27-4-2018

Sentiment Analysis of Amazon Game Reviews

Coursework 2 – Intelligent Systems

André van der Laan;Ianto Horton-Jones

Table of contents

1. Introduction

Sentiment analysis is a way to use computers to gather information on a piece of text and predict whether the sentiment of the text is either positive, negative or neutral. There is a lot of text online that includes sentiment from individual people across many different topics. For example, product reviews on various online stores, movie reviews on IMDB and social media updates, i.e. twitter. Sentiment analysis is a useful tool that allows researchers to gauge the overall sentiment from many pieces of text on a certain topic.

There are many possible applications of sentiment analysis, it can be used to gather people's opinions on an election using tweets from Twitter (Joyce & Deng, 2017), organisations can use it to gather the publics' opinion on the quality of different hospitals around the country (Greaves, et al., 2013). Sentiment analysis is also useful to predict the movement of stocks over a period of time (Hai Nguyen, et al., 2015).

Sentiment analysis uses parts of text, such as words, phrases and emoticons in order to determine the sentiment of words, sentences, paragraphs or documents. To give the analysed parts sentimental value, lexicons, machine learners or hybrids can be deployed. A lexicon is a dictionary that translates words into sentimental values and bases its classification on the sentimental outcome. A machine learner learn by training and tries to predict the sentimental value. A hybrid is a mixture between a lexicon and a machine learning algorithm, where the lexicon is one or multiple feature(s) of the machine learner.

The papers' goal is to predict the sentiment of Amazon game reviews as either positive or negative. For the prediction, several machine learners will be trained with two different strategies. The first strategy is to predict the sentiment of the reviews based on words and/or phrases and the second strategy is to apply a lexicon as a feature of the machine learner. The machine learners are Naïve Bayes, SVM and Random Forest and each of them will be trained with 10,000 reviews. Along with building a most optimal machine learner, the research will give possible alternatives and recommendations for future works.

This paper tries to use the best proven methods for machine learning on sentiment analyses. Ten literature reviews over sentiment analyses were evaluated in depth to gather information about this topic. The paper starts with important definitions related to the topic, followed by the ten literature reviews. The next section of the paper contains the methodology, which describes the theory, dataset, machine learners and settings of the research. The paper will be finished with the experiments and conclusion.

2. Definitions

2.1 Features

2.1.1 Natural Language Toolkit/Software

This is a toolbox to understand text and its context and to feed the classifier with this information. The toolbox analyses the content and creates relations between the words and identifies words such as dates, locations and persons. These kind of tools gives the opportunity to understand text better so that the classifier can improve its performance. In sentiment analysis the understanding of the text really important. The meaning of words can be different between documents and these toolkits help the classifier to understand their meaning. Stamford CoreNLP and NLTK are Natural language toolkits that can be used for this task.

2.1.2 Negation phrases (NOA and NOV)

Each word in a sentence will have a sentiment value of either positive, negative or neutral, this sentiment will allow the classifiers to make an accurate prediction. Negation phrases are used in sentiment analysis to invert the sentiment of words that have a negative prefix. For example, the phrase “not good” without negation would have a positive sentiment even though it is negative in reality. Using negation would use the “not” and invert the sentiment of the “good”.

This method adds context to the words which can lead to a more accurate prediction of sentiment. When negation phrases are used, that chances of false positives and false negative results being classified are reduced. In the work of Fang and Zhan (2015) negation phrases were categorised in to negation-of-verb and negation-of-adjective because verbs and adjectives had different sentiment scores.

2.1.2.1 Negation Phrases Identification

To identify negation phrases, each sentence will be analysed word by word looking for negation phrases. If negation is detected, it looks up if the next word is an adjective or verb. If this is the case, the looked up words’ sentiment will invert. Else it skips a word and looks up if this is an adjective or verb. If this the case, the sentiment of the lookup word will be inverted. If this is not the case, no words’ sentiment will be altered. This process repeats for all words in the sentences (Fang & Zhan, 2015, p. 5).

2.1.3 N-gram (Bag-of-words)

This phrase of n words. The phrase start by its index and ends by n or by end of the text. The following method is applied: $phrase = index \dots index + Min(n, Lenght - index)$

If n is:

- One, unigram
- Two, bigram
- Three, trigram

For feature creation the index moves by one and repeats until the length of text is exceeds.

2.1.4 Parts of speech (POS) tagging

Words can have different meanings in different contexts. In some cases a word can be noun, but in another case a verb. POS is used for disambiguation of word-categories. This is done by identifying the context where the word is placed. In the example: "The man still saw her", the word "man" could be a noun or verb. In this example POS marks the word "man" as a verb.

Part of speech could be useful for sentiment analyses because it helps to understand the content of the text better, which leads to more accuracy.

2.1.5 TF and TF-IDF

TF and TF-IDF is a technique for weighting words in a bag-of-words. TF stands for term frequency and is annotated as $tf_{t,d}$, where subscripts is the term and the document. Each document d has its own weighting for a term t. The order of words is ignored and only the number of occurrences is counted (Manning, et al., 2008).

TF has the limitation that stop words have a high weighting, because of the high amount of distribution in documents. The words are clearly not interesting for classification of documents (Manning, et al., 2008).

TF suffers from frequently used words, which are not able to discriminate the sentence of the document. For example, documents about cars often use the word "car", but appears too often in the documents and is then not helpful to discriminate a document anymore. The idea is that words that appears lesser amount of time in the documents are more powerful to discriminate the documents. TF-IDF stands for term frequency – inverse document frequency and weight opposite to TF. TF-IDF is able to solve the limitation of TF and weight by the following conditions (Manning, et al., 2008):

- The weight of a term t is higher if the distribution of the term appears in small amount of documents more often.
- The weight is lower if the term t occurs in many documents
- The weight is lowest if it appears in almost all the documents.

For calculating, the weight of the term t in a document d, several steps are needed. In the first step the distribution of the term t in all the documents is calculated by the following method (Manning, et al., 2008):

$$idf_t = log\frac{N}{df_t}$$

N is the amount of documents and df is the frequency of a term in the document. In the second step the same method is applied as in TF. To calculate the actual weight of a term in a document (TF-IDF), the outcome of TF is multiplied by the outcome of IDF (Manning, et al., 2008).

$$tf\_idf_{t,d} = \ tf_{t,d} * idf_t$$

2.1.6.1 Normalization

Comparing between documents with almost the same content, but with difference sizes is difficult. In a larger document terms can have larger weights than in a shorter document. The solution is to normalize weighting of a term t by the Euclidean (L2) norm. L2 normalizes a weighting by dividing the weight through the square root of the sum of all the squared absolute weightings of all the terms (Manning, et al., 2008).

$$norm_{l2} = \frac{v}{\sqrt{\sum_{i=0}^{N}|v_i|^2 + \ |v_{i+1}|^2 + \cdots + |v_N|^2}}$$

2.1.6 Topic Modelling

2.1.6.1 LDA-based method

The LDA-based method is a form of topic modelling that assumes that a given piece of text is made up of multiple different hidden topics. The LDA model is only able to see words as the model is based on bag-of-words. Each word has a hidden topic, therefore all documents are made up of many different topics. When training the LDA model, each document is assumed to contain words that are all related to each other (Knispelis, 2016).

The user defines two hyperparameters when training the LDA model, alpha α and beta β. Alpha defines the topic distribution per document. If the value of alpha is high then every document will contain most topics, if the value of alpha is low then every document will include less topics. Beta defines the distribution of words per topic. A high beta value means that many words will be included in each topic, a low beta value means that there are only a small number of words per topic. The user can also define the amount of topics created (Knispelis, 2016).

The LDA model may not be the best method to determine sentiment in text as it does not consider the sentiment of the words, it only views words as hidden topics.

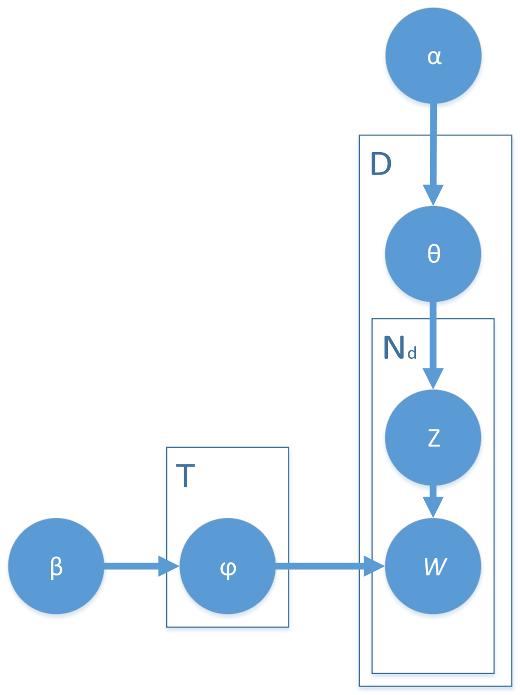

Figure 1 Graphical representation of LDA model ***(Hai Nguyen, et al., 2015)***

Notation	**Definition**
α, β	Hyperparameters
φ	The distribution over words
T	The number of topics
θ	The message specific topic distribution
z	A topic
W	A word in the message *d*
N_d	The number of words in the message *d*
D	The number of messages

2.1.6.2 JST-based method

The Joint Sentiment-Topic Model (JST model) is a model based on the LDA model. The difference between the JST model and LDA model is that the JST model has been designed to model the sentiment of documents. This is done by adding a sentiment layer between document and topic layers. This allows for sentiments to be assigned to topics, thus words will be assigned with topics and sentiment labels (Lin, et al., 2012).

This would be more useful for sentiment analysis than the LDA model as each word has a sentiment value than would affect the classification of the label.

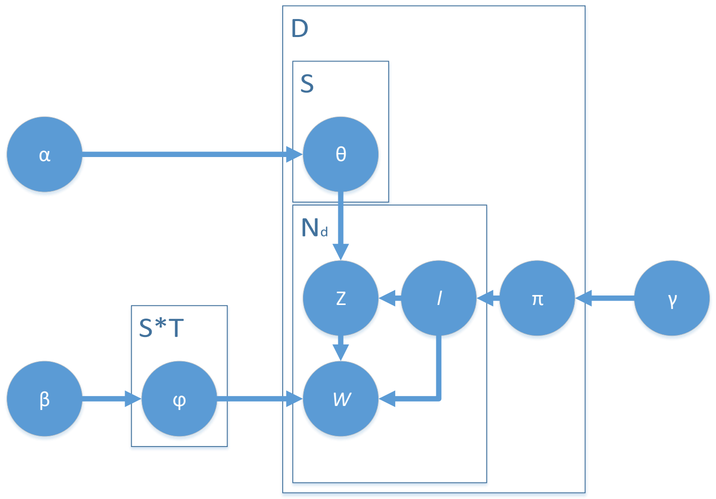

Figure 2 Graphical representation of JST model ***(Hai Nguyen, et al., 2015)***

Notation	Definition
α, β, γ	Hyperparameters
φ	The distribution over words
T	The number of topics
S	The number of sentiments
θ	The message specific topic distribution
z	A topic
W	A word in the message *d*
l	A sentiment label
π	The message specific sentiment distribution
N_d	The number of words in the message *d*
D	The number of messages

2.1.6.3 Aspect-based method

In the work of Hai Nguyen et al. (2015) the aspect-based method is based on each piece of text being represented as a topic and their corresponding sentiment values. This differs from LDA and JST methods where words were made up of hidden topics and sentiment values. In ths method, the consecutive nouns in each sentence are the topic, and the sentiment of opinion words as defined by SentiWordNet[1] with in the sentence will be assigned to that topic.

To use this method Hai Nguyen et al. (2015) extracts all consecutive nouns from the training dataset to be used as topics but disregards any topics that appear less than 10 times in order to avoid rare topics. Sentiment values are then extracted from the opinion words in the sentences that contain topics, the negative and positive sentiment scores were then combined for each topic to create an opinion value. The closer an opinion value is to the topic phrase, the more value the opinion value will have.

This could be useful for sentiment analysis as it considers the sentiment of all sentiments that include consecutive nouns, which could lead to a higher accuracy of classification of labels.

[1] http://sentiwordnet.isti.cnr.it/

2.2 Classifiers

2.2.1 AdaBoost

This classifier works by dividing space into regions. It gives the classifier of the region a weight based on misclassification, a lower amount of correct predictions has a higher weight than a large amount of incorrect predictions. The prediction of the instances in a region is done by a decision tree. This process repeats several times, based on the amount of boost given by its parameter. Afterwards the weights are combined together and comes up with a strong classifier. The strong classifier has surprisingly good accuracy (Dangeti, 2017).

