Building a rudeness classifier: a word-based approach

Freek Boutkan
6303846

Bachelor thesis
Credits: 18 EC

Bachelor Opleiding Kunstmatige Intelligentie

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
Drs. T.M. Kenter

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

June 26th, 2015

Abstract

The main objective of thjs research is to build a rudeness classifier with a high F_1-score; a classifier that can correctly classify a sentence in the Dutch language. A training set of 36,000 non-rude sentences and 313 rude sentences has been provided. A word-based approach will be used, meaning that all text features are based on single words and do not rely on document properties or sentence structures. This approach is easy to comprehend, to debug and to implement parallelly.

The text features used in this research are semantic similarity measurements, key-word presence indicators, part-of-speech tags, key-word based sentiment analysis and word-length distribution. The semantic vectors are built using Word2Vec and their quality will be tested with a semantic task. A Support Vector Machine (SVM) will be used to learn from the word-based features. An F_1-score of 0.84 is reported on a test set that is manually annotated. It can be concluded that word-based features, some of which are based on semantic vectors, are successful in combination with SVMs.

Contents

1 Introduction

The main goal of this research project is to build a rudeness classifier. The rudeness classifier will be used in future research which will examine the possible development in the quantity of rude utterances in newspapers over time. A rudeness classifier is needed for automatic classification of considerable large amounts of text since, it cannot be done manually. In order for the classifier to be of any use, have a good sensitivity and precision. Detecting rudeness in written natural language is a hard task, not only for a computer algorithm but also for humans. In general, any subtle emotion in natural language is difficult to detect as opposed to the identification of vague emotions such as general happiness or sadness which sentiment analysis can detect reasonably well (Saif, He, & Alani, 2012). Moreover, the quantity of training data for this classification task is quite low and the training of the classifier is therefore challenging. Identification of the aforementioned subtle emotions requires an advanced algorithm that understands the meaning of words and sentences.

Semantic Models are widely used in Natural Language Processing tasks such as information retrieval systems, translation systems and question-answer applications. The determination of rudeness in written text is a difficult task which can be approached better by the use of semantic models since rudeness cannot be determined by merely by keywords: all possible lists of keywords would be either incomplete or too broad to detect rudeness accurately. Semantic Models attempt to capture and model the meaning of words rather than its number of occurrences. A *complete* list of rude keywords is no longer required if the meaning of texts can be modelled accurately, because the list of keywords can be extended by searching for similar words in the semantic vector space (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013; Zhang, Xu, Su, & Xu, 2015). Semantic Models are therefore required in more complex text classification tasks to overcome the keyword list problem.

Semantic Vectors are becoming more popular since they are uncomplicated in their implementation and computational efficiency. They have however various properties that can be problematic in this classification task. Firstly, semantic vectors do not take ambiguity into account. A word that has different meanings like ‘bank’ will have only one semantic vector. However, in natural languages the word has two different meanings. Subtleties such as these cannot be taken into account since every word can only have one associated semantic vector. Secondly, semantic vectors have a scope that is limited to a single word, i.e. a vector can represent only a single word. This is problematic for word combinations such as ‘New York’, preprocessing the data can solve this problem though. Moreover, the creation of semantic models for whole sentences or whole texts from separate word vectors is a non-trivial task. A sentence is constructed of separate words which each have their own representation, though building a model that captures the actual relation between those words is a challenge. Lastly, creating vectors for words that do not appear often in training corpora tend to be of poorer quality.

This research project will investigate the results and suitability of a classifier that uses semantic models and other word-based properties, for the construction of its features. If the classification is successful, the detection of other subtle linguistic utterances of emotions and sentiment such as cynicism, sarcasm or humour in written natural language might benefit form this approach. The resulting algorithm can be trained using a relatively small quantity annotated data containing the subtleties of interest in combination with a relatively large amount of general written language. Other advantages of this method are that the algorithm is fast to train, can use existing text corpora and might be used to find other subtleties apart from rudeness as well.

The rest of the paper is structured as follows: in section 2 the theoretical foundation of this research will be outlined and related works will be reviewed. Section 3 will explain the approach used in this paper in more detail. The following section will explain the technical details of the conducted experiments, the results of which will be shown in section 5 and discussed in section 6. The final section, 7, states the conclusions of this research paper and proposes new research subjects in this field.

2 Theoretical Foundation

2.1 Related works

Text and document classification has been researched extensively. Drucker, Wu, and Vapnik (1999) examine the results of support vector machines for the detection of spam messages, a typical document classification task. Text and document classification methods have gained popularity since then mainly because the amount of documents that is publicly available has enormously grown with the rise of the internet (Khoo, Marom, & Albrecht, 2006). Moreover, document classifiers are used in information retrieval systems and search engines that became much more important with the rise of the internet. The terms and words in such documents describe the document and are therefore used in text classification tasks. Because documents tend to contain a considerable amount of text, the vectors representing those document can become high dimensional. In order to make an algorithm that can quickly classify a document, feature selection is an important part of the creation of the classifier (Khoo et al., 2006).

