Preparing Stance Detection
Feature Extraction Methods and Their Performance Used for Feature-Based Machine Learning Algorithms

Kristóf Vass , **Pradeep Murukannaiah**
TU Delft

Abstract

Stance detection is a Natural Language Processing task that aims to detect the stance (support, agreement, or disagreement) of a piece of text towards some target. In this paper, we aim to find the best performing pair of feature extraction method and feature-based machine learning algorithm. By doing so, an explainable method can be found to show how to solve stance detection problems. After researching the most common techniques, twenty different combinations are evaluated. We have found that the best performing pair is Word N-gram used with Logistic Regression, which achieves an F-score of 0.599 and an accuracy of 0.66.

1 Introduction

Stance detection is a Natural Language Processing (NLP) task used to compare text-target pairs and determine their relation. This relation can usually be selected from the following label set: {*Favor, Against, Neutral, Neither*} [Küçük and Can, 2020]. An example of a target and texts with different stances from this set can be found in Table 1. Further, with the help of different labels, such as {*Agrees, Disagrees, Discuss, Unrelated*}, it is also possible to decide whether an article's statement agrees with the topic, helping to decide whether something is considered fake [Küçük and Can, 2020; FNC, 2017].

This is important, as the spread of fake news and disinformation on the internet has been a severe issue. It makes it difficult for users to decide if the news can be trusted. The most popular social media sites, such as Twitter and Facebook, are fighting this problem, however, their tools are sometimes manual. For instance, Twitter is applying a community-based approach to find misinformation [Coleman, 2021], therefore, this can be slow and biased. To solve this issue, Stance Detection can be used [Küçük and Can, 2020]. By automating the process and moving the decisions to algorithms, the bias can be reduced as humans do not have to make classification decisions by themselves.

Recently, there have been several works on stance detection. These works discuss their own specific implementation of stance detection, such as Mohammad *et al.* [2016], Wei *et al.* [2016], Sun *et al.* [2016], and Patra *et al.* [2016]. Additionally, there are other works that summarise such algorithms, providing overviews, benchmarks and surveys, such as Küçük and Can [2020], AlDayel and Magdy [2021], and Ghosh *et al.* [2019].

Table 1: Examples of texts and their stance towards a target [Mohammad *et al.*, 2016]

Target: Climate change is a real concern	
Favor	Today Europe is breaking heat records, while Asia is breaking the lowest temperature records!
Against	ONE Volcano emits more pollution than man has in our HISTORY!
Neutral	Climate change is currently a hot topic to talk about.
Neither	The stock market froze in the summer?!

Stance Detection is usually implemented in two different ways [Küçük and Can, 2020]: using either feature-based machine learning algorithms or using deep learning methods. As described in the previous paragraph, many of the aforementioned papers create a summary of existing implementations. However, previous research does not attempt to combine the different elements of the algorithms to find the best performing one. Therefore, in this research, we strive to identify a pair of feature extraction method and feature-based machine learning algorithm that yields the highest performance in stance detection. We chose to research feature-based machine learning algorithms (instead of deep learning models, which are gaining popularity in the past few years) because of their interoperability and easy explainability [Xu *et al.*, 2019]. By these, we hope to understand and explain the results of our research better.

We seek to answer the following research question:

How do popular feature extraction methods compare to each other when using them with feature-based machine learning algorithms for stance detection?

In order to provide a detailed answer to this question, we seek to answer a set of sub-questions:

1. What are the most commonly used feature extraction methods for stance detection?
2. What are the most commonly used feature-based machine learning algorithms for stance detection?
3. How should the performance of the algorithms for stance detection be measured?
4. How do different feature extraction methods compare to each other when using them on one specific model for stance detection?
5. Which combination of feature extraction method and feature-based machine learning algorithm performs the best for stance detection?

The paper is structured as follows. Section 2 discusses the tasks we have done to answer the sub-questions, gives a deeper understanding of the theory and the applied techniques, and the collected feature extraction methods and feature-based machine learning algorithms. In Section 3, we discuss the experimental setup, the dataset, the performance measure, and the actual implementations of the combined methods. Section 4 presents the results of the experiment. Next, in Section 5, we discuss these results. Then, Section 6 concludes the paper by answering the (sub-)questions and describing future work. Lastly, in Section 7, we address the ethical aspects of the research.

2 Method and Theoretical Background

First, this section describes the tasks performed in order to answer the sub-questions of the research. Then, based on the conducted research, it describes the most commonly used feature extraction methods and feature-based machine learning algorithms used for stance detection to better understand what is implemented later.

2.1 Tasks to answer sub-questions

In order to answer the previously mentioned sub-questions, we undertake the following tasks:

1. Find existing implementations and research about stance detection (comprehensive literature study).
   - The literature study can be found in Section 1.
2. Find at least three and a maximum of six feature extraction methods that can be used for stance detection and implement them.
   - The found methods are shown and explained in Section 2.2.
3. Find one dataset that can be used for all data-extraction methods.
   - The selected dataset and its description can be found in Section 3.2.
4. Find at least two and a maximum of four feature-based machine learning algorithms that can be used for stance detection and implement them in combination with the feature extraction methods.
   - The found methods are shown and explained in Section 2.3.
5. Based on the literature study, identify the most commonly used and descriptive performance measure for stance detection.
   - The selected performance measures and their description can be found in Section 3.3
6. Identify the best-performing combination and conclude the research.
   - The result of this task can be found in Section 4.

2.2 Feature extraction methods

In order to use traditional (feature-based) machine learning to evaluate stance detection tasks, a feature extraction method has to be used. During the feature extraction, we must convert the corpus to a vector representation. The corpus is a collection of documents where each document contains terms. We can perform this conversion in different ways. A high-level overview and the division of methods used, based on the evaluation of forty-three papers, can be found in Figure 1. The list of reviewed papers can be found in Appendix A. This section introduces some of the most commonly used methods implemented and their way of working.