2.2.2 Decision trees

This is a classifier based on a tree. The tree is trained during a training phase. Each node helps the tree to predict the instance. By leaving the node it follows a branch based on the values of the features. Each node "proceeds" one feature. If the instance reaches a leaf, its label is then predicted based on the label that is assigned to the leaf (Joshi, 2017).

Decision trees are applied as weak learners in Random Forest and some research papers about sentiment analysis use this classifier, such as the research of Greaves et al. (2013), Singla et al. (2017), Singh et al. (2017). Singh et al. (2017) argue that a decision tree based classifier suited better than other classifiers on larger datasets.

2.2.3 (Multinomial) Naïve Bayes Algorithm

Naïve Bayes is based on the Bayes' theorem. It predicts by calculating the probability to a label. The predicted instance gets a label based on the highest probability. Each feature has probabilities to each of the labels. To predict the label of an instance it counts all the probabilities of the instance features together and predicts the instance based on their feature probabilities. The algorithm is Naïve because it makes assumptions based on the individual feature rather than the correlation between the features (Bonaccorso, 2017).

Multinomial is useful for models with vectors, where every vector for example represents the number of occurrences of a term. In sentiment analysis it is regularly applied to predict documents based on n-grams. Naïve Bayes has a good performance in natural language processing and prediction of text (Bonaccorso, 2017).

2.2.4 Random Forest

Random Forest is a strong learner and contains several weak learners. In Random Forest, the decision trees are the weak learners. All trees are utilized for prediction of the instance and are called as estimators. In the training phase the features are randomly distributed over the estimators, each estimator has its own subset. Each estimator of the Random Forest gives its own result based on the prediction. All these results will be merged to a single prediction. The merging is done by averaging the results of the estimators (Bonaccorso, 2017).

Fang and Zhan (2015) research shown that Random Forest is very useful for sentiment analysis. This research showed that Random Forest performed well when used for sentiment analysis.

2.2.5 SMO learning algorithm

Sequential Minimal Optimization (SMO) is designed by Platt (1998) and is a derivative of SVM (see: "**2.2.7 SVM**") and it differs in the way that that the hyperplane is projected in the hyperspace. Platt (1998) claims that SMO is conceptually simple and easy to implement. They says also that is faster and more efficient than SVM.

2.2.6 SVM

This is classifier that's able to solve linear and non-linear problems. SVM and neural networks are able to solve high computational complex problems, but for the solving of this problems it needs more resources than other algorithms. Besides the extra needed resources it gives in the most cases significant better results than other algorithms. SVM good in solving problems with relative high dimensions. It is proven that it is good in classifying text, which language has large amount of words (Dangeti, 2017). This makes SVM useful to use for sentiment analysis.

SVM works with a hyperspace and place the instance in the hyperspace. The instance are called vectors and the dimension of the hyperspace is equal to the amount of features. This only for linear, non-linear scales the linear hyperspace up to a higher dimension. To classify the instances the hyperspace is divided in to separate spaces, where the space is divided by a hyperplane. The hyperplane has the same dimension as the hyperspace. The hyperplane is placed based on support vectors, this are vectors on the border of its class. SVM place the hyperplane in the centre of the support vectors. The space between the hyperplane and support vectors is the margin (Dangeti, 2017).

2.2.6.1 Support vector classifier

This contains a C penalty parameter, this parameter allows to move the margins closer or further from the hyperplane. In some real-life cases is the dividing of the classes not correctly possible with a hyperspace. A higher C penalty parameter leads to more robust model, where lower C parameter value leads to smaller margin. This parameter gives the opportunity to predict edge cases better (Dangeti, 2017).

2.2.6.1 Kernel functions

With kernel function can the way of mapping the vectors in the space be altered and this not always needs to be a higher-dimensional space. The kernel function can also lead to more efficient classifier. The kernel function can also be used for the kernel trick, the trick is to handle the data different with this it is possible to classify non-linear data with SVM (Dangeti, 2017).

2.3 Measurement

2.3.1 F-measure

Supervised learning algorithms predict labels from the instances. To measure the performance and accuracy of the algorithm, f-measure is often used. It measures the performance of a given label. If the instance is correctly predicted as the same label as the given label then it is true positive. It is false positive if the algorithm incorrectly predicts that label of the instance is the same as the given label. It is true negative if the algorithm correctly predicts that the label of the instance is different from the given label, and it is false negative if the algorithm incorrectly predicts that the label of the instance is different from the given label. This results in a confusion matrix. The recall and precision is calculate from the confusion matrix.

The precision value indicates the accuracy of the algorithm and can be calculated using the following formula:

$$precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

The recall value shows how many true positives were correctly identified using the algorithm. The recall can be calculated using the following formula:

$$recall = \frac{TruePositives}{TruePositives + TrueNegatives}$$

The F-measure is calculated by using the values of recall and precision by using the following formula:

$$Fmeasure = 2 \times \frac{precision \times recall}{precision + recall}$$

The overall f-measure is the average of the f-measure of all the labels.

2.3.2 K-Fold Cross Validation

In K-Fold Cross Validation the dataset is divided in k folds. K – 1 folds are used for the training and one fold for the testing. To calculate the accuracy the average result of the runs is used. The amount of runs is equal to k. Each time another fold becomes the test portion and the rest (k – 1) is the train portion.

This approve is used to measure the accuracy of the classifier over a given dataset. By increasing k the amount of runs and the size of the training dataset increase. But with a higher value of k, the accuracy value of the classifier is more accurate for the given dataset.

$$accuracy = \frac{\sum accuracy_{fold(1)} + accuracy_{fold(2)} + \ \ldots + accuracy_{folds(k-1)} + accuracy_{fold(k)}}{k}$$

I.e.: If a dataset with 5000 samples and 4 fold cross validation is used. Each fold will contain 1250 samples. In the first run, fold 1 will be the test dataset and the rest (k – 1) the train dataset. In the second run fold 2 will be the test dataset and the rest the train dataset. In the third …. Imagine that the accuracy results are as following: 0.8, 0.79, 0.78 and 0.79. The accuracy of the used classifier will be 0.79 in this example.

Table 1 K-Fold Cross Validation example with 4 folds and 5000 samples

Folds	**5000 samples**			
First	1250 test	3750 train		
Second	1250 train	1250 test	2500 train	
Third	2500 train		1250 test	1250 train
Fourth	3750 train			1250 test

2.3.3 Receiver Operating Characteristic (ROC)

The ROC is used as a method to measure the effectiveness of different classifiers. The ROC score is measured between the values of 0 and 1, thus can be interpreted as a probability (Bonaccorso, 2017).

The ROC curve is visualised on a two dimensional plane as shown in Figure 3. The false positive rate is represented on the x-axis, known as specificity, and the true positive rate is represented by the y-axis, known as sensitivity. The more area the curve covers on the plane, the better the result (Bonaccorso, 2017).

The ROC is a useful measurement tool for sentiment analysis because it allows the researcher to easily analyse the rate of true positives to true negatives. This gives the option to compare classifiers with each other and to get more in-depth information of the performance of a classifier.

Figure 4 shows an example of an ROC plane with the results of three classifiers on it from the work of (Fang & Zhan, 2015). This shows that random forest has the highest result of the three classifiers as the curve covers the most area. The worst result is given by Naïve Bayes as the area below the curve is smaller than the other two classifiers.

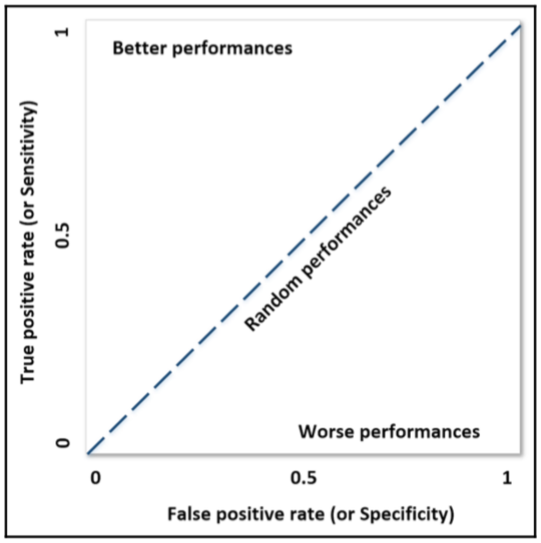
Better performances
Random performances
Worse performances
True positive rate (or Sensitivity)
False positive rate (or Specificity)
0
0.5
1

Figure 3 ROC curve plane (Bonaccorso, 2017)

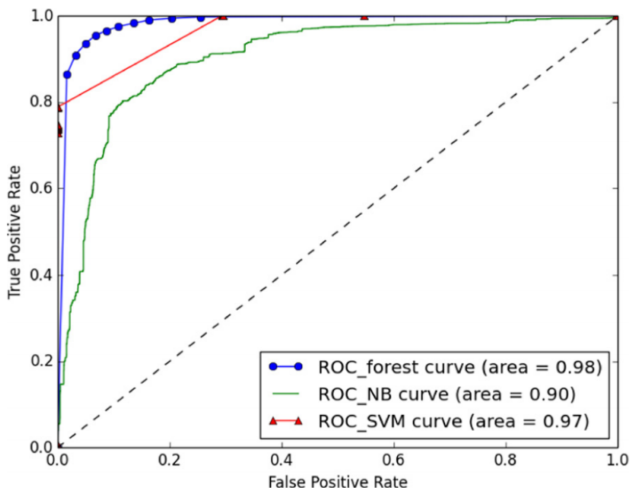

Figure 4 Example of ROC curves (Fang & Zhan, 2015)

2.3.4 Spearman’s rank correlation coefficient

Compare rankings of two instances against each other and calculate the equality of the ranks. The result is measured in rho and is a value between +1 and -1. A higher value of rho means that the ranking of the two instances are more equal. Spearman’s applied the following method:

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$$

$$n = number\ of\ observations$$

$$d_i = difference\ between\ two\ ranks\ of\ each\ observation$$

2.3.4.1 Example

This example explains the Spearman’s rank correlation coefficient based on fictive ranking of pets. There are two rankings, ranking x and ranking y. The rankings and d_i for each ranking is listed in Table 2 Spearman's rank correlation coefficient example.

Table 2 Spearman's rank correlation coefficient example

Pets	Rank x	Rank y	d_i	d_i^2
Cat	1	2	-1	1
Dog	2	3	-1	1
Parrot	3	1	2	4
Hamster	4	4	-1	1
Mouse	5	5	-1	1
Turtle	6	6	-1	1
Snake	7	7	-2	4
Tarantula	8	9	0	0
Lizard	9	8	-1	1
Rat	10	10	0	0
			$\sum d_i^2$	14

The formula $\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$ is used to calculate the Spearman's rank correlation coefficient and $\sum d_i^2$ equals to 14.

$$\rho = 1 - \frac{6 * 14}{10 * (10^2 - 1)} = 0.9\overline{15}$$

In this example the result is 0.915, which would mean that the rankings are almost the same. The table with the example shows that the ranks from 1 – 3 and 8 – 9 are different.

2.3.4.2 Use cases

Spearman’s is useful to verify results and compare predicted ranks to the actual ranks. This gives the opportunity to measure the accuracy of the predicted ranks. Kessler et al. (2015) used Spearman’s to compare the predicted ranks to the actual ranks. Greaves et al. (2013) use Spearman’s to compare predictions to the results of the survey.

2.4 Miscellaneous

2.4.1 Microblogging

This is a limited sized message that can be used to express opinion or current events to the public. Twitter is an example for microblogging. Microblogging often provides the users the ability to add mentions and hashtags to the microblog. With these it is possible to group, categorize and to search for specific topics.

Microblogging is useful to gain information about users' sentiments of given topics. Twitter is common source for sentiment analysis.

3. Literature review

To support this document we will investigate ten papers and relate them to our own work. All the reviews are related to sentiment analysis and each of the papers has its own topic that they had investigated. Each chapter reviews one of the papers and how relates it to our own work.

A review contains an introduction, dataset explanation, features, classifiers and a summary of the results. Several papers use twitter as source for their papers (Pak & Paroubek, 2010; Kouloumpis, et al., 2011; Joyce & Deng, 2017). For other papers (Whitelaw, et al., 2005; Fang & Zhan, 2015; Kessler, et al., 2015; Singla, et al., 2017), review and online database datasets were applied as sources. Alongside these kind of sources were also comments, surveys and historical stock information used by several papers (Greaves, et al., 2013; Hai Nguyen, et al., 2015; Singh, et al., 2017).