Sentence classification differs from text classification in the scope. Text classifiers try to add labels to documents (or to emails, articles, etc.) and sentence classifiers try to label sentences. Some sentiment analysis tasks can be seen as a classification problem. Sentence classification can also be used in automated summary generation tasks (Khoo et al., 2006).

Nowadays, most sentence classification tasks use either neural network techniques (Kim, 2014) or vector space models (Erk & Padó, 2008). Neural network techniques and other unsupervised learning methods require a large amount of data and are therefore not suitable for this problem. *Vector space* or *semantic space* models do not require special data to train on, as they can be trained on any sufficiently large enough corpus of natural language in which the presence of rude sentences is not relevant. The main assumption underlying all the semantic/vector space model algorithms is that the Harris's Distributional Hypothesis is true. The Distributional Hypothesis states that words that occur in the same contexts tend to have similar meanings, so the meaning of words is characterised by the words around it (Harris, 1954). The meaning of the words is therefore determined by the context in which it normally appears. This property of natural language is the theoretical foundation for vector space models.

Vector space models use vectors as representations of the object of interest. In case of semantic models the object of interest can be the semantics of a sentence or a document. Each dimension of the vector corresponds to a single term needed to represent the object, for example the count of certain keywords in the sentences. Distributional Semantic Models such as Latent Semantic Analysis (LSA) use vectors that count occurrences of words with respect to the context of that word (e.g. the neighbouring words in a certain window size) in large corpora of text. Then those vectors will be transformed using geometric techniques to meaningful semantic word-vectors in such a way that similar words will have similar vectors i.e. the distances between those vectors is small (Baroni, Dinu, & Kruszewski, 2014).

2.2 Word2Vec

Mikolov, Chen, Corrado, and Dean (2013) have proposed a new method for synthesising semantic vectors, called *Word2Vec*, that uses a supervised learning task for an unlabelled data set as opposed to the older DSM techniques that use simple count matrices and arithmetic transformations. Mikolov, Chen, et al. thought of techniques that created supervised learning tasks form the unlabelled text. They transformed the unsupervised learning task of learning the correct semantic vectors to a set of supervised learning task. They showed that accurate semantic word vectors could be learned from unlabelled training that, i.e. a large corpus of text. Their algorithm is efficient; it focuses on processing large quantities of text rather than on capturing complicated structures of language in advanced models. Their big-data like approach resulted in good performance on all syntactic and semantic evaluation test.

Word2Vec has two methods to transform the unlabelled training data into supervised learning tasks, namely CBOW and Skip-gram, as illustrated in figure 1. CBOW (continuous bag of words) tries to predict a word ($w(t)$) given the next/previous N words (context of the word) wheres Skip-gram tries to predict the context of a word. Both tasks work well, whereas there are some differences in performance (Mikolov, Chen, et al., 2013). Word2Vec creates a large number of those training tasks (CBOW or Skip-gram). Thus, the unlabelled text input is used to create many classification tasks (predicting the words) resulting in a multi-class classification problem. A two-layered neural network is used to find the solution to the classification tasks of all the words. The weights in the first layer in the neural network correspond to the semantic words vectors (Rong, 2014).

Word2Vec is a state-of-the-art performance in numerous syntactic and semantic tasks (Baroni et al., 2014; Levy, Goldberg, & Dagan, 2015; Mikolov, Chen, et al., 2013; Mikolov, Sutskever, et al., 2013). The performance of Word2Vec has been extensively examined and performs better in all tasks compared to traditional count models (Baroni et al., 2014). Although the performance of Word2Vec is better, the inner working of the algorithm was not well understood: it was not clear what the optimisation objective is and what it means (Levy & Goldberg, 2014). Levy and Goldberg have discovered that one variant of Word2Vec, being skip gram with negative sampling, is implicitly equal to factorising a word-context matrix,

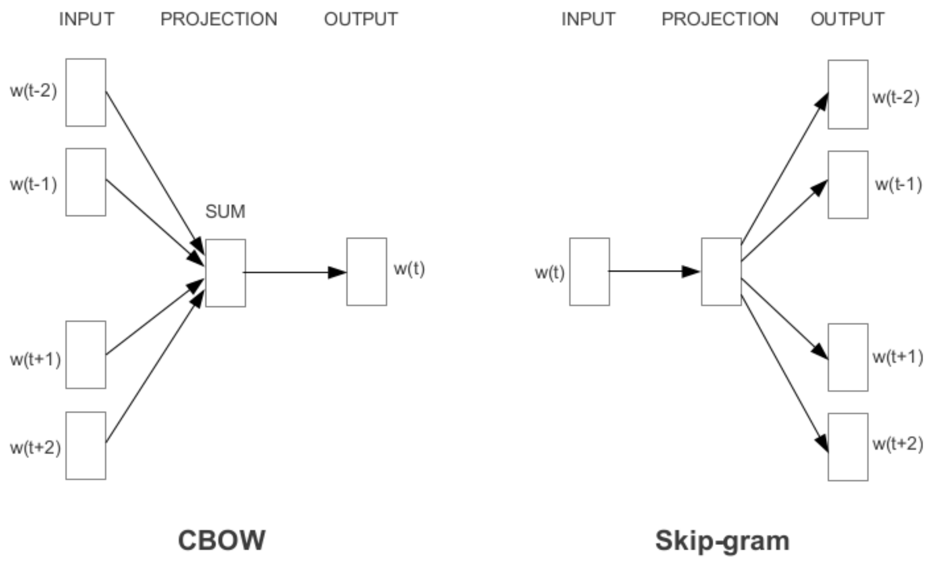


Figure 1: Tasks used to learn word embeddings: CBOW and Skip-gram
Image taken from Mikolov, Chen, et al. (2013, p.5)

