Research Report

Negation Scope Detection with LSTM Neural Networks

Daan de Jong

d.dejong1@uu.nl

5942713

Utrecht University

Faculty of Social and Behavioral Science
Department of Methodology and Statistics
Study programme: MSBBSS
Supervisor: dr. Ayoub Bagheri

1 Introduction

Negation is a complex grammatical phenomenon that has been widely studied by disciplines as philosophy, cognitive science and linguistics (Morante & Sporleder, 2012). Within the field of Natural Language Processing (NLP), it plays an important role in the semantic representation of text and has received considerable attention in recent years (Morante & Blanco, 2012).

A negation can be defined as “the implication of the non-existence of something” (Vincze et al., 2008) or “a grammatical category that comprises devices used to reverse the truth value of propositions” (Morante & Blanco, 2012). It is useful to make a distinction between the *cue* and the *scope* of a negative instance. A negation cue is a word, a morpheme or a group of words that inherently expresses a negation, for example, ‘never’, ‘impossible’ or ‘not at all’. Its negation scope comprises the elements of the text that are affected by it. Consider the following example, where the negation cue is boldfaced and the negation cue is enclosed by square brackets:

‘That’s [**not** a problem **at all**.]’

Note that the negation cue can be discontinuous, and that it is annotated as a part of its scope.

The current report will focus on negation scopes. When the goal is to derive factual knowledge from textual data, it is important that negation scopes are resolved correctly, given the fact that their meaning is heavily affected by their negation cue. Furthermore, negations are reasonably common, with a frequency of 27.6 per 1,000 words in spoken text and 12.8 per 1,000 words in written text (Tottie, 1991), so typical NLP tasks might benefit from adequate tools to handle negations.

1.1 Related work

Earlier efforts to detect the scope of negations in natural language can be categorized into rule-based approaches (e.g. Basile et al., 2012; Özgür & Radev, 2009; Øvrelid et al., 2010) and statistical learning methods (e.g. Fei et al., 2020; Zou et al., 2013; Tang et al., 2010). In both of these categories, models or systems for negation scope detection have performed considerably well. However, they need human-engineered features to reach this performance.

To overcome the need of manually-crafted features, researchers have proposed Neural Network-based approaches for the task of negation scope detection since these kind of models are able to learn unsupervised features from the data. For example, Qian et al. (2016) used a Convolutional Neural Network-based model to extract path features from syntactic trees and combined these with position features to resolve the negation scope. As an alternative, Sergeeva et al. (2019) proposed a Recurrent Neural Network-based approach that utilizes unsupervised word-embeddings and character-level word-embeddings with pre-computed syntactic features as inputs for a bidirectional Long Short-Term Memory (BiLSTM)-layer. The bidirectional character of the LSTM-layer ensures that negation scope parts located at both sides of the negation cue can be captured. A similar approach was adopted by Fancellu et al. (2016), who evaluated a BiLSTM-based approach in comparison to a feed forward Neural Network model, that both integrated pre-specified information about the negation cue and part-of-speech (PoS) tags. All of these models perform at a state-of-the art level.

1.2 The current study

Based on the BiLSTM-model proposed by (Fancellu et al., 2016), the current study adopts a BiLSTM-approach with sentence and cue embedding inputs to perform the task of negation scope detection on biomedical text data. In contrast to Fancellu et al. (2016), both negated and non-negated sentences are used to train this model. Also, the model is slightly extended with an additional dense layer after the LSTM-layers. Furthermore, the current study tries to adapt to the problem described by Fancellu et al. (2017), who suggested that negation scope detection models might overfit the phenomenon that negation scopes are often enclosed by punctuation marks. In this way, this study contributes to the knowledge about Neural Network approaches to resolve negation scopes in natural language.

This report should be read as a intermediate documentation of the current study. In the Methods section, the basic model architecture and data flow are described. The Results section reports on the performance of this model in one specific setting of hyperparameters. In the Discussion section, possible ways to extend the current study are considered.

2 Methods

2.1 Data

The current study made use of the biological full papers and biological scientific abstracts subcorpora from the freely available BioScope corpus (Vincze et al., 2008). In this corpus, speculative and negative keywords and their linguistic scope are annotated for each sentence. These annotions were carried out such that the set of keywords in a sentence is as small as possible, and the corresponding scope is as wide as possible. This results in the fact that none of the negation scopes are discontinuous. In this study, only the negation instances are of interest. A small description of the data set is provided in Table 1.