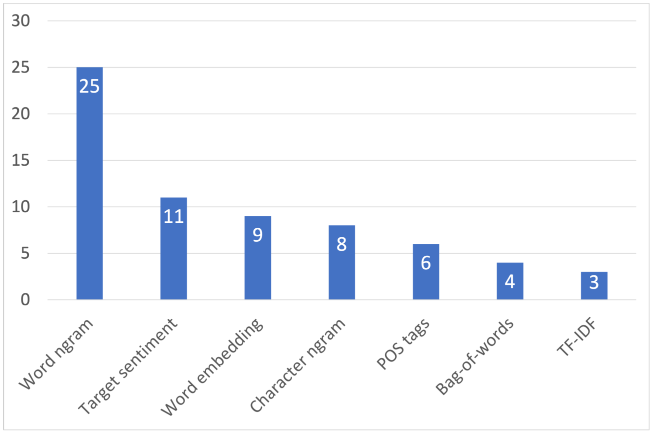


Figure 1: Feature extraction methods

Word N-grams and bag-of-words

The first feature extraction method under review is word N-grams and bag-of-words. Word N-grams are N consecutive words created from a text [Jurafsky and Martin, 2020]. Take the following sentence as an example: *"I like dogs and cats, too."*. To create N-grams for this sentence, see the following examples:

Example 1 (1-gram (unigram)) *"I", "like", "dogs", "and", "cats", "too"*

Example 2 (2-gram (bigram)) *"I like", "like dogs", "dogs and", "and cats", "cats too"*

Example 3 (3-gram (trigram)) *"I like dogs", "like dogs and", "dogs and cats", "and cats too"*

Many times, bigrams and trigrams hold more information than unigrams. For instance, in the previous example, the trigram *"dogs and cats"* shows that the target of the sentence is both animals, while unigrams cannot provide this information. Therefore, using multigrams can improve performance in many cases.

In the bag-of-words model, the "bag" contains all N-grams of the original corpus and is represented as a vector. After creating this vector, a sentence can be represented by showing how many times an element appears in the original vector.

Word embedding

Word embedding is the vectorial representation of words in a way that the result vector represents the meaning of a word. The embedding model is trained such that (contextually) similar words are closer to each other in the vector space [Jurafsky and Martin, 2020]. A word embedding method is word2vec, which is an efficient method to create such vectors. Word2vec was created by Google and it is pre-trained from the Google News corpus, which contains more than three billion words [Mikolov *et al.*, 2013]. When using word2vec, we represent every word as a 300-dimension vector. Figure 2 shows a simplified, high-level overview of such a vector representation of words. This figure shows how the values are distributes within the vector (red means high value, blue means low value), and that "Man" and "Woman" are closer to each other in the vector space than to "king", meaning that they are more related in meaning/association, too [Alammar, 2018].

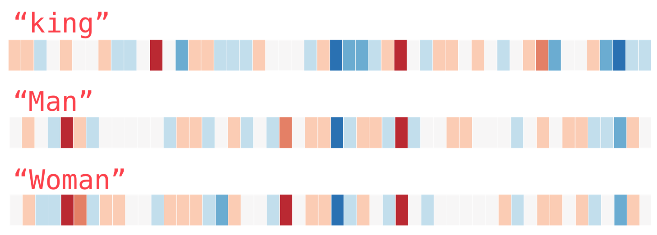


Figure 2: Word2vec representation of words [Alammar, 2018]

POS tags

Part-of-speech (POS) tags are the grammatical tags of every word in a sentence. These tags can be "adjective", "verb", "noun", etc. By using POS tags, one can give a different meaning to words. By using POS tags, further context is applied to the basic unigram bag-of-words approach. For example, the algorithm can identify differences between using "catch" as a noun or a verb.

TF-IDF

Term frequency-inverse document frequency (TF-IDF) is the numerical representation of how important a word is in a corpus [Rajaraman and Ullman, 2011]. Term frequency is "the weight of a term that occurs in a document" [Luhn, 1957]. However, since some words, such as "and" and "the", are common in many sentences, TF can be misleading. To tackle this, IDF can be used. IDF "can be quantified as an inverse function of the number of documents in which it occurs" [Jones, 1972].

TF-IDF is the product of the two values mentioned above, thus giving as many insights into the text as possible. TF can be calculated the following way:

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{1}$$

where *f(t,d)* is the number of times that the term *t* occurs in document *d*. IDF can be calculated the following way:

$$idf(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{2}$$

where N is the number of documents in the corpus, and the denominator is the number of documents where the term t appears. In order to calculate TF-IDF, the product of the TF and IDF has to be calculated:

$$tfidf(t,D) = tf(t,d) * idf(t,D) \tag{3}$$

2.3 Feature-based machine learning algorithms

After the feature extraction is done, we can apply traditional machine learning to the result vector. Based on the conducted literature study and the survey created by Küçük and Can [2020], the overview of the most common models can be found in Figure 3. All these models—SVM, Logistic Regression, Random Forest, Naïve Bayes and Decision Tree [Bishop, 2006]—are supervised machine learning algorithms used for classification. By giving a set of labelled training dataset to the algorithms, they can learn to predict the class of any new data point. Except for the Decision Tree, all most commonly used models have been implemented with the help of the scikit-learn library [Pedregosa *et al.*, 2011]. The reason we excluded the Decision Tree is because we decided to implement a maximum of four methods, as mentioned in Section 2, and the Decision Tree was used the fewest times based on the literature study.

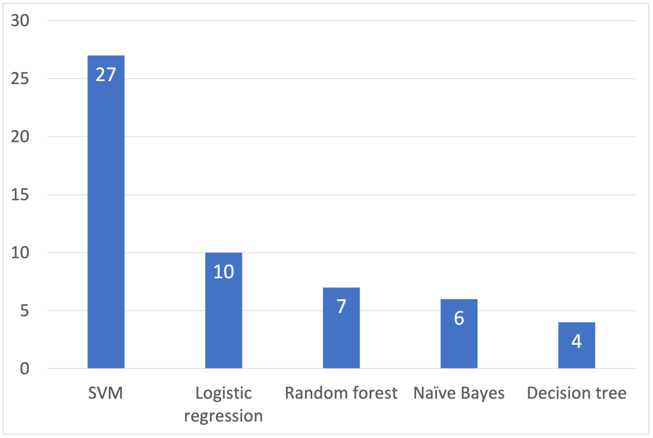
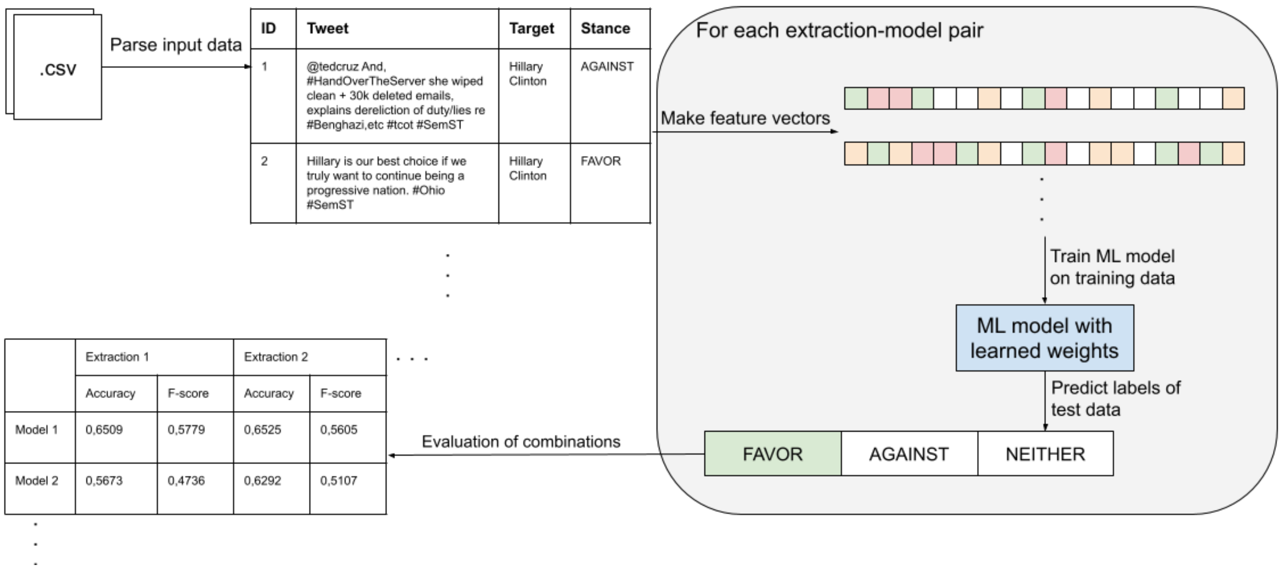