3.1 The Good the Bad and the OMG!

In this paper Kouloumpis et al. (2011) perform a sentimental analysis based on tweets. Kouloumpis et al. (2011) goal is to investigate if features as part-of-speech tags are useful in sentimental analysis. For the prediction of the labels of the instances they use hashtags and an emoticon dataset. The labels are positive, negative or neutral. For the evaluation a manually annotated dataset produced by the ISieve Corporation is used (Kouloumpis, et al., 2011).

3.1.1 Datasets

3.1.1.1 HASH dataset

Dataset based on hashtags is a subset of the Edinburgh corpus that contains 97 million tweets. To reduce this amount all the duplicated and non-English tweets are removed, but also tweets without hashtags. From the left overs Kouloumpis et al. (2011) searched for the most common used hashtags with the criteria that they are easy to classify in the positive, neutral or negative category. Each sorted out hashtag was assigned a category for later use for classifying the tweets and to train the machine learning algorithm. All the tweets that include these hashtags are placed in the final dataset (HASH) (Kouloumpis, et al., 2011).

3.1.1.2 Emoticon dataset

The second used dataset is a dataset of tweets with emoticons (EMOT). Tweets with contradictory emotions are not contained in this dataset. This dataset is created by project of Stanford University. From this dataset Kouloumpis et al. (2011) only used the training dataset. The majority of the tweets are not containing any hashtags. The size of the dataset is 381,381 tweets, where 230,811 are positive and 150,570 are negative (Kouloumpis, et al., 2011).

3.1.1.3 Evaluation dataset

For evaluation the ISieve dataset was used that contains approx. 4000 tweets. This is a hand-annotated dataset and Kouloumpis et al. (2011) use this dataset only for the evaluation.

3.1.2 Increasing performance of the POS tagger

To improve the performance of the POS tagger, Kouloumpis et al. (2011) replaced all the abbreviations and resolve them to their meaning and handle it as a single token. Further Kouloumpis et al. (2011) replaced all the repeated characters to a single character, transform all-Cap words to lowercase and removing all the placeholders as “#” and “@”.

3.1.3 Features

The used features is a combination of n-gram, lexicon, Part-of-speech and Micro-blogging. For n-gram unigrams and bigrams will be used in the form of bag of words. The n-gram features are limited to 1000 and negation terms are appended to the follow word. Besides the n-gram, Kouloumpis et al. (2011) added three extra features based on the MPQA subjectivity lexicon (Wilson, et al., 2009). In this lexicon the words are marked as positive, negative or neutral. The value of the features depends on the presence of given words in the tweet and lexicon. Further Kouloumpis et al. (2011) add Part-of-speech features to the feature set. Micro-blogging features are also added and these contain the features to capture in binary form on presence of blog specific characteristics like emoticons, abbreviations. For the emoticons and abbreviations the Internet Lingo Dictionary (Wasden, 2009) is used (Kouloumpis, et al., 2011).

3.1.4 Results

For the features there are multiple subsets used as described in the previous section. Kouloumpis et al. (2011) target is to gain information about these features and their affectivity, but also to evaluate the used datasets.

For the first set of experiments, Kouloumpis et al. (2011) used 10% randomly from the HASH for validation and the rest of HASH for the training. AdaBoost.MH (Schapire & Singer, 2000) was used for the training with 500 rounds of boosting. This process was repeated ten times and averaged afterwards. Kouloumpis et al. (2011) also did training with SVM that gave similar trends, with lower overall results.

For the next experiments Kouloumpis et al. (2011) extended the used dataset in experiments with 19,000 instances from the EMOT. The instances are randomly chosen and equaly divided between positive and negative. From this stage Kouloumpis et al. (2011) calculated the average F-measure based on the validation set to get the performance upper-bound. Kouloumpis et al. (2011) did the training with only n-grams and with all the features. The use of EMOT gave boost in performance compared to HASH only.

After this Kouloumpis et al. (2011) evaluate test data with ISieve dataset. For the training Kouloumpis et al. (2011) used to following features:

- N-gram + lexicon
- N-gram + POS
- N-gram + lexicon + Micro-blogging
- All the features

The third combination of features gave the best performance with 0.68 F-measure and 0.74 accuracy for HASH + EMOT and 0.65 F-measure and 0.75 accuracy for HASH only. The use of POS tagger decreased the performance of the classifier. The reason is unclear and needs to be researched by other work. The reason could be that it is not useful for microblogging or the POS tagger lacks accuracy. The EMOT data increased the performance of the classifier, but it drops or disappears in the moment that the microblogging features are used (Kouloumpis, et al., 2011).

3.1.5 Conclusion

Kouloumpis et al. (2011) concluded that the use of hashtags and microblogging features are useful and that the use of POS tagger needs more research. The performance boost by emoticons almost disappear if microblogging features are utilized.

3.1.6 Related

We want to experiment with the lexicon in our research and find out if in our case the lexicon will also improve the performance of the classifiers. Kouloumpis et al. (2011) use a combination of unigrams and bigrams and we want to investigate if this combination gives a better score than unigrams. Kouloumpis et al. (2011) paper use SVM and we would also like to use it for classification of the Amazon game reviews.

3.2 2016 US Presidential Election

In 2016 Joyce and Deng (2017) researched the opinions of American voters for their presidential election. For the research was Twitter used. The target of Joyce and Deng (2017) was to predict tweets and then to compare them with polling data to see which correlation they share. A lexicon approve and a Naïve Bayes Algorithm was used to predict the tweets and to classify positive or negative (Joyce & Deng, 2017).

3.2.1 Lexicon

For the lexicon approve Joyce and Deng (2017) merged two lexicons and used them together. The labels of the lexicons are not used, but replaced by the label positive or negative. Words that directly refer to a presidential candidate are removed. For the calculation of the sentiment score of the tweet the following method was applied: $score = \sum postive\ words\ - \sum negative\ words$

If the score is positive, the label of the tweet is positive and if the score is negative, the label of the tweet is negative and else the label is neutral. Neutral labelled tweets are removed from the analysis (Joyce & Deng, 2017).

3.2.2 Naïve Bayes Algorithm

For the other approve, Joyce and Deng (2017) use Naïve Bayes Algorithm with the words in the tweets as features. All the words shorter than three characters and all the stop words are removed. For the implementation, the Nation Language Toolkit (NLTK) was used. The training set contains 500 positive, 500 negative tweets and the same tweets that were also used for the lexicon analysis. All the tweets are hand labelled. From the tweets the obvious misspellings are removed. To reduce the human workload, Joyce and Deng (2017) try to classify the tweets based on the hashtags in the tweets. In the end there was 5000 positive and 5000 negative tweets for each candidate (Joyce & Deng, 2017).

3.2.3 Analyse

The total analysed data was roughly 3.068.000 tweets with the mention of Donald Trump and 4.603.000 for Hillary Clinton. There was some overlap between the tweets. The used polling data was from Pollster.com.

Joyce and Deng (2017) found out that the lexicon did not work as expected because the lexicon couldn’t operate by the following assumption: Negative tweets for one of the candidates is a positive tweet for the other candidates. Joyce and Deng (2017) also found out that the Naïve Bayes algorithm did not have this fault and did follow their assumption.

3.2.4 Result

Naïve Bayes algorithm did it well by it identifying from tweets if the associated hashtags was used for the given candidate, but it didn’t outperform the lexicon analysing compared to Trump polling data. The automatically labelled tweets had better performance than the hand labelled ones. Joyce and Deng (2017) conclude that automatic labelling saves time, removes potential bias and increases performance.

3.2.5 Related

Our work will also contains unigrams as features for the classifiers. Joyce and Deng (2017) use Naïve Bayes which seems to be useful in text prediction, but we want to experiment if this classifier will get better results in comparison to SVM and Random Forest.

3.3 Sentiment Analysis of Customer Product Reviews Using Machine Learning

In the paper of Singla et al. (2017) Amazon reviews of Mobile Phones were used to perform sentimental analysis. The collection contains 4 million reviews from 4500 mobile phones and is pre-processed by removing noisy data. Singla et al. (2017) used Naïve Bayes, Support Vector Machine (SVM) and decision Tree to classify the reviews in the class positive or negative. For evaluation of the classifiers 10 Fold Cross Validation was to be used. 10 Fold Cross Validation means data is divided in to ten subsets and each time one subset is used as the test data and the other subsets as the training data (Singla, et al., 2017).

3.3.1 Labelling of reviews

All the reviews are labelled based on their polarity. The polarity is based on positive and negative words. Words are determined negative or positive by the underlining emotion. Which word is associated with emotion is described in the used lexicon. In the research (Singla, et al., 2017) used the NRC sentiment dictionary from the "Scuzhet" package. To calculate the polarity, the following method is used:

$$polarity = \sum positiveScore - \sum negativeScore$$

The research contains only positive and negatives reviews.

Due larger amount of positive reviews compared to the negative, the predictions could be imbalanced. To solve this problem, Singla et al. (2017) removed a portion of the positive reviews to make the amount between positive and negative equal. From this set of reviews, Singla et al. (2017) took 3000 random samples.

3.3.2 Classification

For the classification Naïve Bayes, SVM and decision tree were used. The training and testing was done by 10 fold cross validation. SVM gave the best results with the accuracy of 81.75%. The decision tree had accuracy of 74.75% and Naïve Bayes accuracy of 66.95%.

Singla et al. (2017) came to the conclusion that the rating of review is not always justified the underlying sentiment of the reviewer and therefore not be a trustworthy source for other buyers.

3.3.3 Related

Singla et al. (2017) labels the reviews with a lexicon calculates from them the polarity value. Where positive value is a positive label and vice versa. We will also deploy this method, but as a feature and not for labelling the reviews. In the experiments we will research if the SVM scores the best results and if Naïve Bayes is faster than all other classifiers in our experiments as were the results of Singla et al. (2017).

3.4 Towards Opinion Mining from Reviews for the Prediction of Product Rankings

In Kessler et al. (2015), Amazon and snapsort.com are used as sources for ranking camera products based on sentiment information. To achieve the best performance, Kessler et al. (2015) utilized multiple approves and compares it with the Spearman's rank correlation coefficient. The ranking of the given approve will be compared against sales ranking in the Amazon's case and in the Snapsort's case the ranking will be compared against the quality ranking.

All approves resulting a score for the given product p, where a higher value means a higher position in the ranking for the given approve (Kessler, et al., 2015).

3.4.1 Dictionary approve

The first approve uses a baseline two kinds of dictionaries, the first based on inquirer dictionary (GI) (Stone, et al., 1996) and the second on MPQA subjectivity clues (MPQA) (Wilson, et al., 2005). To calculate the score of a product p is the following method applied:

$$Score_{dict}(p) = pos(p) - neg(p)$$

The score is the sum of all positive words in all reviews minus the sum of all negative words in all reviews. To fit better on longer reviews, Kessler et al. (2015) normalized the method by applying the following method, where the parameter "all" the amount of reviews is:

$$\overline{score_{d\imath ct}}(P) = \frac{score(p)}{all_p}$$

This method forms their second approve and is an extension of the first approve. The first is easy to implement, but doesn't take context into account. The second is more useful in combination of machine learning with context (Kessler, et al., 2015).

3.4.2 JSFA

For this method, the first and second approve is used in combination of JSFA (Joint Fine-Grained Sentiment Analyses Tool)[2] (Klinger & Cimiano, 2013) to calculate the score. These approves are referred as JFSA and JFSA-NORM (Kessler, et al., 2015).

3.4.3 CSRL

As fifth approve is CSRL (Comparison Sematic Role-Labeler) (Kessler & Kuhn, 2013) that scans the reviews for prefers of the current product and the non-prefers (prefers to other products). For the calculation of the score is the following method is used:

$$score_{CSRL}(p) = pref(p) - npref(p)$$

(Kessler, et al., 2015)

[2] https://bitbucket.org/rklinger/jfsa

3.4.4 Stars and number of reviews

The 6th and 7th approve is not based on the content of the reviews, but the 6th is based on the average star rating of all the reviews and the 7th is based on the amount of reviews. The parameter “r” is the rating of the review (Kessler, et al., 2015).

$$score_{star}(p) = \sum_{i=0}^{all_p} r_{pi} \Big/ all_p$$

$$score_{NumReviews}(p) = all_p$$

Kessler et al. (2015) believe that products with high rank have a lot of reviews and so a higher score, calculated by: $score_{NumReviews}$.