Data preprocessing. The data set consists of 14462 sentences with their neagtion cue and negation scope annotations. After standardization and tokenization, a sentence s_i is represented by a vector $\mathbf{x}_i = (x_{1_i} \ \cdots \ x_{n_i})'$ of n_i tokens, $i \in \{1, 2, \ldots, 14462\}$. In the current study, hyphenated words were considered as one token, and commas and periods were excluded. This resulted in 17602 unique tokens. Let the vocabulary $X = \{x_1, \ldots, x_{17602}\}$ be their ordered set and $V = \{1, \ldots, |X|\}$ the set of corresponding indices. Then, a token vector $\mathbf{x}_i$ can be vectorized by mapping each x_{t_i} to its vocabulary index $v_{t_i} \in V$, $t = 1, 2, \ldots, n$, obtaining a vectorized sentence $\mathbf{v}_i = (v_{1_i} \ \cdots \ v_{n_i})'$ for each s_i, which will be referred to as a sequence.

Each sequence $\mathbf{v}_i$ is associated with a cue indicator vector $\mathbf{c}_i = (c_{1_i} \ \cdots \ c_{n_i})' \in \{0, 1\}^{n_i}$, where $c_{t_i} = 1$ if the t-th token from the corresponding token vector $\mathbf{x}_i$ is annotated as

Table 1: Descriptives of the corpora

	Abstracts	Full Papers	Total
# Sentences	11993	2469	14462
# Negation instances	1719	375	2094
% Negation instances	14.3	15.2	14.5

a negation cue and $c_{t_i} = 0$ otherwise, $t = 1, 2, \ldots, n$. The negation scope of sentence s_i is defined as $\mathbf{y}_i = (y_{1_i} \ \cdots \ y_{n_i})' \in \{0,1\}^{n_i}$, where $y_{t_i} = 1$ if the t-th token from $\mathbf{x}_i$ is annotated as a negation cue or affected by it, and $y_{t_i} = 0$ otherwise. This vector is referred to as the label vector.

The average length of a sentence was 23.67 tokens and the average scope length[1] was 8.38 tokens, where the scope length of $\mathbf{y}_i$ is defined as $\sum_{t=1}^{n} y_{t_i}$. For model training, 11500 randomly selected (sequence, cue, label)-triples were used, which corresponds to about 80% of the data. The remaining triples were used for testing.

2.2 Task modeling

The task is to predict the label vector $\mathbf{y}_i$ given $\mathbf{v}_i$ and $\mathbf{c}_i$. As an example from the data set, sentence s_{169},

This element has no activity in T cells of the Jurkat line,

is labeled as

This/0 element/0 has/0 no/1 activity/1 in/1 T/1 cells/1 of/1 the/1 Jurkat/1 line/1.

So, the task in this example is to predict $\mathbf{y}_{169} = (0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1)'$ given $\mathbf{v}_{169} = (24\ 90\ 75\ 114\ 36\ 3\ 21\ 8\ 2\ 1\ 177\ 122)'$ and $\mathbf{c}_{169} = (0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)'$. To this end, each token is given a predicted probability that it is inside the negation scope. These probabilities are then concatenated into a single vector to predict the label vector.

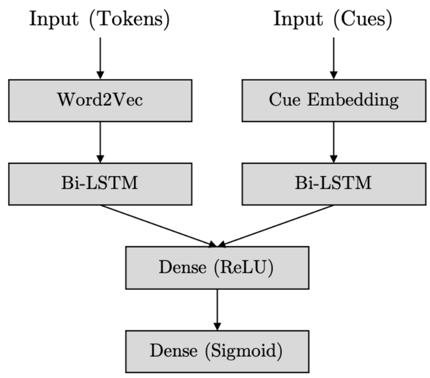

Figure 1: Schematic representation of the model.

2.3 Model architecture

Negation scope classification for each sentence was performed by a neural network consisting of an embedding layer, a Bi-LSTM layer and two dense layers. As inputs, the model takes a sequence and its corresponding negation cue vector. These are feeded into their own embedding layer and Bi-LSTM layer, subsequently concatenated, feeded through a dense layer with a ReLU activation and finally into another dense layer, the output layer. See Figure 1. The values from the output layer can be interpreted as the probability for each token of being part of the negation scope. The following subsub-

[1] average scope length of scopes that were longer than zero

sections describe how a sequence $\mathbf{v}$ and a cue vector $\mathbf{c}$ are processed through the layers of the model[2].