which cells are the Point-wise Mutual Information (PMI) of the respective word and context pairs, shifted by a global constant. Therefore it can be concluded that there is no difference between the count models and the predict models in aforementioned cases. Predict models like Word2Vec, GloVe and NCE seem to optimise the same property as count models do, though in a very efficient manner and therefore resulting in superior performance.

2.3 Support Vector Machines

Support Vector Machines are a set of supervised learning algorithms that can be used for classification or regression problems. SVM became popular after Cortes and Vapnik (1995) developed the existing algorithm in such a way that SVMs were able to perform non-linear classification problems, which was not possible before. SVM finds the optimal decision boundary by placing a line through the vector space that maximises the distance between both classes. The usage of a kernel function is a well-known method nowadays. It maps the input, in the case of SVM being the data of the training examples, to a higher (possibly indefinite) dimensional vector space. In this new vector space the algorithm searches for a decision boundary, so that all training examples are in the right class in this new vector space. For an illustrated example see figure 2. SVM does not need to transform

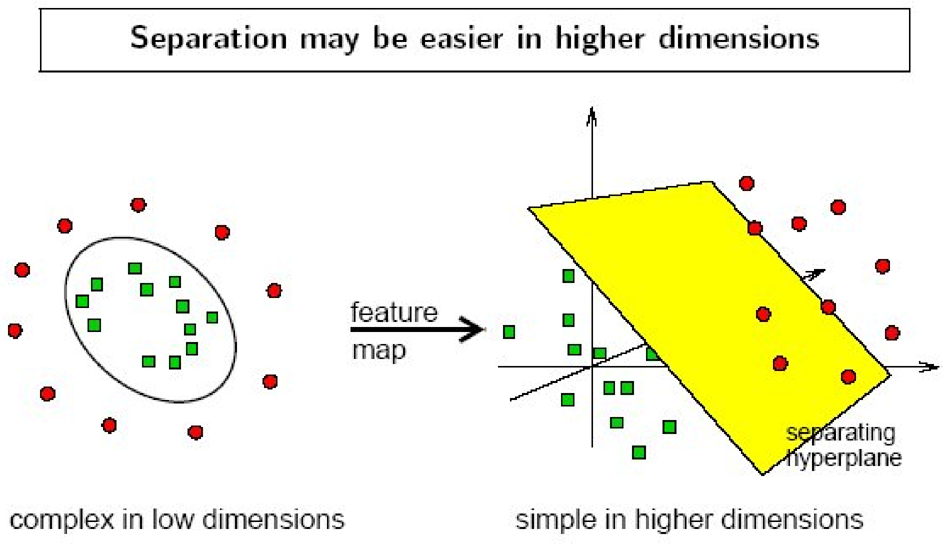

Separation may be easier in higher dimensions
feature
map
separating
hyperplane
complex in low dimensions
simple in higher dimensions

Figure 2: Mapping data to higher dimensional space, the yellow plane is the found decision boundary
Image taken from https://www.dtreg.com/solution/view/20

each vector to the new vector space explicitly, because the only value SVM needs to have as an input is the pair-wise dot product between two vectors in this new vector space. Kernel functions can implicitly calculate the dot product between two vectors in this higher dimensional space without explicitly transforming the vectors into that space.

2.3.1 SVM in combination with Word2Vec

Zhang et al. (2015) demonstrate that the use of semantic vectors in combination with SVM yields positive results in certain sentiment classification tasks, a task that seems related to the rudeness classification. An F1 score of 0.90 has been reported for a binary sentiment classification task (which is to classify each sentence as either positive or negative). Word2vec is used to get the word embeddings for the Chinese texts, following a clustering method is applied to find similar features in the vectors, i.e. different words with similar meanings such as synonyms or singular/plural pairs.

Each text has been given a score based on the Word2Vec-vectors (and aforementioned clusters to account for synonyms) and is enriched with other features (either part-of-speech or lexicon-based features). A support vector classifier is trained using this feature set and is evaluated. The results reported are state-of-the-art: a 90 percent accuracy for both feature sets, one containing part-of-speech tags and the other set containing lexicon-based vectors. The results obtained by Zhang et al. indicate that Word2Vec and SVM are useful tools for semantic binary classification tasks such as a rudeness classifier. Furthermore, they indicate that use of part-of-speech tags can be a suitable feature for this classification problem.