3.4.5 Experiments

The products used in the research of Kessler et al. (2015) are the retrieved products from Amazon with search term “camera” and “camera” with specific brand name like “Sony”. But also products in the category “’Camera & Photo” with the description “Amazon Best Sellers Rank”. The dataset contains in total 920 products with 71.409 reviews. The titles are reduced to the first six tokens. From the top 150 products in the Amazon sales ranking 56 are found on Snapsort. For gathering, the category “”best overall” is used (Kessler, et al., 2015).

3.4.6 Results

The rankings of both sources are different and gain only $\rho = -0.04$. The NumReviews has the best performance in the case of Amazon with $\rho = 0.33$ and this approve scores by Snapsort only $\rho = 0.1$. Kessler et al. (2015) suggest that one reasons for difference is that the Amazon reviews and products come from the same source and that it don’t give the guarantee that popularity leads to higher amount of reviews and gives higher amount of sales. The popularity factor by Snapsort rating is lower and has much less influence. The reason for the good performance in the case of Amazon is may the chicken-egg dilemma. There are many reviews because the product sold many times.

If the approve is used with the average stars, than the results by Amazon are almost random and not useful. The reason is that some of the products suffer on the amount of reviews and then the value is not representative. This is lesser the case by Snapsort.

The dictionary-based approves results are not perfect and by using normalization the performance decreased, but both dictionary-based approves gain better performance than the JFSA. The score of JFSA decreases if the normalization is applied.

CSRL has mediocre performance with Amazon, because there is lack of references and comparisons in the reviews of the products. This approve depends on this information. By Snapsort gives this method the best overall with $p = 0.51$. (Kessler, et al., 2015)

3.4.7 Related

We plan to conduct experiments with $Score_{dict}$ and $\overline{Score_{dict}}$ for our lexicon feature and to observe if the normalization will reduce the performance in comparison to the non-normalized. Because of Kessler et al. (2015) we do not expect much performance increase from the lexicon feature.

3.5 Twitter as a Corpus for Sentiment Analysis and Opinion Mining

In the research of Pak and Paroubek (2010), the target is to create a sentimental analysis based on Twitter. The dataset of the research contains 300.000 tweets that have positive/negative emotions or based on facts without emotions. Twitter accounts of popular newspapers and magazines was used for fact based tweets. The positive and negative tweets were retrieved based on contained emoticons. Because of the short messages what twitter allowed, they assume that an emoticon has a high influence on the sentiment of a tweet (Pak & Paroubek, 2010).

3.5.1 Features and classifiers

For the features, n-gram was used in the form of unigrams, bigrams and trigrams. The value of the features are binary and based on the presence of an n-gram. Short form of words are treated as a single word. All the stop words are removed from the text. Negations of words were attached to the n-grams to increase accuracy.

In the research of Pak and Paroubek (2010), three classifiers were applied. The multinomial Naïve Bayes, SVM and CRF. The Naïve Bayes gave the best performance. For the training two setups were used, one based on n-grams and one on POS. The improve the accuracy, Pak and Paroubek (2010) reduced the amount of features by using two strategies, the first by removing non meaning n-grams and the second by removing n-grams that occur too few times.

For testing the classifiers hand-annotated tweets were used (Pak & Paroubek, 2010).

3.5.2 Results

In the paper of Pak and Paroubek (2010) bigrams scored the best result. It is a hybrid between unigrams and trigrams. Unigrams are good in coverage and trigrams in sentiment expression patterns. The attachment of negations to n-grams did increase the accuracy of the classifiers, but decreased the recall. Increasing the sample size did increase the overall performance of the system to given point. Removing n-grams that occurred too few times increased the performance and worked better than removing non meaning n-grams (Pak & Paroubek, 2010).

3.5.3 Related

We consider to use negation for n-grams as Pak and Paroubek (2010) wrote that it did increase accuracy, but we believe that it is difficult to implement effectively. There are many factors that are necessary to consider. For correct negation it is important to understand the text and its context.

Pak and Paroubek (2010) use bigrams, which gave the best results and we want to prove if this is also the case in our work.

3.6 Capturing Patient Experience from Free-Text Comments Posted Online

In the paper of Greaves et al. (2013), the authors used online comments from patients about hospitals from the NHS choices website from 2010, the goal was the try and automatically predict the patients' views on a number of topics. These topics were whether the patient would recommend the hospital or not, whether the hospital was clean or dirty and whether the patient was treated with dignity and respect. The algorithm was trained using 13,802 comments that were gathered from the NHS Choices website from 2008, 2009 and 2011 (Greaves, et al., 2013).

3.6.1 Methodology

The pre-processing methods used by Greaves et al. (2013) in the machine learning process were bag of words, prior polarity and information gain. Using the bag of words approach they analysed the total body of words used a combination of unigrams and bigrams as the basic units of analysis, using higher n-grams could improve results but is costlier with computer power and processing time. The prior polarity method consisted of extracting the 1000 most common single words and 1000 most common two-word phrases in the body of text and manually rating each sentiment as positive, neutral or negative for each question under consideration. Information gain was used to remove words that had a lower certainty of being either positive or negative, this is to reduce computation time and achieve more accurate results (Greaves, et al., 2013).

3.6.2 Classifiers

Four different classifiers were used, Naïve Bayes Multinomial (NBM), decision trees, bagging and support vector machines (SVM). Bagging is an ensemble algorithm designed by Breiman (1996). For each method they calculated the accuracy, the F-measure, the Receiver Operating Characteristic (ROC) and the time take to complete the task. In order to reduce the time taken to complete each task they limited the words in the learning process to the most frequently used 10,000 words (Greaves, et al., 2013).

3.6.3 Experiments

The experiments of Greaves et al. (2013) be comparing their results to patient ratings posted on the NHS Choices website where users could recommend the hospital or not and could rate the cleanliness and the level of respect they were given on a Likert scale. A Likert is a scale from one to five that comes in the form of a questionnaire that participants answer to determine opinions on certain topics. The three most positive answers on both were grouped into positive classes while the two most negative answers were grouped into negative classes (Greaves, et al., 2013).

3.6.4 Results

The results of Greaves et al. (2013) shows that there was agreement between the sentiment analysis and the patients rating between 80.8% and 88.6% of the time on whether the patient recommend the hospital, between 83.7% and 84.5% of the time on whether the patient was treated with dignity, and between 81.2% and 89.2% of the time on whether the hospital was clean or not. Bagging took the longest time to compute by far, it took between 2018 seconds and 4871 seconds to complete the tasks. NBM was the fastest and the most accurate overall. All algorithms performed worse with the question whether a hospital is clean or not, this could indicate a lack of words relating to cleanliness in the reviews on the website (Greaves, et al., 2013).

3.6.5 Related

Greaves et al. (2013) research deployed SVM and Naïve Bayes for their machine learners, where Naïve Bayes gave the best results. We will verify if these kind of results will also be shown in our experiments. In the research of Greaves et al. (2013) the combination of unigrams and bigrams was used an gave good results, we want to investigate if these results be better than the results of unigrams.

3.7 Using Appraisal Groups for Sentiment Analysis

In this paper by Whitelaw et al. (2005) the use of appraisal groups are investigated and experimented with to determine whether they can improve on the more common methods of sentiment analysis such as bag-of-words. In this paper Whitelaw et al. (2005) focus only on adjectival appraisal groups rather than nominal or verbal appraisal groups.

3.7.1 Taxonomies of Appraisal

In order to extract appraisal groups, Whitelaw et al. (2005) assigned four main types of attributes, attidude, orientation, graduation and polarity.

Attitude categorises what type of appraisal is being expressed in the expression. Affect refers to a personal emotional state, appreciation refers to evaluation of object properties and judgement refers to a social judgement based on object properties. Attitude can be expressed with nouns, adjectives and verbs. Orientation dictates whether the appraisal is positive or negative. Graduation includes two different factors, force and focus. These are expressed using modifiers, modifiers such as “very” and “slightly” change the force and modifiers such as “truly” and “sort of” change the focus. Polarity can be either marked or unmarked, this depends on whether there has been a negation such as “not” used in the expression. An appraisal group has a head adjective that defines the attitude type and then a list of appraisal modifiers which take into account the words around the head adjective that would change the context of the head adjective. For example “not very good”, “good” would be the head adjective, while “not” and “very” would be the appraisal modifiers (Whitelaw, et al., 2005).

The following list clearly shows the taxonomies of appraisal that are necessary to extract an appraisal group:

- Attitude
  - Affect
  - Appreciation
  - Judgement
- Orientation
  - Positive
  - Negative
- Graduation
  - Force
  - Focus
- Polarity
  - Marked
  - Unmarked

3.7.2 Lexicon

Whitelaw et al. (2005) built a lexicon using semi automated methods in order to give all relevant terms appraisal attribute values. Some words in the lexicon will affect multiple appraisal attributes at the same time. The lexicon contained 1329 terms and covers 29.2% of adjectives in the test set (Whitelaw, et al., 2005).

3.7.3 Feature Sets

Whitelaw et al. (2005) define feature sets as disjunctions of appraisal group atribute values or lexical items. The features are normalized as a variant of "**2.1.5.1 Normalization**".

The feature sets used were:

- Words by attitude
- Systems by attitude
- Systems by attitude and orientation
- Appraisal group by attitude
- Appraisal group by attitude and orientation
- Appraisal group by attitude orientation and force
- Bag of words
- Union of bag of words and appraisal group by attitude and orientation
- Union of bag of words and appraisal group by attitude orientation and force

To test the accuracy of the feature sets, a testbed of 1000 positive movie reviews and 1000 negative movie reviews were used with all neutral reviews removed. Weka's implementation of the SMO learning algorithm was used using a linear kernel and default parameters (Whitelaw, et al., 2005).

3.7.4 Results

In the paper of Whitelaw et al. (2005), evaluation was performed with 40 independently chosen train-test partitions and 10-fold cross-validation with 1950 training and 50 text documents per run. The lowest accuracy score was for the Words by attitude feature set with 77.6 with cross-validation and 77.7 with random select. The bag of words feature set on its own scored 87.0 and 87.6 with cross-validation and random select respectively. The highest scoring was the feature set of the union of bag of words and appraisal group by attitude and orientation with 90.2 for cross validation and 90.1 with random select (Whitelaw, et al., 2005).

3.7.5 Related

Whitelaw et al. (2005) focused on using appraisal groups to classify reviews as either positive or negative rather than using bag-of-words. We believe that this work is out of the scope of what we want to achieve with this document because the method is not related to deploying machine learning with n-grams and a lexicon. The research of Whitelaw et al. (2005) contains a lot of complexity and would overcomplicate our work in the limitted time available.

3.8 Sentiment analysis using product review data

In this paper by Fang and Zhan (2015) they experimented with using sentiment polarity categorization. To do this they conduct experiments on sentence-level categorization and review-level categorization.

3.8.1 Dataset

The data used in the research of Fang and Zhan (2015) are reviews taken from amazon.com from February 2014 to April 2014, in total there are over 5.1 million reviews in this dataset. The reviews are split between 4 categories of products on amazon.com, beauty, book, electronic and home and each review is rated between 1 and 5 stars (Fang & Zhan, 2015).

3.8.2 Features

In the research of Fang and Zhan (2015), all sentiment sentences content was extracted, sentiment sentences are sentences that contain at least one positive or negative word. A part-of-speech (POS) tagger was used for each sentence to identify the words that convey sentiment, 25 million adjectives, 22 million adverbs and 56 million verbs were tagged from all sentiment sentences. To tackle the problem of negation phrases, Fang and Zhan (2015) identified two types of phrases, negation-of-adjective (NOA) and negation-of-verb (NOV) using an algorithm. The paper described that a sentiment score was given to every word token and phrase token, 11,478 word tokens were selected that occurred at least 30 times in the dataset and 3,023 phrase tokens were selected that occurred at least 30 times in the dataset. The formula used to determine the sentiment score for each token was:

$$SS(t) = \frac{\sum_{i=1}^{5} i * \gamma 5, i * Occurrence_i(t)}{\sum_{i=1}^{5} \gamma 5, i * \ Occurrence_i(t)}$$

$$\gamma 5, i = \frac{|S - star|}{|i - star|}$$

The mean and median values for positive word tokens were 3.18 and 3.16 respectively, while the mean and median values for negative values were 2.75 and 2.71 respectively. The research uses a bag-of-words model to add ground truth tags to the sentence-level categorization to determine whether each sentence is positive or negative, if there are more positive tokens than negative tokens in a sentence it will be tagged as positive and vice versa. Training data for review-level categorization did not need ground truth tags as they already used the star ratings. The authors then transform sentiment tokens and sentiment scores into a feature vector for each categorization (Fang & Zhan, 2015).