Embedding layers. Within a sequence $\mathbf{v}$, a 50-dimensional word embedding $\mathbf{e}_t \in \mathbb{R}^{50}$ is obtained for each v_t using a Word2Vec model (Mikolov et al., 2013). Word embeddings were trained on the entire data set with the skip-gram algorithm, were the task is to correctly identify the context word of a target word from a set of random negative samples and the true context word. Here, a context window size of 5 and a negative sample size of 15 were used. An embedded sentence can then be represented as a $d \times n$ matrix $E = (\mathbf{e}_1 \ \cdots \ \mathbf{e}_n)$. Word embeddings were not updated during training of the model on the negation scope detection task.

For each c_t within a cue vector $\mathbf{c} = (c_1 \ \cdots \ c_n)'$, the cue embedding layer yields a cue embedding $\mathbf{q}_t \in \{1\}^{50}$ if $c_t = 1$ and $\{0\}^{50}$ if $c_t = 0$. Then, the cue embedding of sentence s can be represented as a $d \times n$ matrix $Q = (\mathbf{q}_1 \ \cdots \ \mathbf{q}_n)$. The cue embedding layer was specified to be non-trainable, since the cue embeddings should function in the same way for each $\mathbf{c}$.

Bi-LSTM layers. The embedded sentence E and the embedded negation cue Q are passed to their own Bi-LSTM layer (Hochreiter & Schmidhuber, 1997). Both layers consist of 64 units: 32 units in the forward direction where the time-step t moves from 1 to n and 32 units in the backward direction where t moves from n to 1. Time-step t corresponds to the t-th column of the input matrix. In this subsection, only the Bi-LSTM layer with input E will be discussed, since the layer with input Q is designed identically.

Let $\mathbf{i}_t, \mathbf{f}_t, \mathbf{o}_t$ and $\mathbf{h}_t$ denote the input gate, forget gate, output gate and hidden layer state of the LSTM-cell, respectively, at time-step t. Let W_e denote the weight matrices of the embedding input and W_h the weight matrices of the previous hidden state, and let superscripts $(i), (f)$ and (o) indicate to which gate the matrix belongs. Let $\mathbf{b}$ be the bias vectors with similar superscripts. Then the input, forget and output gate of the Bi-LSTM layer can be described as

$$\begin{aligned}
\mathbf{i}_t &= \sigma\big(W_e^{(i)}\mathbf{e}_t + W_h^{(i)}\mathbf{h}_{t-1} + \mathbf{b}^{(i)}\big),\\
\mathbf{f}_t &= \sigma\big(W_e^{(f)}\mathbf{e}_t + W_h^{(f)}\mathbf{h}_{t-1} + \mathbf{b}^{(f)}\big),\\
\mathbf{o}_t &= \sigma\big(W_e^{(o)}\mathbf{e}_t + W_h^{(o)}\mathbf{h}_{t-1} + \mathbf{b}^{(o)}\big),
\end{aligned}$$

where the sigmoid function $\sigma : \mathbb{R} \to (0,1)$ is given by $x \mapsto \frac{1}{1+e^{-x}}$.

The cell state $\boldsymbol{\gamma}$ at time-step t is a combination of the previous cell state $\boldsymbol{\gamma}_{t-1}$ and the candidate cell state $\tilde{\boldsymbol{\gamma}}$:

$$\boldsymbol{\gamma}_t = \mathbf{f}_t * \boldsymbol{\gamma}_{t-1} + \mathbf{i}_t * \tilde{\boldsymbol{\gamma}}_t,$$

where '$*$' denotes element-wise multiplication and the candidate cell state is defined as

$$\tilde{\boldsymbol{\gamma}}_t = \tanh(W_e^{(\tilde{\gamma})}\mathbf{e}_t + W_h^{(\tilde{\gamma})}\mathbf{h}_{t-1} + \mathbf{b}^{(\tilde{\gamma})}),$$

where the hyperbolic tangent function tanh : $\mathbb{R} \to (-1,1)$ is given by $x \mapsto \frac{e^x - e^{-x}}{e^x + e^{-x}}$. Finally, the hidden-layer state $\mathbf{h}$ at time-step t, which is the output vector for token t, can be com-

[2]The subscript i will be omitted in this section, for purposes of clearness.

puted by

$$\mathbf{h}_t = \mathbf{o}_t * \tanh(\boldsymbol{\gamma}_t).$$

Each token and each cue is associated with one output vector with a length equal to the number of units from the Bi-LSTM layer. So, each token and each cue is represented by a 64-dimensional vector. The outputs of both Bi-LSTM layers were subsequently concatenated to yield an output $\mathbf{h}_t^{(c)} \in \mathbb{R}^{128}$ for each combination of a token and its corresponding cue indicator.