3 The word-based approach

3.1 The word-based approach explained

The rudeness classifier must be able to correctly classify a given sentence as either rude or non-rude; no additional rudeness scores will be returned other than a binary return value. This task can thus be seen as a binary classification task. The training examples consist of an annotated data set consisting of 1,000 articles from Dutch newspapers. The total number of sentences is 36,000 of which 313 phrases have been given a label indicating rude language utterances. If a sentences contains a rude part the sentence as a whole is marked as rude. The training set contains therefore 36,000 negative examples and 313 positive examples of rudeness. The classifier must learn on these 36,000 annotated sentences.

A sentence is a set of words and for each of the words in the set properties will be computed, resulting in properties for the sentence as a whole. Those properties will be used as features in the classifier. All the features that the classifier uses are based on properties of the single words that combined characterise the sentence, hence the name *word-based* approach[1].

Other approaches try to do the opposite by using features based on sentences or documents as a whole, for example Mikolov, Sutskever, et al. (2013). Numerous other studies compare and cluster documents or use document properties as learning features, both of which will not be used in this approach that is purely word based.

3.2 Word-based features

Feature engineering is an essential part of building an accurate classifier (Khoo et al., 2006) because that is all the information a classifier can use. In this study a variety of features will be used. The features can be divided into the following categories: semantic similarity, indicator, newspaper, part-of-speech tag, word length and sentiment. Based on the work of Zhang et al. the semantic similarity is expected to be the most important feature, because it encodes much more information than for example a part-of-speech tag or information about the word length in a sentence.

3.2.1 Semantic Similarity features

The features in the semantic similarity category are based on the semantic vectors, created by a Word2Vec implementation. For each of the 313 rude sentences a

[1]There is one exception, see section 4 for details.

semantic vector is calculated by adding the semantic vectors of the single words combined with a weighting term together. The weighting term favours words that contain more information, i.e. rare words. Words like 'and', 'or', 'the', 'of' appear in almost all sentences and are therefore not relevant.

For every sentence in the training and test set a semantic vector is computed just as described above. That vector is being compared to the 313 rude sentences vectors. The highest similarity score of those 313 vectors is used as a feature as well as information about the distribution of the similarity scores i.e. how many of all known rude sentences are closely, vaguely, not at all related to this sentence.

3.2.2 Other features

The other features are easy to compute: part-of-speech tags are counted and reported as a feature, presence of keyword or names in a sentence is indicated by a set of binary features. The newspaper in which the sentence has been found is the only non-keyword-based feature. The reason that it has been implemented is that future research might want to use that feature. Word-length-distribution is implemented in the same way as semantic similarly score is used: it indicates how many words are longer than x chars for some values of x. The final feature is a sentiment score consisting of two numbers: the first one indicates the sentiment (very bad to very high) and the second number is a objectivity measurement. Both calculations are based on a keyword-based so it can be called a word based feature.

3.3 Motivation for word-based approach

For this particular classification task no specific requirements have been set, other than high recall and precision scores. However, for the practical application of this classifier it is important not to be dependent on too many resources such as memory usage and running time. Moreover, a simple algorithm is favourable as opposed to a more complex version, because of more human ease in its comprehension and thus in its development and the debugging process.

A third advantage of word-based approach is the fact that it is computationally cheap: no complex calculations are needed. Once the shared resources have been calculated the scoring process for the newly to be classified sentences is simply a matter of searching values in dictionaries and simple arithmetics. The sharable resources that might have a longer duration to be calculated (such as the semantic vectors) can be produced before running the algorithm which effectively has a positive impact on the total classification time. Another benefit of the aforementioned resources is that they can be shared across multiple applications, so they might already be available. Lastly, this algorithm is easy to apply parallelly. The properties for each single word can be computed independently and therefore this can be executed for large quantities of data divided over several CPUs.

4 Experimental Setup

4.1 Feature Computation

The semantic vectors are created in Python using the gensim library (Řehůřek & Sojka, 2010). A high-quality Dutch corpus, called COW (corpora from the web) (Schäfer & DFG, n.d.) was used as training set for the semantic vectors. Various parameters for the training of the semantic vectors have been tried but the used model is in all cases skip-gram. The accuracy of each set has been tested using a semantic task that tries to find the correct capital for each country, e.g. Athens is to Greece as Baghdad is to ...(Iraq). Semantic tasks are more complex than syntactic tasks. The accuracy of the task is however not a perfect indicator for the quality of the vectors, the quality of the semantic vector can only be tested by using it in the application. The Python package Pattern (De Smedt & Daelemans, 2012) has been used for the generation of the part-of-speech tags and sentiment analysis (see table 1 for the number of features in each category.) All data is stored in an SVM-light file format.

The semantic similarity measure is defined as the cosine similarity between the semantic vector for each sentence. The semantic vector for a sentence is simple the sum of the inverse document frequenty (IDF) weighted vectors of all the words of that sentence.

Feature Category	Number
Semantic similarity	7
Word length	4
Part-of-Speech	12
Entity Indicators	30
Newspaper	3
Sentiment	2
Total	58

Table 1: This table shows the number of features for each category

4.2 The training set

The training of the classifier will be done using the manually annotated set of 1,000 newspaper articles. As mentioned before, all sentences that contain one or more words that have been annotated as rude will be marked as rude, whereas all other sentences will be assumed to be non-rude, although those sentences are not labelled as non-rude. The following preprocessing steps are applied to the data: all sentences in the articles are tokenized using the Python package NLTK (Bird, 2006) that contains a special tokenizer that has been optimised for the Dutch language. Following, all punctuation was removed and the text was converted to all lowercase characters.