3.8.3 Experiments

Fang and Zhan (2015) test three different classifiers, naïve Bayes, SVM and random forest and measures the F-measure and ROC of each classification model. ROC is explained in "**2.3.3 Receiver Operating Characteristic (ROC)**". The results are being evaluated using 10-fold cross validation, this technique is explained in "**2.3.2 K-Fold Cross Validation**". For sentence-level categorization, all three classifiers had the same result of 0.85 based on 200 manually-labelled sentences. The ROC values for the manually-labelled sentences showed that random forest had the highest result at 0.92, Naïve Bayes was the second highest with 0.90 and the lowest ROC value was given by SVM at 0.82.

For machine-labelled sentences there were 2 million feature vectors that were generated from 2 million machine-labelled sentences. Four subsets are used, one with 200 vectors, one with 2000 vectors, one with 20,000 vectors and one with 200,000 vectors. The number of vectors with positive and negative labels are always equal. Results of five vector sets will be gathered in order to evaluate the performance of all classifiers, all four subsets and one complete set. The best classifier was random forest, but as the training data increased from 180 to 1.8 million, SVM improved from an F-measure of 0.61 to 0.94.

For review-level classification 3 million feature vectors are formed, ratings of 4 and 5 stars were labelled as positive, 1 and 2-star reviews were labelled as negative while 3-star reviews were used to prepare neutral class vectors. Four subsets are obtained, one with 300 vectors, one with 3,000 vectors and one with 300,000 vectors. The classification models will be evaluated on all subsets and the complete set. The results show that the performance of Naïve Bayes and SVM classifiers are identical but random forest performs worse than both of them. For the complete set, the ROC value for random forest was the highest of the three classifiers a 0.98, the second highest value was SVM with 0.97 and the lowest value was Naïve Bayes with 0.90. (Fang & Zhan, 2015).

3.8.4 Related

This work of Fang and Zhan (2015) use an advanced way to negate verb and adjective by deploying POS. We want to focus on different n-grams, n-gram baselines and lexicon features, therefore we will not use this kind of technique. We think that this method would be good in a future work of ours. Fang and Zhan (2015) contains all the classifiers that we are interested in conducting our experiments with.

3.9 Sentiment analysis on social media for stock movement prediction

The paper of Hai Nguyen et al. (2015) analyses the use of sentiment analysis to predict stock price movement.

3.9.1 Dataset

In the paper of Hai Nguyen et al. (2015), two datasets were used. The first is the historical prices of the 18 stocks they are analysing from Yahoo Finance. The second dataset is the Yahoo Finance message board from July 23rd, 2012 to July 19th, 2013. In 15.6% of the messages a sentiment tag is included, strong buy, buy, hold, sell and strong sell (Hai Nguyen, et al., 2015).

3.9.2 Features

Hai Nguyen et al. (2015) describe that the six sets of features that were used are Price only, Human sentiment, Sentiment classification, LDA-based method, JST-based method and Aspect-based method. In the Price only method, only the historical prices are used to predict the stock movement. In the Human sentiment method, the historical prices and the sentiment included in 15.6% of the messages, disregarding the content of the messages. In the Sentiment classification method, Hai Nguyen et al. (2015) used the 15.6% of the messages with a human sentiment rating to create a test dataset which they would use to determine the sentiment from the 84.4% of messages that did not include a human sentiment rating. Stop words were removed from all messages and the words were lemmatized using the Stamford CoreNLP (Hai Nguyen, et al., 2015).

3.9.3 Experiments

The classification model used to predict the stock price movement was SVM with the linear kernel (Hai Nguyen, et al., 2015). The Aspect-based sentiment feature set achieve the highest accuracy out of all feature sets with an accuracy of 54.41% on average between all 18 stocks. Hai Nguyen et al. (2015) cite that 56% accuracy is deemed a satisfactory result for predicting stock prices, therefore these results are positive. In some specific stocks such as AMZN and DELL the results are 71.05% and 64.47% accuracy respectively, which is very satisfactory in comparison. The average result for the human sentiment feature set was not far behind with an accuracy of 54.25% and the feature set that only considered historical prices scored an accuracy of 52.34%. The lowest scoring feature set was the JST-based model that scored 51.54% accuracy. The researchers state that stock price is difficult to predict because there are many factors involved that can influence it and they believe that the results would be increased if more factors were considered on top of historical prices and sentiments derived from social media (Hai Nguyen, et al., 2015).

3.9.4 Related

The research of Hai Nguyen et al. (2015) was quite interesting, because the results were sometimes really low and other times relatively high and has shown that predicting the stock market is difficult to perform even with a range of methods. It difficult to relate this to our own work and if we would use these methods, then it would be difficult to measure beforehand what we could expect form them.

3.10 Optimization of sentiment analysis using machine learning classifiers

Singh et al. (2017) aims to compare the effectiveness of four different classifier algorithms, the WEKA freeware was used to carry out these experiments. This report details that there are four methods that can be used to determine the sentiment of text, document level, sentence level, word level and character level. To determine the sentiment on character level a neural network is used that can extract features of each character window from a given word (Singh, et al., 2017).

3.10.1 Dataset

Singh et al. (2017) used three different datasets that have been manually annotated. The first dataset was taken from product reviews of a Woodland's wallet on *amazon.in,* 88 reviews were used to create a training dataset while 12 reviews were chosen to create the test dataset. The second dataset is formed of digital camera reviews from Sony, 7465 reviews were used to create the training dataset and 1000 reviews were used to create the test dataset. The third dataset was created using movie reviews taken from IMDB, there are 2421 reviews used for the training dataset and 500 reviews used for the test dataset (Singh, et al., 2017).

3.10.2 Classifiers

Singh et al. (2017) uses four different classifiers in their research, Naïve Bayes, J48 decision tree, BFTREE and OneR. Singh et al. (2017) writes that the Naïve Bayes algorithm uses conditional probability to classify words in to one of three categories, positive, negative or neutral. The Naïve Bayes algorithm is beneficial to their research because it only requires a small training dataset to be effective. For the Naïve Bayes classifier, the text undergoes pre-processing where numbers, foreign words, html tags and special symbols are removed, and words are tagged manually as positive, negative or neutral by experts. The J48 algorithm is a decision tree based classifier and is better suited to larger datasets than many other classifiers. The BFTREE algorithm is another decision tree based algorithm, but it outperforms the J48 algorithm. The difference between this algorithm and J48 is that the BFTREE expands only the best node in depth-first order. The OneR algorithm restricts the decision tree to only one level, thus only creating one rule that makes a prediction on word feature terms. This algorithm has a minimal error rate due to repetitive assessment of word occurrences (Singh, et al., 2017).

3.10.3 Methodology

Singh et al. (2017) use python 3.5 using NLTK and bs4 libraries to pre-process the raw text from the web. The functionality of NLTK is described in "**2.1.1 National Language Toolkit"**. The bs4 library from Richardson (2017) gathers information by following hyperlinks and builds a linked tree with the followed links. From the tree, linked information can be extracted. Singh et al. (2017) obtained 15 features from the first dataset and 42 features from the second and third dataset were obtained. The format of files are converted to ARFF to be compatible with WEKA 3.8 and two sentiment labels are used for assigning sentences, Pos for positive and Neg for negative. Feature selection is implemented using three different methods, Document Frequency, Mutual Information and Information Gain (Singh, et al., 2017). Document frequency is described in "**2.1.5 TF and TF-IDF**". Mutual Information is described in (Cang, 2011) as a symmetric, non-negative measure of the amount of information between two variables, the measure is zero only if the variables are independent. (Lei, 2012), describes Information Gain as the amount of information provided by feature items for the text category and is entropy-based.

3.10.4 Results

The results given in the paper of Singh et al. (2017) is the average of all results after 29 runs. Overall out of the four classifiers, the Naïve Bayes algorithm is by far the quickest to run with an average time of 7.79s with BFTREE the second fastest at 21.12s. The highest F-measure comes from the OneR algorithm with an F-measure of 0.97 and the lowest F-measure comes from BFTREE with a score of 0.721. When testing the algorithms for the datasets, all algorithms on all datasets improved when using 42 features rather than 15 features. The first dataset has the highest accuracy on average across all algorithms, this is possibly because it is a much smaller dataset than the other two (Singh, et al., 2017).

3.10.5 Related

Singh et al. (2017) has shown that Naïve Bayes only needs a small dataset to operate with good results. This behaviour could be beneficial to our predictions of our small dataset of Amazon game reviews. Based on this information and from other works, we consider to use this algorithm in our work.

4. Methodology

Our topic is to classify game reviews on Amazon as positive or negative. This classification will be done by SVM, Naïve Bayes and Random Forest algorithms. The features for the classification are based on n-grams and the Game sentiment lexicon.

Our goal is to compare different forms of n-grams with and without the use of the Game sentiment lexicon using different classifiers. The different forms of n-grams are based on different length combinations and different ways of counting present words in a review. Afterwards we will compare these results with each other based on accuracy and compute time.

4.1 Environment

We use will Scikit-learn for the classification and feature extraction. Scikit-learn is a high-level machine learning platform written in Python. This platform contains state-of-the-art implementations and many well-known machine learning algorithms. Besides the algorithms, Scikit-learn delivers a set of tools that can be used for feature extraction, validation and other purposes that are related to machine learning (Pedregosa, et al., 2011).

4.2 Relation to the Literature

The use cases that we reviewed on the topic of sentiment analysis included work that used social media, stock markets and online reviews to experiment with different approaches of sentiment analysis. Papers from Singla et al. (2017), Kessler et al. (2015), Fang & Zhan (2015) and Singh et al. (2017) used Amazon reviews in their research, we are also using Amazon as a source for our research. We will be able to use their work to measure how succesful our results are by comparing them to the reviewed works that are based on Amazon reviews.

We selected our classifiers based on proven performance on sentiment analysis that we have reviewed in the literature review. We believe that Naïve Bayes, SVM and Random Forest could give good results based on the reviewed work.

The papers of Singla et al. (2017) and Kessler et al. (2015) used lexicon based methods for their research on sentiment analysis. We plan on using similar methods to these in our experiments, the main difference being that we will use the lexicon as a feature of the machine learners and not for labelling the reviews. We want to investigate if the normalization of the lexicon gives drawback in comparison to the non-normalized version of the lexicon as is found in the work of Kessler et al. (2015).

4.3 Dataset

The dataset contains 10,000 game reviews from Amazon and each is labelled as positive or negative. The delivered dataset is already pre-processed and normalized in the following form:

- All the characters are lower-case
- Spaces between all the words and punctuation marks

The dataset may contain sarcasm and unrelated parts of text in certain reviews, which can influence the classification of the given reviews. In this report this limitation is not addressed because this goes out of the scope of what we are aiming to achieve. Our goal is to compare algorithms and utilize the lexicon.

To increase the performance of the classifiers, we normalize the data by removing stop words.

4.3.1 Training and test dataset

For the training and testing we use 3 fold cross validation, that means that each time 33% of dataset is for the test dataset and 66% for the training dataset. Table 3 3 fold cross validation shows how each fold is divided between training and test dataset. For information about K-fold Cross Validation see "**2.3.2 K-Fold Cross validation**".

Table 3 3 fold cross validation

Folds	Dataset (10,000 reviews)		
First fold	6666 training reviews		3333 test reviews
Second fold	3333 training reviews	3333 test reviews	3333 training reviews
Third fold	3333 test reviews	6666 training reviews	

4.3.2 Lexicon

In the research of Singla et al. (2017), they use sentiment lexicon to label the reviews as positive or negative based a polarity value. In our research we will also use a sentiment lexicon to calculate the polarity, but we will be using it as a feature and not for labelling the reviews. The Lexicon that we will use is a Game sentiment lexicon that contains 4,980 words.