Dense layers. The concatenated output vectors of the Bi-LSTM layers were feeded to a 64-unit fully connected feed-forward neural network, the first dense layer. In this layer, a rectified linear unit (ReLU) activation function $\mathbb{R} \to [0, \infty)$ was used, which is defined as $x \mapsto 0$ if $x \leq 0$ and $x \mapsto x$ if $x > 0$. The output $\boldsymbol{\omega}_t \in \mathbb{R}^{64}$ for token t in this layer is given by

$$\boldsymbol{\omega}_t = \mathrm{ReLU}\big(W_h^{(d)}\mathbf{h}_t^{(c)} + \mathbf{b}^{(d)}\big),$$

where the superscript (d) denotes that the parameter vectors belong to the dense layer.

Eventually, $\boldsymbol{\omega}_t$ was feeded into a one-unit dense layer with the sigmoid activation function σ defined as before, yielding the predicted probability π that the t-th token is inside the negation scope:

$$\pi_t = \sigma\big(\mathbf{w}_y^{(s)} \cdot \boldsymbol{\omega}_t + b^{(s)}\big),$$

where $\mathbf{w}_y^{(s)}$ is the weight vector and $b^{(s)}$ is a bias scalar, with (s) denoting their association with the sigmoid dense layer. Concatenation of all π_t his yields the output vector $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)'$ for sentence s.

2.4 Model compiling and fitting

The model was compiled and fitted with the Keras functional API for Tensorflow 2.3.1 in Python 3.7.6 (Abadi et al., 2015; Van Rossum & Drake, 2009), using the Adam optimizer. The model was trained with 3 epochs on 80% of the training data, reserving 20% for validation data. Mini-batch gradient descent with a batch size of 32 was used to maximize the likelihood $\mathcal{L}(\theta)$ of the correct predictions $\boldsymbol{\pi}$ compared to the true labels $\mathbf{y}$, given θ as a shorthand symbol for all model parameters:

$$\mathcal{L}(\theta) = \prod_{t=1}^{n} p_t^{y_t}(1 - p_t)^{1-y_t}.$$

The value of this function was maximized by minimizing its negative log-likelihood, divided by the length of the sentence:

$$L(\theta) = \frac{-1}{n}\sum_{t=1}^{n} y_t \log(p_t) + (1 - y_t)\log(1 - p_t).$$

2.5 Performance evaluation

The performance of the model was evaluated at the token level and at the sentence level, using the test data. At the token level, precision, recall and the F_1-measure were used as indicators of performance[3]. To this end, a classification vector $\mathbf{p}_i = (p_{1_i} \ \cdots \ p_{n_i})'$ for sentence s_i is element-wise compared to the label vector $\mathbf{y}_i$, where the classification p_{t_i} equals 1 if $\pi_{t_i} \geq .5$ and 0 if $\pi_{t_i} < .5$.

[3]Precision is defined as $\frac{\text{tp}}{\text{tp+fp}}$, recall as $\frac{\text{tp}}{\text{tp+fn}}$ and F_1 as $\frac{2\cdot\text{precision}\cdot\text{recall}}{\text{precision+recall}}$, where tp = true positives, fp = false positives and fn = false negatives

True Positive		**False Negative**	**False Positive**	**True Negative**

1.1 Exact match
`Y = 0 1 1 1 0`
`C = 0 1 1 1 0`
	1.2 Too wide
	`Y = 0 1 1 1 0`
	`C = 1 1 1 1 1`
		`Y = 0 1 1 1 0`
		`C = 0 0 0 0 0`
			`Y = 0 0 0 0 0`
			`C = 0 1 1 1 0`
				`Y = 0 0 0 0 0`
				`C = 0 0 0 0 0`