4.3 Applying SVM

The used kernel for the SVM is rbf. The rbf kernel has one hyper-parameter γ that has to be set. SVM itself also has a hyper-parameter that can be set, being C. Each classifier is therefore characterised by these two hyper-parameters: C and γ. All data is normalised using the mean and standard deviation of each feature, so that the mean is 0 and the variance is 1 after normalisation. Both the training set and the test set are normalised using the mean and standard deviation of the training set. A grid search will be conducted to find the optimal hyper-parameter settings. A part of the training set (313 rude sentences + 1000, 2000, 3000 or all of the non-rude sentences) is used to evaluate the performance. Evaluation is done by leave-one-out cross-validation and the classifiers with the highest F1-scores are selected for test set evaluation. The experimental setup follows the recommendations from Hsu, Chang, Lin, et al. (2003).

4.4 Evaluation

The classifiers are evaluated by comparing its performance on a test set that is annotated by humans. The generation of the test set was done using a heuristic function because rude sentences appear infrequently in texts. If 1000 sentences were randomly selected it could be the case that not a single rude or far too few sentences emerge in the test set. The test set contains 999 lines and has been annotated manually resulting in 120 rude sentences. It can therefore be conclucde that the used heuristic has an accuracy of 12%.[2] The used evaluation measure is the F_1-score.

[2]Actually, two heuristic functions were used, each selecting 500 lines, their accuracy was 8% resp. 16%

5 Results

5.1 Semantic Vectors

The results of the accuracy test are shown in table 2. When hierarchical soft-max is used the accuracy is very low as shown in the table. The size of the training set is reported as the number of sentences. Min_c indicates the minimal number of occurrences for each word in the training set in order to be considered as part of the vocabulary. An accuracy of 11.2% percent is reported, indicating that the training size is more important than the number of dimensions of the semantic vectors.

5.2 Feature Sets

Two different feature sets have been tested; table 3 shows the distribution of features in each set. Feature set 1 contains almost all features whereas feature set 2 only contains a subset of those features extended with an extra similarity measure and sentiment analysis.

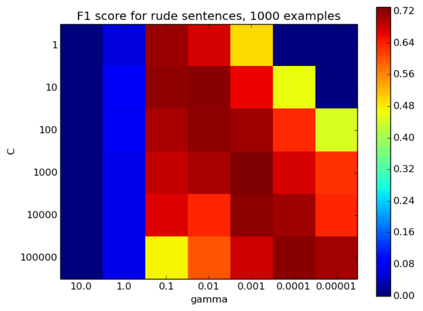
The grid search for the best parameters has been successfully conducted. Figure 3 shows the F1 scores obtained via leave-one-out cross-validation. Following the data in the table, generaly speaking the performance is better for higher C values and lower γ values. This pattern has also been observed in feature set 2 (see figure 6 in appendix A on page 25 for the heatmap plot).

5.3 Rudeness classification performance

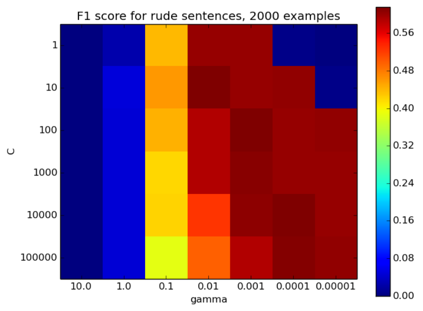
After the optimal hyper-parameters were found the test set could be evaluated. Feature set 1 did not result in any usable results, the results are therefore omitted, see section 6 for more information. The results for feature set two are shown in figure 4 and table 4.

Size	Dimensions	Window	min_c	sample	neg	hs	acc
1000000	200	5	1	0.000000	3	0	3.1
1000000	300	5	10	0.000010	x	1	0
10000000	200	5	5	0.000000	5	0	11.2
10000000	300	5	5	0.000010	x	1	0.5

Table 2: This table shows the accuracy on the Word2Vec training task



(a) 1,000 non + 313 rude examples



(b) 2,000 non + 313 rude examples

Figure 3: Grid Search results for feature set 1, F_1-score is shown

Feature Category	Set 1	Set 2
Semantic similarity	6	7
Word length	4	0
Part-of-Speech	12	0
Entity Indicators	29	1
Newspaper	3	0
Sentiment	0	2
Total	54	10

Table 3: the number of features in each feature set

The best classifier in terms of the averaged F_1-score is the classifier characterised by $C = 100,000$ and $\gamma = 0.000001$ with an average F_1-score of 0.84 on the test set containing 120 rude examples and 879 non-rude examples. The accuracy for this classifier is 86% which is near the best score of 88%.

The best classifier in terms of the F_1-score for just the class of rude sentences is characterised by $C = 100$ and $\gamma = 0.01$ resulting in an F_1-score of 0.36 and an accuracy of 82%. In all cases, the F_1-scores for the rude-sentences class is lower than the ones for the non-rude sentences. In some cases the F_1-score for rude sentences is even 0, which means not a single example has been classified(correctly). The averaged F_1 score is largely determined by the F_1 score for non-rude examples because the number of non-rude examples in the training set is considerably bigger.