In other research, such as Kouloumpis et al. (2011) the use of sentiment analysis increased the performance and accuracy of the SVM classifier. In Kessler et al. (2015), a lexicon approve was used resulting in mediocre performance, but in our case we want it use as a feature and not for the prediction or labelling. Because of the work of Singla et al. (2017) and Kouloumpis et al. (2011), we argue that the use of a lexicon could give our performance a small boost in case of SVM and Random Forest. These classifiers treat the features as dependent on each other and the use of a lexicon has influence on the hyperspace in SVM and the way that the trees are built in Random Forest. In our opinion, the use of a lexicon in Naïve Bayes will be less helpful because this classifier treats each feature independent from each other and would only see this feature as an extra word in the bag-of-words.

4.3.2.1 Game sentiment lexicon

Each word in the Game sentiment lexicon has a score between -5.39 and 4.49. A positive score means that the given word expresses positive sentiment and a negative score means the opposite. More extreme values have larger weight on the sentiment of a word in a review. This could mean that a more weighted word has more influence on score of the review than a less weighted one. I.e.:

A review containing words with the score of -0.2, -0.5 and +1.5, would lead to the total score of 0.8. This happens even then when negatives are majority as demonstrated in the example. In the report we refer to the Game sentiment lexicon as LEX.

LEX has the potential limitation that is described in the example and we assume that it could be prevented by only counting the number of positive and negative words and subtracting sum of negative words from the sum of positive words. The result of the subtraction would be the score, this method will be referred to as LEX-NORM. In the research of Kessler et al. (2015), the normalization gave a decrement in performance, we would like to research if this would also be the case in our work.

4.3.2.2 Lexicon feature – polarity

Each sample gets the feature **polarity** and is calculated based on LEX or LEX-NORM. In some of the experiments this feature will be disabled so that we are able to compare the results against non-lexicon approves. LEX and LEX-NORM are calculated by the following methods:

$$polarity_{LEX} = \sum positive_{score} + \sum negative_{score}$$

$$polarity_{LEX-NORM} = \sum Occurrence(positive) - \sum Occurrence(negative)$$

For the calculation we use single words in the bag-of-words of the review that also appears in the lexicon and all other words in the review are ignored.

Naïve Bayes can't handle negative number features and as workaround we scale the numbers between 0 and 1. The next section will describe the method behind the scaling.

4.3.2.3 Scaling

The **polarity** feature value is considerably large in comparison to the n-grams features. Extremely high and low values lead to problems in the optimization process of the machine learning. Use of different scales can lead algorithms based on linear combinations to domination because of features with larger range or variance. With algorithms such as SVM this is the case Boschetti and Massaron (2015).

Naïve Bayes cannot handle negative number features. To solve this problem we scale the data with MinMaxScaler[3] from Scikit-learn. The **polarity** value will be scaled between 0 and +1.

The scaling is done by the following method:

$$polarity_{scaled} = \frac{polarity - polarity_{min}}{polarity_{max} - polarity_{min}}$$

4.3.3 N-grams

For n-grams we use unigram, bigram and unigram and bigrams. Bigrams had the best performance in the research of Pak and Paroubek (2010). In their research they used unigram, bigram and trigram. Pak and Paroubek (2010) argue that the bigrams is more of a hybrid between unigrams and trigrams. Trigrams has the advantage of expression pattern and unigrams have advantage of good coverage. Pedersen (2001) says that the presence of bigrams is only effective for disambiguation of words.

In the research of Pang et al. (2002) the presence of bigrams only had a negative impact on the accuracy of the classifiers, but the use of unigrams and bigrams was an improvement in performance in comparison to the unigram only n-gram. Pang et al. (2002) says that the use of bigrams goes against the principals of Naïve Bayes.

In other works that we have reviewed, unigram, bigrams and combination of unigrams and bigrams have been used. For example the work of Kouloumpis et al. (2011) and Greaves et al. (2013) used combination of unigrams and bigrams and the work of Joyce and Deng (2017) used unigrams.

[3] http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

In our research we will create n-grams based on Count, Binary, TF and TF-IDF. By using this diversity on creating n-grams, we are able to conclude which of the n-grams gives the best performance in sentiment analysis of Amazon Game reviews. The baseline will have influence on the outcome performance of the classifier and it is possible than some combinations of baselines and n-grams will give bad results. We think there will be some trends between the different n-grams and its baseline performance.

4.3.3.1 Counting word and phrase presents

Each n-gram token that is present in the corpus has a value. The value of a token is different for each document. In our case, a document is a review and a token is a word in the case of unigrams and in the case of bigrams a token is a two word phrase. If a token is not present in a review then it obtains the value of zero. If a token is present, then the value depends on the way that the given baseline calculates the value. For calculating the value, we use Count, Binary, TF and TF-IDF. The Count baseline counts the amount of the same tokens present in a document, the value of the token is the total number of its occurrences. The binary baseline gives a token a value of zero if a token is not present and one if a token is present. TF stands for term frequency and TF-IDF for term frequency – inverse document frequency, for explanations see "**2.1.5 TF and TF-IDF**". For TF and TF-IDF is the Euclidean norm (L2) normalization used, see "**2.1.5.1 Normalization**".

We use the feature extraction tools CountVectorizer[4] and TfidfVectorizer[5] from Scikit-learn to create the n-grams. The first tool is for n-grams based on Count or Binary and the second tool for n-grams based on TF or TF-IDF.

4.3.3.2 Limiting n-grams

We limiting the amount of n-grams to 2000, this will reduce the compute-time and complexity of the classifiers. Beside the reducing compute-time and complexity, we predict that it will increase in accuracy of the Random Forest classifiers because it will reduce the amount of nodes in the trees and therefore reduce the probability of overfitting.

4.3.4 Negation

In the work of Pak and Paroubek (2010) and Fang and Zhan (2015) negations of words were implemented to increase accuracy. In our work we do not apply these methods, not using these methods could lead to a decrement in the performance outcome. In our case decrement could happen in the n-grams and polarity feature based on the Lexicon.

The sentence "This is software does not have a good design" is the opposite meaning to "This is software has a good design". For the machine, the presence of the word "not" is the only difference between them. To create a good negation method it is important to address the negation of the verb, rather than to address it of the adverb. In our example the verb is "design" and the adverb is "good". This makes the implementation difficult because the order of words and the amount of adverbs can always vary between the different sentences. To create a good negation method, the machine needs to understand the text and its meaning. The adverb of "good" would increase the value of the verb "design" and the adverb "not" will gain a stronger meaning because of the context. By adding more adverbs the value of word can

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

increase or decrease. All these variables make it difficult to implement a well done negation of verb with its given value.

4.4 Classifiers

Classification of the reviews will be done by Random forest[6], SVM[7] and Multinomial Naïve Bayes[8] from Scikit-learn. In the most of the referred papers they use one or multiple of these classifiers that we want to use for our research.

4.4.1 Random Forest

Random Forest is an ensemble learner and illustrated in in "**2.2.4 Random Forest**".

It is less often used in sentiment analysis, but we want to know how Random Forest performs in classifying. In our literature review was Fang and Zhan (2015) the only one that used Random Forest and in that research it gave mixed results. We want to know how this classifier performs and if this gives good results in our work.

4.4.1.1 Parameters

There are two main parameters to fine-tune with the Random Forest. The first parameter is the amount of estimators e. Increasing amount of estimators reduces the variance of the model. A low variance allows a good model coverage and very stable solution. A lower amount of estimators has very high variance and therefore a lot of space to increase performance. After stabilisation by increasing estimators, the further increase of estimators only slightly fluctuates the results. Lower variance gives better accuracy than a higher variance (Bonaccorso, 2017). The drawback is that the Increase of the estimators also increases the compute-time.

For our experiments, we use 10 to 300 estimators in steps of 10.

Dangeti (2017) says that the rule of thumb for the amount features per estimator is the square root of the sum of the features. This prevents correlation among the individual estimators. Dangeti (2017) claims that this significantly improves the accuracy of the classifier. In our research, we will be following the rule of thumb of Dangeti (2017).

4.4.2 SVM

SVM works with hyperspace and split the space into regions by putting a hyperplane into to the space, see "**2.2.6 SVM**". This classifier is used in the papers Kouloumpis et al. (2011), Singla et al. (2017), Greaves et al. (2013), Fang and Zhan (2015), Hai Nguyen et al. (2015) and Pang et al. (2002). In these research papers, SVM in the most cases delivered good or the best results. Some of the research used only SVM and no other classifiers.

In the research of Singla et al. (2017) SVM had the best performance with 81.75% and the subject and source that was used is almost the same what we want to target with our research. The difference is that they did the research on Mobile Phones reviews and we will do it on Game reviews.

[6] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[7] http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[8] http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

It seems that the performance of SVM increases if the dataset increases the amount of samples as is shown in the research of Fang and Zhan (2015). Their performance increased from 0.61 to 0.94.

In the research with SVM in the literature reviews, SVM with linear kernel was applied which shows that the data that was used was linear. This concludes that the chance is high that our data is also linear, but to prove it we will run the experiments with the linear and nonlinear kernels (RBF). If one of the labels not predictable with the linear kernel, it means that the data is nonlinear.

4.4.2.1 Fine-tuning

SVM has the flaw that it favours classes with higher frequency. The solution is to alter the C penalty parameter. Increasing the C parameter increases the margin, ignores more noise and generalizes the entire model more. The best is C parameter value is normally 10^{x}, where x is in the collection of $\{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}\}$ (Boschetti & Massaron, 2015, pp. 215-217).

In our experiments we do not fine-tune the C penalty parameter and our experiments use the C parameter as the default 1 (10^{0}).

4.4.3 Naïve Bayes

Naïve Bayes is based on the Bayes' theorem. It predicts by calculating the probability to a label. The algorithm is explained in "**2.2.3 (Multinomial) Naïve Bayes Algorithm**". We will be using Multinomial Naïve Bayes for our research because this algorithm is good in predicting documents with n-grams.

Naïve Bayes is widely used for sentiment analysis and in the research of Pak and Paroubek (2010), Greaves et al. (2013) and Fang and Zhan (2015) Naïve Bayes gave the best results. In Greaves et al. (2013) and Singh et al. (2017) Naïve Bayes was the fastest classifier of all the classifiers that were used in the research. In the research of Joyce and Deng (2017) and Singla et al. (2017) Naïve Bayes was also used to classify positive or negative sentiment. Joyce and Deng (2017) says that Naïve Bayes is good to distinguish between positive and negative. In their research they compare voters from two candidates and Naïve Bayes followed the assumptions that if a tweet is good for one of them, it is bad for the other. Singh et al. (2017) that Naïve Bayes only needs a small dataset to operate effective.

4.4.4 Accuracy

The performance of a classifier is expressed in average of the F-measure of the three folds. This value is also known as the accuracy of the classifier. The following method is used to calculate the accuracy:

$$accuracy = \frac{Fmeasure_{fold1} + Fmeasure_{fold2} + Fmeasure_{fold3}}{3}$$

4.5 Features

Each classifier will be trained with 2000 n-grams features and one lexicon feature. The value of the lexicon feature depends on whether LEX or LEX-NORM is used. Classification without the lexicon will put the value of the lexicon feature at zero for each sample. The reason that the lexicon feature is present in cases that the lexicon is not used, is that we than easily can use the GridSearchCV[9] tool from Scikit-learn. With this tool it is possible to train and predict the classifier with different classifier parameters and input. Each combination of the classifier parameters and input calls a run. Each run will be done using 3 fold cross validation. Afterwards GridSearchCV will return a report with details about accuracy and compute-time of each run.

4.6 Configurations

In our experiments we use multiple classifiers as described in "**4.4 Classifiers**". To train the classifiers we train them with different features sets, a set is built from the following features:

- **N-gram only**
    - Unigram or
    - Bigram or
    - Unigram and bigram
- **N-gram with lexicon feature**
    - **N-gram**
        - Unigram or
        - Bigram or
        - Unigram and bigram
    - **Lexicon feature**
        - LEX or
        - LEX-NORM

The total amount of feature sets will be 9, but for each n-gram we will try different ways to count the presence of a given word or phrase. This is described in the section "**4.3.3.1 Counting word and phrase presents**". Besides the different ways of counting presence of n-grams, we also experiment with different ways to measure the polarity of the lexicon feature, see "**4.3.2 Lexicon**".

This will increase the total amount runs and we will refer to the feature set with the given parameters as a configuration. The total amount of configurations will be 60. Each configuration will be trained by all the classifiers. Each classifier will be setup with a range of parameters, so that we can determine which setup gives the best performance with the incorporation of the used configuration. The classification will be done using 3 fold cross validation to make the results are more reliable. In the Figure 5 Classification of reviews is the process visualized.