Figure 3: Feature-based machine learning algorithms

3 Experiment

This section describes the actual implementation of the aforementioned methods and shows the experiment's pipeline. Next, it introduces the dataset used for the evaluation and shows how the performance is measured. Lastly, it shows how the different elements are combined to run the experiment.


.CSV
Parse input data
For each extraction-model pair
Make feature vectors
Train ML model on training data
ML model with learned weights
Predict labels of test data
FAVOR
AGAINST
NEITHER
Evaluation of combinations

ID	Tweet	Target	Stance
1	@tedcruz And, #HandOverTheServer she wiped clean + 30k deleted emails, explains dereliction of duty/lies re #Benghazi,etc #tcot #SemST	Hillary Clinton	AGAINST
2	Hillary is our best choice if we truly want to continue being a progressive nation. #Ohio #SemST	Hillary Clinton	FAVOR

	Extraction 1		Extraction 2	
	Accuracy	F-score	Accuracy	F-score
Model 1	0,6509	0,5779	0,6525	0,5605
Model 2	0,5673	0,4736	0,6292	0,5107

Figure 4: Visual representation of the experimental pipeline

3.1 Experimental setup

The setup of this experiment consists of 6 steps:

1. Parse input data.
2. For every feature extraction method - machine learning algorithm pair:
   (a) make feature vectors from both training and test data,
   (b) train machine learning algorithm on training data,
   (c) perform classification on test data.
3. Evaluation of the different combinations.

The aforementioned steps are explained later in the section. The visual representation of the pipeline can be found in Figure 4. The experiment is run on a MacBook Pro with 8GB memory and a 2.6 GHz Dual-Core Intel Core i5 processor, running macOS 11.3. The Python version used is 3.8.7. The libraries used and their version number can be found in Table 7 in Appendix B. Any library's method is used with default parameters or stated otherwise. Further, the whole codebase can be found on the EEMCS GitLab repository[1].

3.2 Dataset

One of the earliest competitions on stance detection was the SemEval-2016 shared task on Twitter stance detection, described by Mohammad *et al.* [2016]. During this competition, the goal was to determine the stance of tweets towards five targets, with three labels. The dataset created for this competition is a set of tweets manually annotated for stance towards a given target. The five targets are the following: Atheism (AT), Climate Change is a Real Concern (CC), Feminist Movement (FM), Hillary Clinton (HC), and Legalization of Abortion (LA). The labels are the following: Favor (F), Against (A), Neither (N). It contains around 2900 training instances for the five targets.

We chose this dataset because several other papers and implementations use it, which helps to indicate the expected performance. Further, analysing tweets is a problem that can be seen as a real-life use case of such algorithms.

The division of topics and the split of the training and test instances can be found in Table 2. Further, some examples from the dataset can be found in Table 3, showing the target, the original tweet and the related stance.

Table 2: Division of topics within the training and test sets.

Topic	Training			Test		
	F	A	N	F	A	N
AT	92	304	117	32	160	28
CC	212	15	168	123	11	35
FM	210	328	126	58	183	44
HC	118	393	178	45	172	78
LA	121	355	177	46	189	45
	753	***1395***	***766***	***304***	***715***	***230***
SUM		***2914***			***1249***	
			4136			

3.3 Performance measure

To measure the performance of the implementation, both accuracy and F-score are used. As most of the previous implementations use F-score as their primary evaluation metric, it is reasonable to use it because existing research can be used for comparison. Further, accuracy is also used because it is an easily explainable and commonly known measure.

F-score is a statistical measure calculated from precision (P) and recall (R) of the test. The graphical representation of the points used in the F-score can be found in Figure 6

[1]https://gitlab.ewi.tudelft.nl/cse3000/2020-2021/rp-group-65/rp-group-65-kvass.git

Table 3: Examples of the SemEval-2016 Stance dataset

ID	Target	Tweet	Stance
1	Hillary Clinton	@tedcruz And, #HandOverTheServer she wiped clean + 30k deleted emails, explains dereliction of duty/lies re #Benghazi,etc #tcot #SemST	Against
2	Hillary Clinton	Hillary is our best choice if we truly want to continue being a progressive nation. #Ohio #SemST	Favor
52	Legalization of Abortion	@tooprettyclub Are you OK with #GOP males telling you what you can and can't do with your own body? #SemST	Favor

in Appendix C. By the definition of Rijsbergen [Rijsbergen, 1979], it is calculated as follows:

$$P = \frac{Correct}{Correct + Spurious} \tag{4}$$

$$R = \frac{Correct}{Correct + Missing} \tag{5}$$

$$F = \frac{2 * P * R}{P + R} \tag{6}$$

To apply this measure to stance detection, we can use the macro-average of the F-scores of the three labels (*Favor*, *Against*, *Neither*). Macro-average is selected as it is the most commonly used average for stance detection tasks [Mohammad *et al.*, 2016; Küçük and Can, 2020]. By doing so, our F-score performance measure (*F*) is calculated the following way:

$$F_{Favor} = \frac{2 * P_{Favor} * R_{Favor}}{P_{Favor} + R_{Favor}} \tag{7}$$

$$F_{Against} = \frac{2 * P_{Against} * R_{Against}}{P_{Against} + R_{Against}} \tag{8}$$

$$F_{Neither} = \frac{2 * P_{Neither} * R_{Neither}}{P_{Neither} + R_{Neither}} \tag{9}$$

$$F = \frac{F_{Favor} + F_{Against} + F_{Neither}}{3} \tag{10}$$

Accuracy (A) is calculated the following way:

$$A = \frac{Correct\ classification}{All\ classification} \tag{11}$$

3.4 Parse input data

The complete annotated input data can be found and downloaded from the SemEval-2016 Stance Dataset website[2]. After downloading the dataset, we parse the train and test datasets, convert to *DataFrame*s, and remove the rows with missing values. Further pre-processing is conducted while creating the feature vectors, and they are explained in the later sections.

