022. URL: https://rb.gy/od4qk0.

[20] Petkov TsvetkovDimiter Petkov Tsvetkov Nikolay StanevskiDimiter. "Using Support Vector Machine as a Binary Classifier". In: URL: https://www.researchgate.net/publication/237653736_Using_Support_Vector_Machine_as_a_Binary_Classifier.

[21] A. Deshpande P. G. Mutalik and P. M. Nadkarni. "Use of general-purpose negation detection to augment concept indexing of medical documents: a quantitative study using the UMLS." In: *Journal of the American Medical Informatics Association : JAMIA, 8(6):598–609.* (2001).

[22] Anthony Liekens Roser Morante and Walter Daelemans. "Learning the scope of negation in biomedical texts." In: *In Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing,* (2008b).

[23] Massimiliano Pontil Theodoros Evgeniou. "Support Vector Machines: Theory and Applications". In: URL: https://www.researchgate.net/publication/221621494_Support_Vector_Machines_Theory_and_Applications.

[24] Janyce Wiebe Theresa Wilson and Paul Hoffmann. "Recognizing contextual polarity in phrase-level sentiment analysis". In: *In Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, HLT '05, pages 347–354, Stroudsburg, PA, USA. Association for Computational Linguistics.* (2005).

[25] Torsten Zesch Tobias Horsmann. "Assigning Fine-grained PoS Tags based on High-precision Coarse-grained Tagging". In: URL: https://www.researchgate.net/publication/313652213_Assigning_Fine-grained_PoS_Tags_based_on_High-precision_Coarse-grained_Tagging.

[26] Michael Wiegand et al. "A survey on the role of negation in sentiment analysis". In: *Proceedings of the workshop on negation and speculation in natural language processing.* 2010, pp. 60–68.

[27] Malcah Yaeger-Dror and Gunnel Tottie. "Negation in English Speech and Writing: A Study in Variation". In: *Language* 69 (Sept. 1993), p. 590. DOI: 10.2307/416702.