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

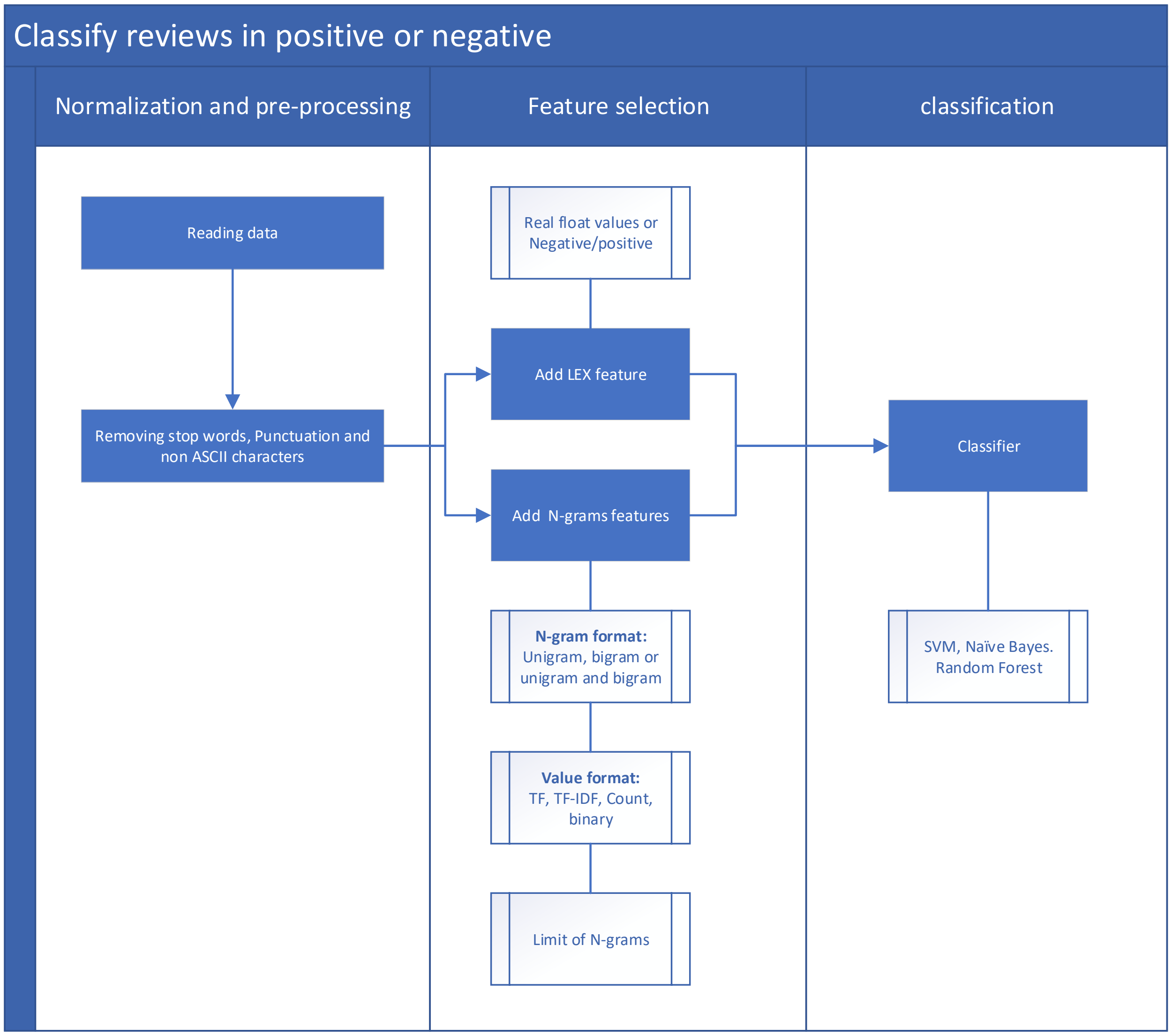
Classify reviews in positive or negative
Normalization and pre-processing
Feature selection
classification
Reading data
Removing stop words, Punctuation and non ASCII characters
Real float values or Negative/positive
Add LEX feature
Add N-grams features
N-gram format: Unigram, bigram or unigram and bigram
Value format: TF, TF-IDF, Count, binary
Limit of N-grams
Classifier
SVM, Naïve Bayes. Random Forest

Figure 5 Classification of reviews

5. Experiments

For our research we analyse the performance of the different n-grams with the different baselines and the lexicon feature. We're following the section "**4.6 Configurations**" for our experiments. Each section of these chapter will highlight and discuss the results of each given n-gram type. The first section will highlight the unigrams, the second bigrams and the last n-grams based on unigrams and bigrams. The features that will be used are described in "**4.5 Features**".

5.1 Unigrams

5.1.1 Random Forest

5.1.1.1 Compute time

In the observations of the train time of the Random Forest algorithm with unigrams, where we find out that the performance is $O = n^2$, where n is the amount of estimators. This following our expectations in the methodology.

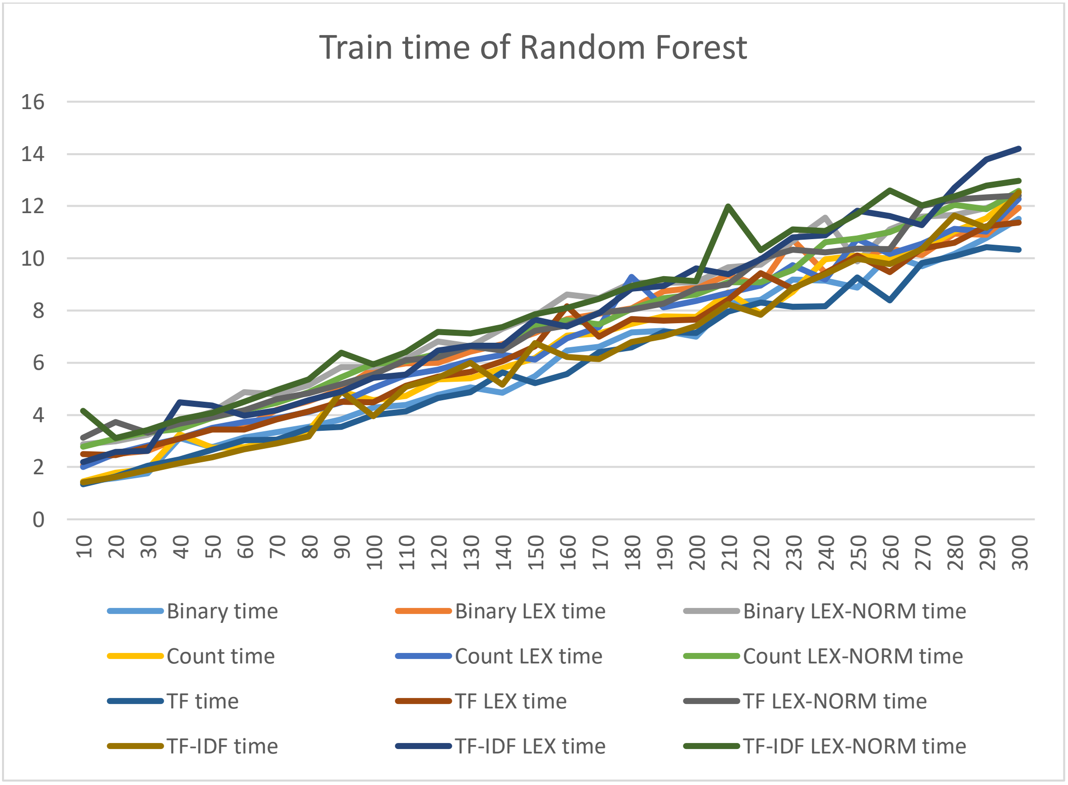

Figure 6 Random Forest compute time increase by different amount of estimators (unigrams)

5.1.1.2 Performance

The performance of every classifier with the same amount of estimators is following the extremely similar pattern. It makes no difference with whichever baseline we use for the unigrams.

From conducting our experiments it is evident that the performance of configurations which use LEX is the best solution as it has a higher accuracy than the LEX-NORM and non-lexicon configuration as shown in Figure 7, Figure 8, Figure 9 and Figure 10. The LEX-NORM has lesser performance than LEX and it has similarities to the research of (Kessler, et al., 2015). The reason is that stronger words lose their value in comparison to words with almost zero value, because in LEX-NORM each word has the value of one. This value is either positive or negative. The performance of the non-lexicon configuration is almost the same as LEX-NORM across all the different baselines. There are some fluctuations in the performance of the configuration, but it depends on the amount of estimators that are used. We need to take in account that Random Forest has some randomness in its self and therefore its internal trees could be flawed or improved that will increase or decrease the performance of the Random Forest. To prevent this behaviour we using 3 fold cross validation as described in “**4.3.1 Training and test dataset**”.

We see in the results the same behaviour as Bonaccorso (2017) if we increase the amount of estimators. After certain of amount of estimators, the results only fluctuated.