3.5 Implementation

As explained in Section 2.2, we implement five different feature extraction methods (TF-IDF, Word Embedding, Bag-of-words, Word N-gram, and POS tags) and four machine learning algorithms (SVM, Logistic Regression, Random Forest, and Naïve Bayes). The details of the implementations can be found in Appendix D.

4 Results

This section describes the experiment results, and shows the performance of the twenty different combinations of feature extraction methods and feature-based machine learning algorithms. The accuracy scores of the pairs can be found in Table 4, and the F-score can be found in Table 5.

Table 4: Accuracy scores of the pairs

	SVM	Logistic Regression	Random Forest	Naïve Bayes
TF-IDF	0.6173	0.6221	0.6589	0.6597
WE	0.6397	0.6045	**0.6661**	0.5749
BOF	0.5356	0.6157	0.6149	0.6477
Ngram	0.5869	0.6557	0.6197	0.6445
POS	0.6005	0.6229	0.6621	0.6557

As shown in Table 4, the best performing feature extraction method, when measured with accuracy, is TF-IDF, with an average score of 0.6395, while the most accurate machine learning algorithm (when using them with different feature extractions) is Random Forest with an average accuracy score of 0.6444. However, the best performing combination is not the combination of the most accurate methods, but the combination of using Word Embedding with a Random Forest classifier.

Table 5: F-scores of the pairs

	SVM	Logistic Regression	Random Forest	Naïve Bayes
TF-IDF	0.5663	0.5684	0.5500	0.4379
WE	0.5978	0.5396	0.5359	0.4726
BOW	0.5062	0.5686	0.5672	0.5418
Ngram	0.5467	**0.5991**	0.5741	0.5536
POS	0.5482	0.5649	0.4793	0.4674

[2]https://www.saifmohammad.com/WebDocs/stance-data-all-annotations.zip

In Table 5, the performance of the combinations, when measuring them with F-score, can be found. The best performing feature extraction method is Word N-grams with an average F-score of 0.5684. Additionally, the best result in terms of F-score for machine learning algorithms was achieved by Logistic Regression, with an average score of 0.5681. In this case, the best performing combination was the combination of the two, with an F-score of 0.5991.

As mentioned previously, F-score is commonly used in other papers that discuss the topic of stance detection. Further, when false results have to be taken into account, F-score gives a better indication (and penalty) than accuracy. Therefore, to select the best performing combination of feature extraction method and feature-based machine learning algorithm, when used on stance detection, F-score is used. As shown in the previous paragraph, the best combination is using Word N-grams (1-, 2-, and 3-grams) with Logistic Regression, which achieves an accuracy of 0.6557 and an F-score of 0.5991.

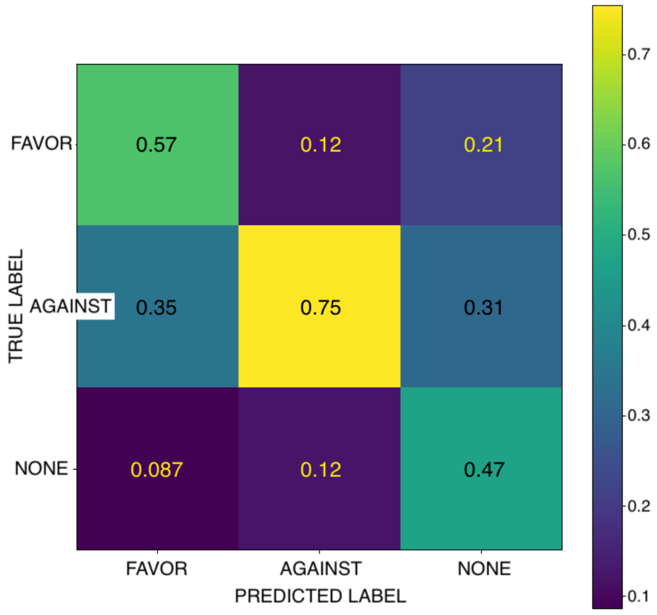


Figure 5: Normalised confusion matrix of the combination Word Ngram and Logistic Regression

Figure 5 shows the normalised confusion matrix of the aforementioned combination when evaluating the test dataset. This figure shows that the three highest values are on the diagonal, meaning that the test instances were predicted more correctly than incorrectly. Notably, the “AGAINST” label was predicted the most times correctly, with an accuracy of 0.75.

5 Discussion

This section discusses the results from Section 4. It explains the differences between the different methods and the conclusions derived from them. Then, it compares the best performing result to already existing implementations for further comparison. Lastly, it discusses the recommendations of the author and the limitations taken during the research.

5.1 Results of the different combinations

When inspecting the results of the evaluation, some conclusions can be drawn. The average accuracy of the different combinations is 0.6253 with a standard deviation of 0.034. The average F-score is 0.5393 with a standard deviation of 0.044. This means that the different pairs are very close to each other in performance, and the difference between them is negligible. It is also clear that there is a considerable difference between the results of accuracy and F-score, since the best result with accuracy is the seventeenth result in F-score, and the best result in F-score is eleventh with accuracy. Since the dataset is somewhat imbalanced, it is reasonable that F-score is lower than accuracy.

As discussed in Section 2, all feature extraction methods create different sizes of feature vectors. Therefore, they influence in the end results. As TF-IDF creates an extremely sparse and huge feature vector, it gives several insights into the corpus. However, this sparse vector seems to be too specific to give good results for general use cases. On the other hand, word embedding vectors are too small and dense to contain enough information about the corpus. It is also possible that normalising the vectors of all words into one vector results in the vector losing its meaning, since the result might be the representation of a (non-)existing word. The bag-of-words and Word N-gram methods create similar feature vector sizes, and their average accuracy is very similar in terms of both accuracy and F-score. Lastly, the method of POS tags creates a relatively dense feature vector again, however, this performs quite well for accuracy.