1.3 Too narrow
`Y = 0 1 1 1 0`
`C = 0 0 1 0 0`
	1.4 Too early
	`Y = 0 1 1 1 0`
	`C = 1 1 1 0 0`
		Complete miss
		`Y = 0 1 1 1 0`
		`C = 1 0 0 0 0`
			Complete miss
			`Y = 0 1 1 1 0`
			`C = 0 0 0 0 1`	

1.5 Too late
`Y = 0 1 1 1 0`
`C = 0 0 1 1 1`
	2 Discontinuous
	`Y = 0 1 1 1 0`
	`C = 0 1 0 1 0`			

Figure 2: Different types of predictions at the scope level. Note: Y denotes the vector of true scope labels, C denotes its classification vector. These are hypothetical examples for a sentence with 5 tokens.

At the sentence level, a classification vector is regarded a true positive if it contains at least one true positive prediction at the token level, following the guidelines of the *SEM Shared Task 2012 (Morante & Blanco, 2012). This is the case when there is a negation scope, and at least one of the scope tokens is detected. False negatives are those instances where there is a negation scope, but none of the scope tokens is detected. True negatives are those instances where a sentence does not contain a negation scope, and the classification vector predicts no negation scope. Finally, false positives are those instances where each *positive* prediction at the token level is a false positive, regardless of whether there is a negation scope. A special case is when there is a negation scope, the classification vector predicts a scope but none of the true negation tokens are detected. This is referred to as a *complete miss.*

To gain more insight into the true positives, which include any partially correct prediction, a distinction is made between (1) continuous scope predictions and (2) discontinuous scope predictions. A discontinuous scope prediction is a classification vector that contains the sequence $(1\ 0^*\ 1)'$, where 0^* denotes a sequence of one or more zeros. The continuous scope predictions are further divided into (1.1) exact scope matches, and scope predictions that are (1.2) too wide, (1.3) too narrow, (1.4) too early or (1.5) too late. See Figure 2 for an overview of all types of predictions at the scope level.

At the scope level, too, precision, recall and the F_1-measure were computed.

3 Results

The model performs reasonably well at the token level: The precision is 80.0%, the recall is 74.7% and the F_1 is 77.3%. At the scope level, the model performs very well: The precision is 100.0%, the recall is 96.9% and the F_1 is 98.4%, see Table 2. Furthermore, none of the false negatives were complete misses. Most of the true positives (43.5%) are exact matches, see Table 3.

Table 3: Distribution of true positives at the scope level

Table 2: Model performance at the token level and the scope level.

	Total	tp	fn	tn	fp	Precision	Recall	F_1
Tokens	70496	2,575	874	66,405	642	80.0%	74.7%	77.3%
Scopes	2962	405	13	2544	0	100.0%	96.9%	98.4%

Note: tp = true positives, fn = false negatives, tn = true negatives and fp = false positives.

Type of true positive	Percentage
1.1 Exact matches	43.5%
1.2 Too wide	27.9%
1.3 Too narrow	22.7%
1.4 Too early	0%
1.5 Too late	4.7%
2 Discontinuous	1.2%

4 Discussion

At the scope level, the proposed model achieves an excellent performance, comparable to state-of-art methods. At the token level, however, the model slightly underperforms compared to the model proposed by Fancellu et al. (2016). In this section, possible stategies to extend this study are discussed.

First, experiments with different hyperparameter settings could optimize the performance of the model. For example, the threshold of a positive prediction could be decreased, given the fact that the recall is rather low at the token level and many of the scope predictions were too wide.

Second, the scope predictions could be improved by applying a 'continuizer' to them, given the fact that all true scopes are continuous. This could be a simple rule that converts discontinuous scope segments $(1\ 0 *\ 1)'$ to $(1\ 1 *\ 1)'$. For the current model, discontinuous scope predictions are not common, but these may be more prevalent with a lower threshold setting.

Third, a part-of-speech tagger could be added to provide the model with more syntactic information, following Sergeeva et al. (2019) and Fancellu et al. (2016). Also, the model could benefit from another source of external information by adding pre-trained word embeddings, for example, contextualized word embeddings from ELMo (Peters et al., 2018).

Finally, consistent with the goal to be less dependent on manually-crafted features, information about the negation cues can be predicted from a preceding part of the model.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Retrieved from `https://www.tensorflow.org/` (Software available from tensorflow.org)

Basile, V., Johan, B., Kilian, E., & Noortje, V. (2012). Ugroningen: Negation detection with discourse representation structures. In *First joint conference on lexical and computational semantics (* sem)* (pp. 301–309).