		F_1-score			Accuracy
C	γ	R	NR	avg	total
10	0.01	0.90	0.34	**0.84**	0.83
	0.001	0.90	0.31	0.83	0.82
	0.0001	0.94	0.00	0.82	0.88
	0.00001	0.94	0.00	0.82	0.88
	0.000001	0.94	0.00	0.82	0.88
100	0.01	0.89	**0.36**	0.83	0.82
	0.001	0.89	0.32	0.82	0.81
	0.0001	0.91	0.30	0.83	0.84
	0.00001	0.94	0.00	0.82	0.88
	0.000001	0.94	0.00	0.82	0.88
1000	0.01	0.84	0.29	0.77	0.74
	0.001	0.85	0.34	0.79	0.75
	0.0001	0.90	0.31	0.83	0.83
	0.00001	0.94	0.00	0.82	0.88
	0.000001	0.94	0.00	0.82	0.88
10000	0.01	0.82	0.28	0.76	0.71
	0.001	0.84	0.33	0.78	0.74
	0.0001	0.90	0.32	0.83	0.83
	0.00001	0.91	0.30	0.83	0.83
	0.000001	0.94	0.00	0.82	0.88
100000	0.01	0.82	0.28	0.76	0.72
	0.001	0.85	0.33	0.79	0.76
	0.0001	0.83	0.32	0.77	0.73
	0.00001	0.90	0.30	0.83	0.83
	0.000001	0.92	0.27	**0.84**	0.86

Table 4: F_1-scores for rude (R) non-rude (NR) classes, averaged and overall accuracy for various classifiers characterised by C- and γ-values.

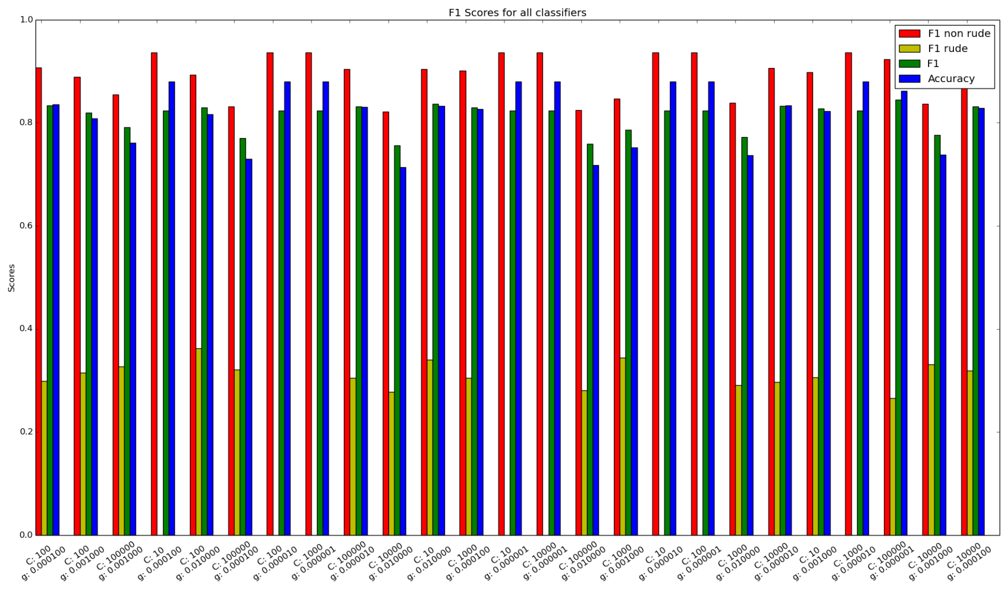


Figure 4: F_1-scores and accuracy for various classifiers. The full resolution graph can be found in Appendix A on page 24.

6 Discussion

6.1 Semantic Vectors

The semantic vectors are successfully trained for the Dutch language. The accuracy scores are low compared to the original study by Mikolov, Chen, et al. for similar semantic tasks, though this can be explained by the size of the training corpus. Mikolov, Chen, et al. used a corpus containing 6 billion tokens as opposed to 10 million sentences in the corpus used in this study.

The evaluation of the semantic vector can not be done using only artificial tasks, the quality can only be tested in the application in which it will be used. The semantic vectors are used in the similarity measure that is used in the algorithm that created the test set. The accuracy of the test set is 16% and the test set is generated only using semantic similarity. If the semantic vector were of poor quality, the number of rude sentences in the test set would probably be lower. So it can be concluded that the semantic vectors are sufficiently trained for the goals of this study.

6.2 Feature set 1

As noted in the results, feature set 1 did not yield any useful classifiers. All classifiers predicted either random classes or just one class (a naive predictor). A classifier ($C = 10, \gamma = 0.1$) predicted a non-rude class for all entries in the test set. The accuracy of this classifier was 879/999 = 88%, which is fairly good. But the recall and precision for the rude-class are consequently 0, so it can be concluded that the classifier did not learn anything and is of no use.