When considering accuracy, TF-IDF and POS tags performed the best. As previously mentioned, these two methods create the two sparsest feature vectors. Therefore, we can conclude that when accuracy is the primary performance measure, thus false positives and negatives do not play an important role in the evaluation, sparse vectors perform the best. On the other hand, when working with F-score, a lower dimensionality in the feature vector pays off. Lastly, if the feature vector size is too small, such as when using word embedding, traditional machine learning algorithms cannot perform very well.

5.2 Best result

As shown in Section 4, the best performing combination is using Word N-grams for extracting features and evaluating that with Logistic Regression. The confusion matrix for this combination in Figure 5 shows some interesting insights. The best accuracy of the “AGAINST” label can be explained by the fact that the training dataset contains twice as much training data for this label as for the others, therefore, the trained model can predict this label wit higher success. This imbalance in the dataset has definitely has its impacts on the results, as the trained algorithm can be misleading. This should be addressed in the future, which we will describe in Section 6.

5.3 Comparison to known results

The paper written by Mohammad *et al.* [2016] discusses the SemEval-2016 Task 6 and its results. It is possible to use this paper and the mentioned implementations and their results for comparison. However, during the competition, the performance measure was somewhat different, as the macro-average of F-score was only calculated for positive classes ("Favor" and "Against", but not "Neither"). Therefore, to compare to these results, only for the sake of comparison, the implementation has been changed to evaluate the results the same way. Table 6 summarises the best and average results of this research.

Table 6: Comparison to known results

Implementation	**F-score**
This research with modified evaluation	0.6533
Baseline - SVM N-grams	0.6898
MITRE	0.6782
pkudblab	0.6683
Average in paper	0.6251
Baseline - Combined SVM N-grams	0.6206

The table shows that the results of this paper are better than the average results during the competition. Most implementations in the contest have been created in a way that all five targets are predicted with five different machine learning algorithms. This means that these models can perform with high performance on already seen target, and they perform very well for a specific task. The baseline implementation of the competition also contained a model with combined SVM (one model for all targets), which received an F-score of 0.6206, meaning that the implementation of this research was able to perform better. As this paper aims to find the best solution for general use cases, using one combined model is more reasonable in this case. Additionally, further limitations were taken, as explained in 5.5.

5.4 Recommendations

This paper researched the best techniques for stance detection, tested explicitly on the SemEval-2016 Stance dataset. Even though the achieved results are reasonable, it is not recommended to use a trained model for other use cases as the training dataset is very limited. However, if a bigger, and more general dataset can be collected to be used for training, such algorithm can be used for general use cases, such as fake-news detection or rumour classification [Küçük and Can, 2020].

5.5 Limitations

Even though the goal of this research was to find the best technique for stance detection, some limitations were present as the time-scope of the project would not allow the exploration of every possibility. Therefore, only the most commonly used methods have been used during the research and almost all with default parameters. However, other feature extraction methods and machine learning algorithms might exist, which were not used many times since 2016 (the first stance detection competition) but may perform better. Next to that, further tuning the hyper-parameters could help to increase the performance of the best result.

6 Conclusions and Future Work

During this project, we researched different combinations of feature extraction methods and feature-based machine learning algorithms in order to find the best performing pair for stance detection. To do so, we reviewed existing research, and found that in existing implementations the most common feature extraction methods are Word N-gram, TF-IDF, Bag-of-words, word embedding and POS tags. Additionally, the most common machine learning algorithms are SVM, Logistic Regression, Random Forest and Naïve Bayes. To find the best technique, we used the SemEval-2016 Stance dataset as a training and test dataset, and measured the performance with accuracy and F-score. As F-score is more commonly used for evaluating stance detection and is more sensitive to false results, we used it to determine the best combination.

After implementing all the aforementioned methods and combining them with the machine learning algorithms, we trained all twenty pairs with the training dataset. Then, we used the models to predict the results of the test dataset. Based on evaluating the results, the best pair is Word N-gram as a feature extraction method in combination with Logistic Regression. This combination resulted in an accuracy of 0.6557 and an F-score of 0.5991.

To improve these results, some future work can be done. First, it can be researched whether better data cleaning (such as lemmatisation, removing numbers or hashtags) would increase the performance of the algorithms. As mentioned in Section 5, an imbalanced dataset can also impact the results. This can be addressed by considering oversampling [Barua *et al.*, 2014] or using some method, such as SMOTE [Chawla *et al.*, 2002], to balance the classes. Then, it could be researched whether differently creating the feature vectors, such as using a different method for word embedding than normalisation, would further improve the accuracy. Lastly, since deep learning is often used in recent research, it would also be necessary to research whether deep learning methods would perform better for stance detection. Such research has been conducted by Jacob Roeters van Lennep [2021] as his Research Project.

7 Responsible Research

This research was conducted during the course of CSE3000 Research Project at the Delft University of Technology. The experiment was performed without any conflict of personal interest. The context of the research does not raise any ethical concerns as no sensitive data have been collected or stored, and the used dataset was retrieved from a publicly available website[3]. Since tweets can be sensitive, it is also important

[3]https://www.saifmohammad.com/WebPages/StanceDataset.htm

to mention that they are anonymised, and their tweet ID is not connected to them, therefore they cannot be traced back to their creators. The bias, if there is any, in the dataset might influence the working of the algorithm. The tweets can contain bias, which can change the way the algorithm will work, even though the algorithm itself is not biased in any way. Therefore, it is essential to mention that if such algorithms are used for real-life use cases, one must take care of a balanced and unbiased dataset. However, in this case, the trained algorithm will not be used in real-life use cases and is only evaluated on a test dataset created by the same organisation. Thus, one can conclude that in this context, bias does not play an important role.

To ensure reproducibility, all used packages, software and their versions have been mentioned in Section 3. Additionally, the complete codebase is available on the EEMCS GitLab repository[4]. This is a private repository, but access can be requested through my e-mail address. Since probability only took part in the research during a small and insignificant segment (as it only affects one machine learning algorithm, Random Forest), the same results, with the exception of Random Forest, are reproducible just by implementing everything based on the aforementioned descriptions or by running the code from GitLab.

Acknowledgement

I would like to express my appreciation and gratefulness to my supervisor and responsible professor, Pradeep Murukannaiah, for supporting this research with his knowledge, and insights into the topic. Further, I would like to thank my peer group — Wout Haakman, Jacob Roeters van Lennep, Abel van Steenweghen and Simon Mariën — for the ten weeks of work together, feedback and support.

[4]https://gitlab.ewi.tudelft.nl/cse3000/2020-2021/rp-group-65/rp-group-65-kvass.git

References

[Alammar, 2018] Jay Alammar. The illustrated word2vec [blog post], 2018.