Fancellu, F., Lopez, A., & Webber, B. (2016). Neural networks for negation scope detection. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: long papers)* (pp. 495–504).

Fancellu, F., Lopez, A., Webber, B., & He, H. (2017). Detecting negation scope is easy, except when it isn't. In *Proceedings of the 15th conference of the european chapter of the association for computational linguistics: Volume 2, short papers* (pp. 58–63).

Fei, H., Ren, Y., & Ji, D. (2020). Negation and speculation scope detection using recursive neural conditional random fields. *Neurocomputing*, *374*, 22–29.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Mikolov, T., Corrado, G., Chen, K., & Dean, J. (2013, 01). Efficient estimation of word representations in vector space. In (p. 1-12).

Morante, R., & Blanco, E. (2012). *sem 2012 shared task: Resolving the scope and focus of negation. In *Proceedings of the first joint conference on lexical and computational semantics - volume 1: Proceedings of the main conference and the shared task, and volume 2: Proceedings of the sixth international workshop on semantic evaluation* (p. 265–274). USA: Association for Computational Linguistics.

Morante, R., & Sporleder, C. (2012). Modality and negation: An introduction to the special issue. *Computational Linguistics*, *38*(2), 223-260. doi: 10.1162/COLI_a_00095

Øvrelid, L., Velldal, E., & Oepen, S. (2010, August). Syntactic scope resolution in uncertainty analysis. In *Proceedings of the 23rd international conference on computational linguistics (coling 2010)* (pp. 1379–1387). Beijing, China: Coling 2010 Organizing Committee. Retrieved from `https://www.aclweb.org/anthology/C10-1155`

Özgür, A., & Radev, D. R. (2009, August). Detecting speculations and their scopes in scientific text. In *Proceedings of the 2009 conference on empirical methods in natural language processing* (pp. 1398–1407). Singapore: Association for Computational Linguistics. Retrieved from `https://www.aclweb.org/anthology/D09-1145`

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018, June). Deep contextualized word representations. In *Proceedings of the 2018 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long papers)* (pp. 2227–2237). New Orleans, Louisiana: Association for Computational Linguistics. Retrieved from `https://www.aclweb.org/anthology/N18-1202` doi: 10.18653/v1/N18-1202

Qian, Z., Li, P., Zhu, Q., Zhou, G., Luo, Z., & Luo, W. (2016, November). Speculation and negation scope detection via convolutional neural networks. In *Proceedings of the 2016 conference on empirical methods in natural language processing* (pp. 815–825). Austin, Texas: Association for Computational Linguistics. Retrieved from `https://www.aclweb.org/anthology/D16-1078` doi: 10.18653/v1/D16-1078

Sergeeva, E., Zhu, H., Prinsen, P., & Tahmasebi, A. (2019). Negation scope detection in clinical notes and scientific abstracts: A feature-enriched lstm-based approach. *AMIA Joint Summits on Translational Science proceedings. AMIA Joint Summits on Translational Science*, *2019*, 212—221. Retrieved from `https://europepmc.org/articles/PMC6568093`

Tang, B., Wang, X., Wang, X., Yuan, B., & Fan, S. (2010, July). A cascade method for detecting hedges and their scope in natural language text. In *Proceedings of the fourteenth conference on computational natural language learning – shared task* (pp. 13–17). Uppsala, Sweden: Association for Computational Linguistics. Retrieved from `https://www.aclweb.org/anthology/W10-3002`

Tottie, G. (1991). *Negation in english speech and writing: A study in variation* (Vol. 4). Academic Pr.

Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual*. Scotts Valley, CA: CreateSpace.

Vincze, V., Szarvas, G., Farkas, R., Móra, G., & Csirik, J. (2008, November). The BioScope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC Bioinformatics*, *9*(11), S9. Retrieved from `https://doi.org/10.1186/1471-2105-9-S11-S9` doi: 10.1186/1471-2105-9-S11-S9

Zou, B., Zhou, G., & Zhu, Q. (2013, October). Tree kernel-based negation and speculation scope detection with structured syntactic parse features. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 968–976). Seattle, Washington, USA: Association for Computational Linguistics. Retrieved from `https://www.aclweb.org/anthology/D13-1099`