It is not known why this feature set performed so poorly. Zhang et al. (2015) reported high quality results using part-of-speech tags, which are present in this feature set, for a similar binary classification task for sentences. They also used semantic vectors produced by the Word2Vec algorithm and an SVM classifier. It may be the case that there were too many features (58) for 313 positive examples in the training set.

6.3 Feature set 2

The smaller feature set, only containing semantic similarity measures, sentiment scores and a single indicator indicating whether one or more of the keywords from a list is present in the sentence or not, performed significantly better than the first feature set. The smaller number of features (10) in this features set can be a reason why it performed better. It is also possible that the features in the first set are completely independent of the rude/non-rude label and therefore do not contribute any useful information to the classifier. In fact they might add noise to data of the classifier, that complicated the classification process.

The best classifier is characterised by $C = 100,000$ and $\gamma = 0.000001$ but the same score is achieved by classifier characterised by $C = 10$ and $\gamma = 0.01$ which is other then expected. The latter also has an F_1-score for rude sentences of .34 which approximates the best score. This result is as expected since the training set, evaluated using leave-one-out cross-validation, performed well with the indicated values in any case (see figure 3).

6.4 The training set

Initially, the number of rude examples in the training set (313) was much lower than expected and indicated a possible problem, because it appeared that more data would be needed to build a detector for such a subtle linguistic property as rudeness. But the second feature set resulted in reasonably acceptable results.

6.5 Classifier quality

The test set contains 120 rude sentences, 12% of the total file. In the training set only 313 of 36000 (0.9%) of the sentences is rude. The rudeness distribution in natural languages is clearly very skewed. Building classifiers for such skewed data sets is difficult, especially if a large training set is not available. The reported averaged F_1-scores depend much more on the classification of non-rude sentences than on the classification of the rude sentences, simply because there are much more non-rude sentences.

The question remains which classifier is the best in rudeness detection: the one with the highest overall F_1-score or the one with the highest F_1-score for rude sentences. The answer to that depends on the exact needs for the classifier. If its goal is to classify a sentence as rude or non-rude the best classifier is the one with the highest averaged F_1-score. If one would be interested in detecting rude sentences without any concern for the non-rude sentences the classifier with the highest F_1-score for rude sentences would be the best classifier.

7 Conclusion and future research

The main objective of the research project was to build a rudeness classifier with a high F_1-score; a classifier that can correctly classify a new Dutch sentence as either rude or non-rude. The main problem was the small amount of positive examples in the training data.

A word-based approach has been chosen, meaning that all text features are based on single words, not on document properties or sentence structures. This approach is easy to comprehend to debug and to implement parallelly. The text features used in this research are semantic similarity measurements, key-word presence indicators, part-of-speech tags, key-word based sentiment analysis and word-length-distribution. Part-of-speech tags and word-length features did not contribute to successful classification of sentences, whereas semantic similarity measurements and sentiment analysis did.

As discussed, Word2Vec is used to acquire semantic vectors. Different parameters have been attempted each providing different results. The creation of the test set heavily relies on the semantic vectors' quality. Since the quality of the test set was high, 16% rude sentences as opposed to 0.9% in the training set, it can be concluded that the quality is indeed high. The word-based approach in this study did yield positive results for certain classifiers. Overall, averaged F_1-scores of 0.84 have been reported on the final test set.

For future research the key-word approach should be extended. Results from this study can be considered as satisfactory, as well as promising for future research given the fact that the use of semantic vectors in the current approach is simple. More complicated features based on semantic vectors can be considered. Further, a structured study to feature importance and selection will be needed in order to determine which features do in fact contribute and which do not. It can be beneficial to annotate polite sentences as well so that the binary classification problem becomes a 3-class-classification problem consisting of neutral, polite and rude sentences.

References

Baroni, M., Dinu, G., & Kruszewski, G. (2014). Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd annual meeting of the association for computational linguistics* (Vol. 1, pp. 238–247).

Bird, S. (2006). Nltk: the natural language toolkit. In *Proceedings of the coling/acl on interactive presentation sessions* (pp. 69–72).

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273–297.

De Smedt, T., & Daelemans, W. (2012). Pattern for python. *The Journal of Machine Learning Research*, *13*(1), 2063–2067.

Drucker, H., Wu, D., & Vapnik, V. N. (1999). Support vector machines for spam categorization. *Neural Networks, IEEE Transactions on*, *10*(5), 1048–1054.

Erk, K., & Padó, S. (2008). A structured vector space model for word meaning in context. In *Proceedings of the conference on empirical methods in natural language processing* (pp. 897–906).

Harris, Z. S. (1954). Distributional structure. *Word*.

Hsu, C.-W., Chang, C.-C., Lin, C.-J., et al. (2003). *A practical guide to support vector classification.*

Khoo, A., Marom, Y., & Albrecht, D. (2006). Experiments with sentence classification. In *Proceedings of the 2006 australasian language technology workshop* (pp. 18–25).

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Levy, O., & Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems* (pp. 2177–2185).

Levy, O., Goldberg, Y., & Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, *3*, 211–225.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111–3119).