[AlDayel and Magdy, 2021] Abeer AlDayel and Walid Magdy. Stance detection on social media: State of the art and trends. *Information Processing & Management*, 58(4):102597, 2021.

[Barua *et al.*, 2014] Sukarna Barua, Md. Monirul Islam, Xin Yao, and Kazuyuki Murase. Mwmote–majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):405–425, 2014.

[Bishop, 2006] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[Chawla *et al.*, 2002] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[Coleman, 2021] Keith Coleman. Introducing birdwatch, a community-based approach to misinformation, Jan 2021.

[FNC, 2017] Fake news challenge stage 1 (fnc-i): Stance detection, 2017.

[Ghosh *et al.*, 2019] Shalmoli Ghosh, Prajwal Singhania, Siddharth Singh, Koustav Rudra, and Saptarshi Ghosh. Stance detection in web and social media: a comparative study. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 75–87. Springer, 2019.

[Jones, 1972] Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

[Jurafsky and Martin, 2020] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Stanford University, 2020.

[Küçük and Can, 2020] Dilek Küçük and Fazli Can. Stance detection: A survey. *ACM Computing Surveys (CSUR)*, 53(1):1–37, 2020.

[Luhn, 1957] H. P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, 1(4):309–317, 1957.

[Mikolov *et al.*, 2013] Tomas Mikolov, G.s Corrado, Kai Chen, and Jeffrey Dean. Efficient estimation of word representations in vector space. pages 1–12, 01 2013.

[Mohammad *et al.*, 2016] Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41, 2016.

[Patra *et al.*, 2016] Braja Gopal Patra, Dipankar Das, and Sivaji Bandyopadhyay. Ju_nlp at semeval-2016 task 6: detecting stance in tweets using support vector machines. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 440–444, 2016.

[Pedregosa *et al.*, 2011] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[Rajaraman and Ullman, 2011] Anand Rajaraman and Jeffrey David Ullman. *Data Mining*. Cambridge University Press, 2011.

[Řehůřek and Sojka, 2010] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

[Rijsbergen, 1979] C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 2nd edition, 1979.

[Roeters van Lennep, 2021] J. F. Roeters van Lennep. Deep learning for stance detection: A review and comparison of the state-of-the-art approaches. unpublished, 2021.

[Sun *et al.*, 2016] Qingying Sun, Zhongqing Wang, Qiaoming Zhu, and Guodong Zhou. Exploring various linguistic features for stance detection. In *Natural Language Understanding and Intelligent Applications*, pages 840–847. Springer, 2016.

[Taulé *et al.*, 2017] Mariona Taulé, M Antonia Martí, Francisco M Rangel, Paolo Rosso, Cristina Bosco, Viviana Patti, et al. Overview of the task on stance and gender detection in tweets on catalan independence at ibereval 2017. In *2nd Workshop on Evaluation of Human Language Technologies for Iberian Languages, IberEval 2017*, volume 1881, pages 157–177. CEUR-WS, 2017.

[Wei *et al.*, 2016] Wan Wei, Xiao Zhang, Xuqin Liu, Wei Chen, and Tengjiao Wang. pkudblab at semeval-2016 task 6: A specific convolutional neural network system for effective stance detection. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pages 384–388, 2016.

[Xu *et al.*, 2016] Ruifeng Xu, Yu Zhou, Dongyin Wu, Lin Gui, Jiachen Du, and Yun Xue. Overview of nlpcc shared task 4: Stance detection in chinese microblogs. In *Natural Language Understanding and Intelligent Applications*, pages 907–916. Springer, 2016.

[Xu *et al.*, 2019] Feiyu Xu, Hans Uszkoreit, Yangzhou Du, Wei Fan, Dongyan Zhao, and Jun Zhu. Explainable ai: A brief survey on history, research areas, approaches and challenges. In Jie Tang, Min-Yen Kan, Dongyan Zhao, Sujian Li, and Hongying Zan, editors, *Natural Language Processing and Chinese Computing*, pages 563–574, Cham, 2019. Springer International Publishing.

A Literature Survey

This section lists the reviewed papers we used to create an overview of the most commonly used techniques. The following competitions are reviewed: SemEval-2016 [Mohammad *et al.*, 2016], NLPCC-ICCPOL-2016 [Xu *et al.*, 2016], and Shared Task of Stance and Gender Detection in Tweets on Catalan Independence at IberEval-2017 [Taulé *et al.*, 2017]. The complete list of papers can be found below.

[Addawood *et al.*, 2017] Aseel Addawood, Jodi Schneider, and Masooda Bashir. Stance classification of twitter debates: The encryption debate as a use case. In *Proceedings of the 8th International Conference on Social Media & Society*, SMSociety17, New York, NY, USA, 2017. Association for Computing Machinery.

[Ambrosini and Nicolo, 2017] Luca Ambrosini and Giancarlo Nicolo. Neural models for stancecat shared task at ibereval 2017. In *IberEval@ SEPLN*, pages 210–216, 2017.

[Augenstein *et al.*, 2016] Isabelle Augenstein, Andreas Vlachos, and Kalina Bontcheva. Usfd at semeval-2016 task 6: Any-target stance detection on twitter with autoencoders. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 389–393, 2016.

[Bar-Haim *et al.*, 2017] Roy Bar-Haim, Indrajit Bhattacharya, Francesco Dinuzzo, Amrita Saha, and Noam Slonim. Stance classification of context-dependent claims. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 251–261, 2017.

[Barbieri, 2017] Francesco Barbieri. Shared task on stance and gender detection in tweets on catalan independence-lastus system description. In *IberEval@ SEPLN*, pages 217–221, 2017.

[Bøhler *et al.*, 2016] Henrik Bøhler, Petter Asla, Erwin Marsi, and Rune Sætre. Idi@ ntnu at semeval-2016 task 6: Detecting stance in tweets using shallow features and glove vectors for word representation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 445–450, 2016.

[Dey *et al.*, 2017] Kuntal Dey, Ritvik Shrivastava, and Saroj Kaushik. Twitter stance detection — a subjectivity and sentiment polarity inspired two-phase approach. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 365–372, 2017.

[Dias and Becker, 2016] Marcelo Dias and Karin Becker. Inf-ufrgs-opinion-mining at semeval-2016 task 6: Automatic generation of a training corpus for unsupervised identification of stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 378–383, 2016.

[Elfardy and Diab, 2016] Heba Elfardy and Mona Diab. Cu-gwu perspective at semeval-2016 task 6: Ideological stance detection in informal text. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 434–439, 2016.

[Ferreira and Vlachos, 2016] William Ferreira and Andreas Vlachos. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 1163–1168, 2016.

[Gadek *et al.*, 2017] Guillaume Gadek, Josefin Betsholtz, Alexandre Pauchet, Stéphan Brunessaux, Nicolas Malandain, and Laurent Vercouter. Extracting contextonyms from twitter for stance detection. In *ICAART (2)*, pages 132–141, 2017.

[Garcıa and Flor, 2017] Diego Aineto Garcıa and Antonio Manuel Larriba Flor. Stance detection at ibereval 2017: A biased representation for a biased problem. *System*, 2:1, 2017.

[González *et al.*, 2017] José-Ángel González, Ferran Pla, and Lluís-Felip Hurtado. Elirf-upv at ibereval 2017: Stance and gender detection in tweets. In *IberEval@ SEPLN*, pages 193–198, 2017.

[Grčar *et al.*, 2017] Miha Grčar, Darko Cherepnalkoski, Igor Mozetič, and Petra Kralj Novak. Stance and influence of twitter users regarding the brexit referendum. *Computational social networks*, 4(1):1–25, 2017.

[HaCohen-Kerner *et al.*, 2017] Yaakov HaCohen-Kerner, Ziv Ido, and Ronen Ya'akobov. Stance classification of tweets using skip char ngrams. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 266–278. Springer, 2017.

[Hercig *et al.*, 2017] Tomás Hercig, Peter Krejzl, Barbora Hourová, Josef Steinberger, and Ladislav Lenc. Detecting stance in czech news commentaries. In *ITAT*, pages 176–180, 2017.

[Igarashi *et al.*, 2016] Yuki Igarashi, Hiroya Komatsu, Sosuke Kobayashi, Naoaki Okazaki, and Kentaro Inui. Tohoku at semeval-2016 task 6: Feature-based model versus convolutional neural network for stance detection. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 401–407, 2016.

[Krejzl and Steinberger, 2016] Peter Krejzl and Josef Steinberger. Uwb at semeval-2016 task 6: Stance detection. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 408–412, 2016.

[Küçük, 2017] Dilek Küçük. Joint named entity recognition and stance detection in tweets. *arXiv preprint arXiv:1707.09611*, 2017.

[Lai *et al.*, 2017] Mirko Lai, Alessandra Teresa Cignarella, Delia Irazu Hernandez Farias, et al. itacos at ibereval2017: Detecting stance in catalan and spanish tweets. In *IberEval*

2017, volume 1881, pages 185–192. CEUR-WS. org, 2017.

[Liu *et al.*, 2016a] Can Liu, Wen Li, Bradford Demarest, Yue Chen, Sara Couture, Daniel Dakota, Nikita Haduong, Noah Kaufman, Andrew Lamont, Manan Pancholi, et al. Iucl at semeval-2016 task 6: An ensemble model for stance detection in twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 394–400, 2016.

[Liu *et al.*, 2016b] Liran Liu, Shi Feng, Daling Wang, and Yifei Zhang. An empirical study on chinese microblog stance detection using supervised and semi-supervised machine learning methods. In *Natural Language Understanding and Intelligent Applications*, pages 753–765. Springer, 2016.

[Lozhnikov *et al.*, 2018] Nikita Lozhnikov, Leon Derczynski, and Manuel Mazzara. Stance prediction for russian: data and analysis. In *International Conference in Software Engineering for Defence Applications*, pages 176–186. Springer, 2018.

[Misra *et al.*, 2017] Amita Misra, Brian Ecker, Theodore Handleman, Nicolas Hahn, and Marilyn Walker. A semi-supervised approach to detecting stance in tweets. *arXiv preprint arXiv:1709.01895*, 2017.

[Mohammad *et al.*, 2016] Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41, 2016.

[Patra *et al.*, 2016] Braja Gopal Patra, Dipankar Das, and Sivaji Bandyopadhyay. Ju_nlp at semeval-2016 task 6: detecting stance in tweets using support vector machines. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 440–444, 2016.

[Simaki *et al.*, 2017] Vasiliki Simaki, Carita Paradis, and Andreas Kerren. Stance classification in texts from blogs on the 2016 british referendum. *Speech and Computer Lecture Notes in Computer Science*, page 700–709, Aug 2017.

[Sun *et al.*, 2016] Qingying Sun, Zhongqing Wang, Qiaoming Zhu, and Guodong Zhou. Exploring various linguistic features for stance detection. In *Natural Language Understanding and Intelligent Applications*, pages 840–847. Springer, 2016.

[Swami *et al.*, 2017] Sahil Swami, Ankush Khandelwal, Manish Shrivastava, and Syed Sarfaraz Akhtar. Ltrc iiith at ibereval 2017: Stance and gender detection in tweets on catalan independence. In *IberEval@ SEPLN*, pages 199–203, 2017.

[Taulé *et al.*, 2017] Mariona Taulé, M Antonia Martí, Francisco M Rangel, Paolo Rosso, Cristina Bosco, Viviana Patti, et al. Overview of the task on stance and gender detection in tweets on catalan independence at ibereval 2017. In *2nd Workshop on Evaluation of Human Language Technologies for Iberian Languages, IberEval 2017*, volume 1881, pages 157–177. CEUR-WS, 2017.

[Tsakalidis *et al.*, 2018] Adam Tsakalidis, Nikolaos Aletras, Alexandra I. Cristea, and Maria Liakata. Nowcasting the stance of social media users in a sudden vote: The case of the greek referendum. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 367–376, New York, NY, USA, 2018. Association for Computing Machinery.

[Tutek *et al.*, 2016] Martin Tutek, Ivan Sekulić, Paula Gombar, Ivan Paljak, Filip Čulinović, Filip Boltužić, Mladen Karan, Domagoj Alagić, and Jan Šnajder. Takelab at semeval-2016 task 6: Stance classification in tweets using a genetic algorithm based ensemble. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pages 464–468, 2016.

[Vijayaraghavan *et al.*, 2016] Prashanth Vijayaraghavan, Ivan Sysoev, Soroush Vosoughi, and Deb Roy. Deepstance at semeval-2016 task 6: Detecting stance in tweets using character and word-level cnns. *arXiv preprint arXiv:1606.05694*, 2016.

[Vinayakumar *et al.*, 2017] R Vinayakumar, S Sachin Kumar, B Premjith, Prabaharan Poornachandran, and Soman Kotti Padannayil. Deep stance and gender detection in tweets on catalan independence@ ibereval 2017. In *IberEval@ SEPLN*, pages 222–229, 2017.

[Wei *et al.*, 2016] Wan Wei, Xiao Zhang, Xuqin Liu, Wei Chen, and Tengjiao Wang. pkudblab at semeval-2016 task 6: A specific convolutional neural network system for effective stance detection. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pages 384–388, 2016.