Řehůřek, R., & Sojka, P. (2010, May 22). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* (pp. 45–50). Valletta, Malta: ELRA. (`http://is.muni.cz/publication/884893/en`)

Rong, X. (2014). word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.

Saif, H., He, Y., & Alani, H. (2012). Semantic sentiment analysis of twitter. In *The semantic web–iswc 2012* (pp. 508–524). Springer.

Schäfer, R., & DFG, L. W. C. (n.d.). Processing and querying large web corpora with the cow14 architecture.

Zhang, D., Xu, H., Su, Z., & Xu, Y. (2015). Chinese comments sentiment classification based on word2vec and svm perf. *Expert Systems with Applications*, *42*(4), 1857–1863.

Appendix A Figures

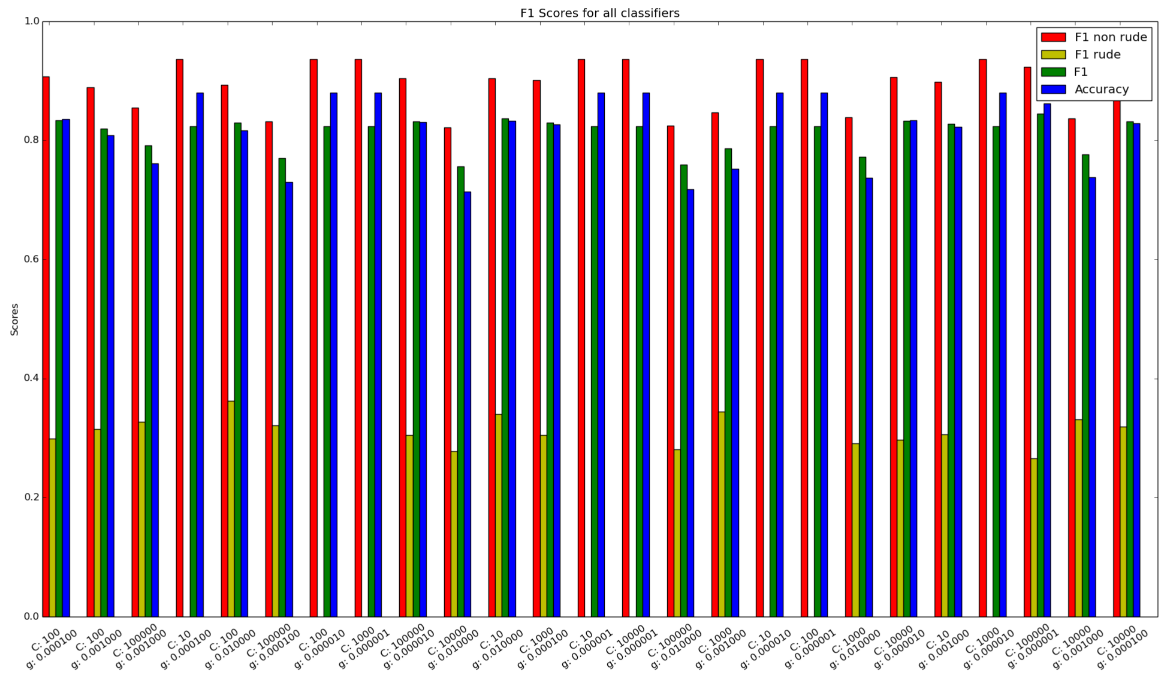


Figure 5: F1 scores and accuracy for various classifiers

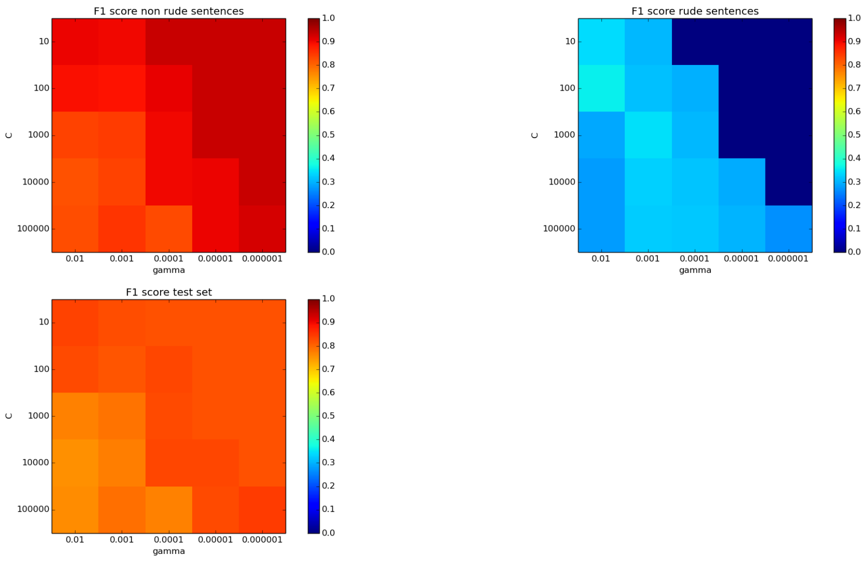


Figure 6: F1 scores for GridSearch for feature set 2