[Wojatzki and Zesch, 2016] Michael Wojatzki and Torsten Zesch. ltl. uni-due at semeval-2016 task 6: Stance detection in social media using stacked classifiers. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 428–433, 2016.

[Wojatzki and Zesch, 2017] Michael Wojatzki and Torsten Zesch. Neural, non-neural and hybrid stance detection in tweets on catalan independence. In *IberEval@ SEPLN*, pages 178–184, 2017.

[Xu *et al.*, 2016] Jiaming Xu, Suncong Zheng, Jing Shi, Yiqun Yao, and Bo Xu. Ensemble of feature sets and classification methods for stance detection. In *Natural Language Understanding and Intelligent Applications*, pages 679–688. Springer, 2016.

[Yu *et al.*, 2016] Nan Yu, Da Pan, Meishan Zhang, and Guohong Fu. Stance detection in chinese microblogs with neural networks. In *Natural Language Understanding and Intelligent Applications*, pages 893–900. Springer, 2016.

[Zarrella and Marsh, 2016] Guido Zarrella and Amy Marsh. Mitre at semeval-2016 task 6: Transfer learning for stance detection. *arXiv preprint arXiv:1606.03784*, 2016.

[Zhang and Lan, 2016] Zhihua Zhang and Man Lan. Ecnu at semeval 2016 task 6: Relevant or not? supportive or not? a two-step learning system for automatic detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 451–457, 2016.

[Zhang *et al.*, 2017] Shaodian Zhang, Lin Qiu, Frank Chen, Weinan Zhang, Yong Yu, and Noémie Elhadad. We make choices we think are going to save us: Debate and stance identification for online breast cancer cam discussions. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, page 1073–1081, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

B Experimental setup

Table 7: Libraries used for the experiment

Library	Version
gensim	4.0.1
matplotlib	3.4.2
nltk	3.6.2
numpy	1.20.3
pandas	1.2.4
scikit-learn	0.24.2
scipy	1.6.3

C Performance measure

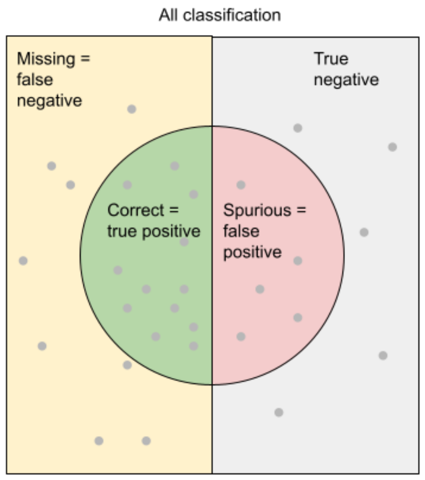


Figure 6: F-score datapoints graphical representation

D Implementation details

D.1 Feature extraction algotihms

This section describes how the feature extraction methods are used and what the results of the extractions are. In order to consistently have the same structure, we created an abstract class *FeatureExtractionMethod* with two methods (*_set_up()* and *make_feature_vectors()*), which are used by all the feature extraction methods. The corpus always contains all elements of the training dataset's tweets and targets.

TF-IDF

To create the feature vector with TF-IDF, we use scikit-learn's [Pedregosa *et al.*, 2011] TfidfVectorizer[5]. This method (by default) creates unigrams from the corpus and calculates the TF-IDF values for every word in the text-target pairs based on the formula introduced in Section 2.2. To filter contextually-low-meaning words, the English stop words are not used with the setting *stop_words='english'*. Further, scikit-learn automatically filters punctuation from the corpus.

To gain extra insight into the context, we calculate the cosine similarity of the vectors. Then, we append the two TF-IDF vectors and cosine similarity value next to each other, resulting in an highly sparse feature vector of size of 17523.

Word Embedding

To use word embedding to create a feature vector, the word2vec vector of every word in the target text is collected with the help of the gensim library's [Řehůřek and Sojka, 2010] KeyedVectors class[6], which helps to map between words and their vectors. The binary file used for word2vec can be found on the Google Code Archive's Google Drive[7]. If the dictionary does not contain a word (such as hashtags), the word is simply skipped. After collecting the vectors, they are added together, and lastly, normalised. Therefore, the result feature vector is the normalised word2vec vector of the text and target appended after each other, resulting in a vector size of 600.

Bag-of-words

To create the feature vector with bag-of-words, we create unigrams from the corpus, and every target text is represented using that vector, showing how many times each unigram appears in the text. To create the bag-of-word vector, we use scikit-learn's [Pedregosa *et al.*, 2011] CountVectorizer[8]. As explained previously, the English stop words are filtered here as well, and *min_df* has been set to 3 to filter words that appear less than 3 times in the whole corpus. Next to that, such as with TF-IDF, punctuation is automatically filtered. The count vector of the text and the target are appended after each other, resulting in a feature vector of size of 1735.

[5]https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[6]https://radimrehurek.com/gensim/models/keyedvectors.html

[7]https://code.google.com/archive/p/word2vec/

[8]https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

Word N-gram

Similar to bag-of-words, we create Word N-grams with CountVectorizer. The only difference to bag-of-words is that instead of using just unigrams in the "bag", unigrams, bigrams and trigrams are created and used by setting *ngram_range* to *(1,3)*. The same filtering as bag-of-words is applied here as well. The feature vector of using word N-grams results in a vector of size of 2267.

POS tags

To use POS tags for creating feature vectors, we retrieve the POS tag of every word within the corpus. Then, we attach the tag to the word with a "/" separator and used the same way as bag-of-words was used. When using POS tags with unigram bag-of-words, the created feature vector resulted in a size of 9060.

D.2 Machine learning algorithms

After creating the feature vectors, the machine learning algorithms are trained on the training dataset. We use four different models to classify the results with the help of the scikit-learn [Pedregosa *et al.*, 2011] library. For SVM, the SVC module[9] is used with 5-fold cross-validation with the help of GridSearchCV[10] to find the best C-value and kernel type. For Linear Regression, the module LogisticRegressionCV[11] with *random_state=0* and *cv=5* for 5-fold cross-validation is used. Random Forest is run by the RandomForestClassifier module[12]. Lastly, for Naïve Bayes, the MultinomialNB module[13] is used.

Lastly, we combine all five feature extraction methods with the four models, resulting in twenty different combinations. After training the models, evaluation is done on the test dataset. To evaluate the results, and measure accuracy and F-score, we use scikit-learn's [Pedregosa *et al.*, 2011] metrics module[14].

[9]https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[10]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[11]https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegressionCV.html

[12]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[13]https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

[14]https://scikit-learn.org/stable/modules/classes.html