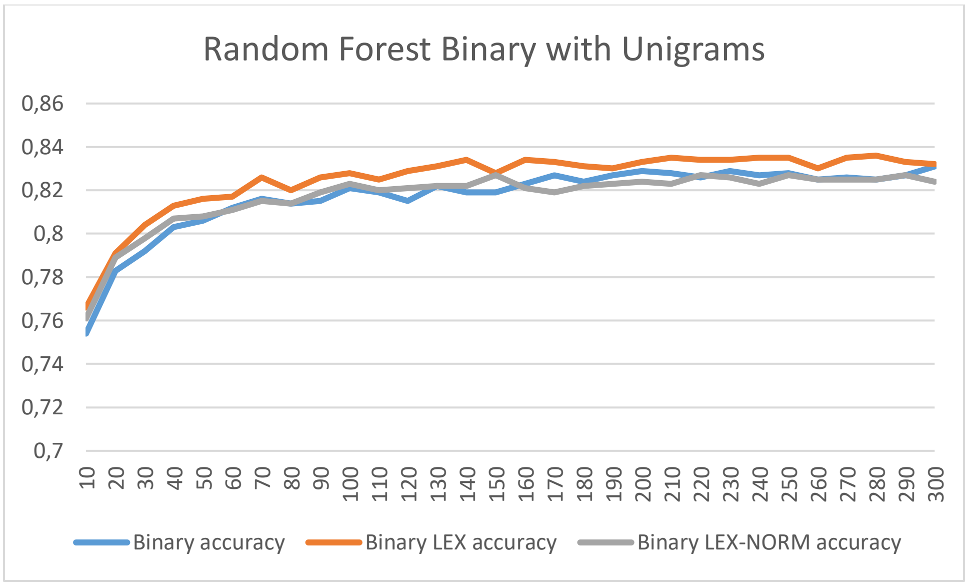

Figure 7 Accuracy of Random forest with Binary unigrams

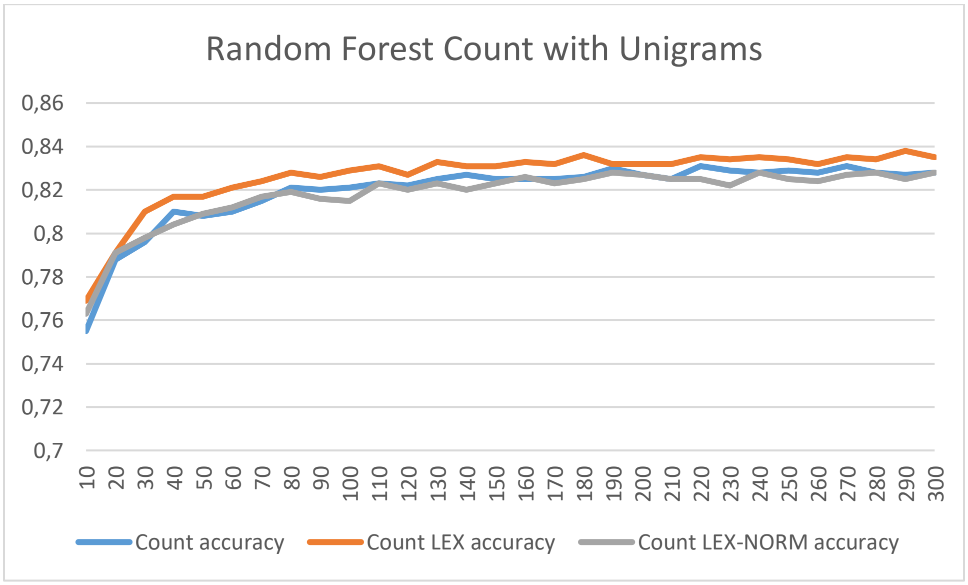

Figure 8 Accuracy of Random forest with Count Unigrams

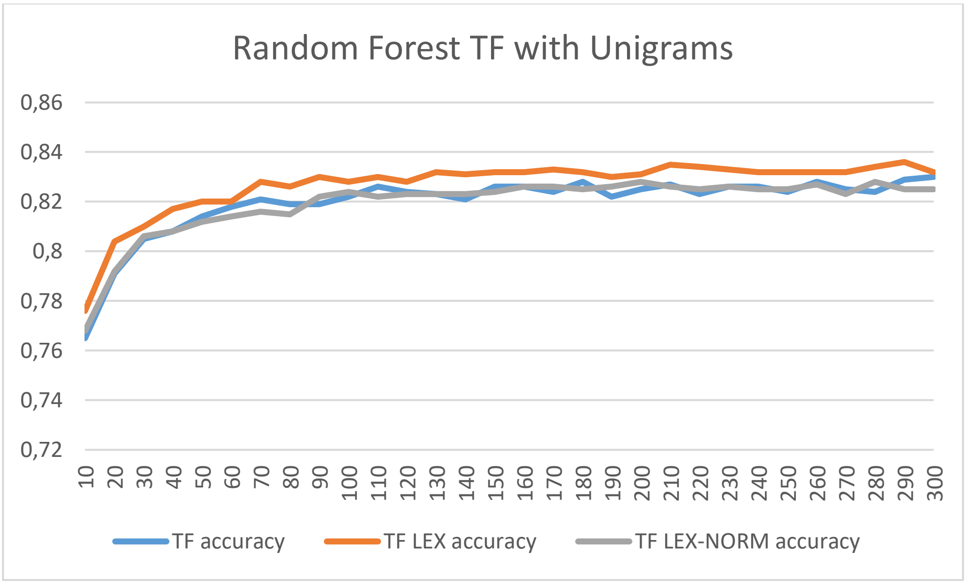

Figure 9 Accuracy of Random forest with TF unigrams

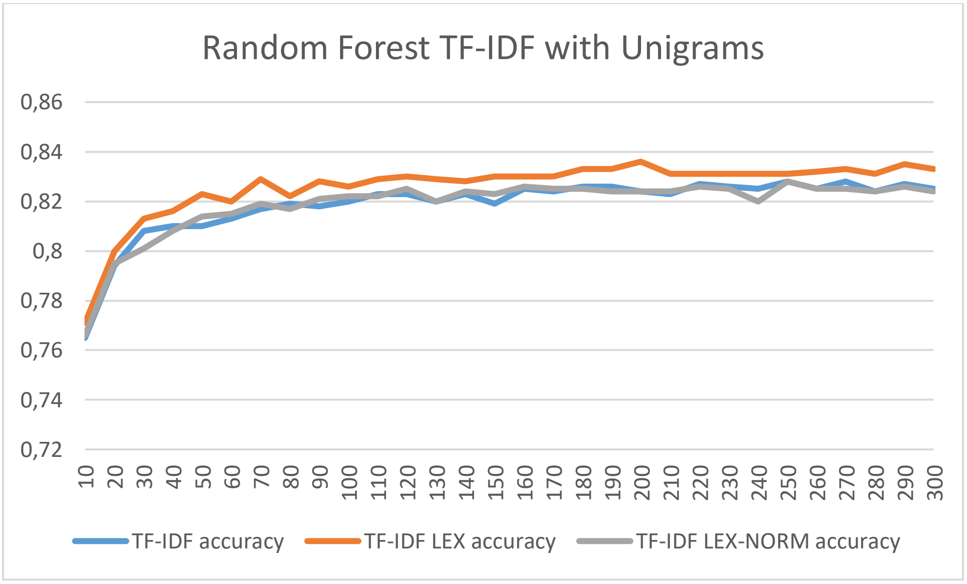

Figure 10 Accuracy of Random forest with TF-IDF unigrams

5.1.1.3 Best performance among the different baselines

As earlier explained gave the results with LEX the best performance across all the baselines. The performance difference between all the baselines is only 1%. The improvement of LEX in comparison to non-lexicon is between 0.6% and 1%. If you consider that the use of lexicon give some overhead in training, predicting and pre-processing, then the question is if the use of LEX and LEX-NORM is still practical.

The use of LEX is dependent on whether the user of these methods prefer shorter compute time or slightly increased accuracy. It could be that other implementations of the lexicon feature has lesser impact on the compute-time, which would make it more logical to use. The Lexicon adds information to the n-grams and this improves the performance of the classifier.

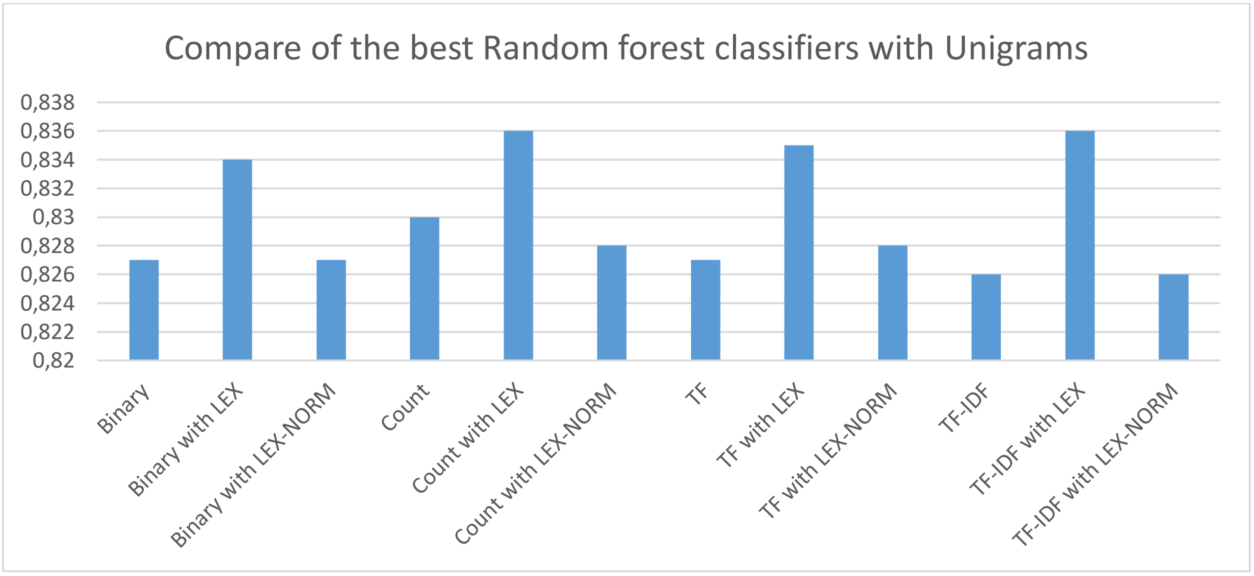

Figure 11 Accuracy of Random forest with unigrams

The best accuracy is 83.6% and is achieved by the Count with LEX baseline at 180 estimators and TF-IDF with LEX baseline at 200 estimators. It makes almost no difference which one is used because the compute time is very similar, so the user free to choose either one of them. If we compare Table 4 Performance of Random Forest with unigrams and Figure 11 Accuracy of Random forest with unigrams, we observe that the TF-IDF non-lexicon with 180 estimators has the worst accuracy of 82.6%.

The results are listed in the Table 4 Performance of Random Forest with unigrams.

Table 4 Performance of Random Forest with unigrams

Name	Estimators	Accuracy	Train time	Predict time
Binary	170	0.827	6.619	0.781
Binary with LEX	140	0.834	6.714	1.409
Binary with LEX-NORM	150	0.827	7.796	1.768
Count	190	0.83	7.781	0.783
Count with LEX	180	0.836	9.298	1.052
Count with LEX-NORM	190	0.828	8.469	1.81
TF	210	0.827	7.953	0.682
TF with LEX	210	0.835	8.406	1.276
TF with LEX-NORM	200	0.828	8.834	1.77
TF-IDF	180	0.826	6.787	0.967
TF-IDF with LEX	200	0.836	9.617	1.116
TF-IDF with LEX-NORM	220	0.826	10.319	2.08

5.1.2 SVM

The conduction results shows that the dataset is linear which was also the case in the reviewed researches about sentimental analysis. The RBF Kernel is more complex than the linear, because the kernel predicts the instances by increasing the hyperspace dimensions where instances are located. This increases the compute time and does not always increase the accuracy of the SVM classifier. This behaviour is viewable in Figure 12 Accuracy of SVM with unigrams and Table 5 Performance of SVM with unigrams.

The prediction time of the linear kernel is 42% to 71% faster than the RBF kernel and the training time of the linear kernel is 9% to 49% faster than the RBF kernel.

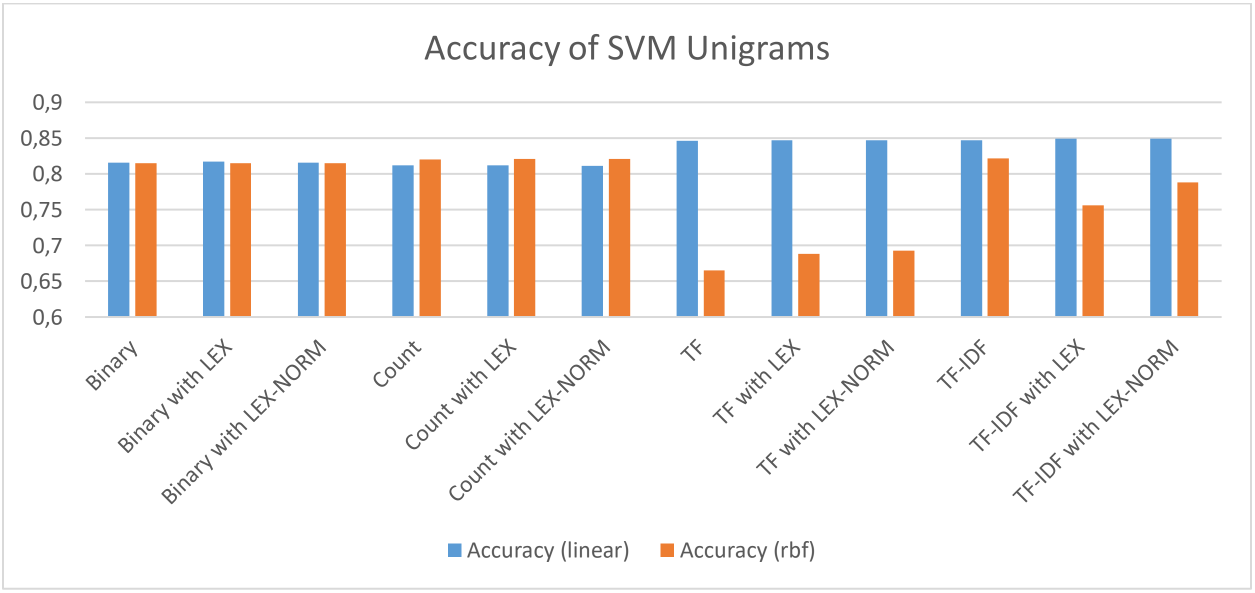

Figure 12 Accuracy of SVM with unigrams

The results show that the accuracy of classifications with the baseline TF or TF-IDF for n-grams and linear kernel scores the best results. The use of lexicon does increase or decrease the results slightly by maximum of 0.2%. Therefore there is no reason to use a lexicon in combination of the SVM classifier. The difference between TF and TF-IDF is only 0.1%.

The classifier almost ignores the lexicon feature, it makes around -0.1% and 0.2% difference with way the value of this feature is calculated compared to the non-lexicon configuration.

The best results of unigrams is 84.7% without a lexicon and the n-gram baseline TF or TF-IDF.

Table 5 Performance of SVM with unigrams

Name	Accuracy (linear)	Accuracy (rbf)	Train time (linear)	Predict time (linear)	Train time % (linear of rbf)	Predict time % (linear of rbf)
Binary	0.816	0.815	13.209	2.47	65%	29%
Binary with LEX	0.817	0.815	13.38	3.722	63%	42%
Binary with LEX-NORM	0.816	0.815	15.076	3.148	73%	32%
Count	0.812	0.82	16.602	2.288	91%	31%
Count with LEX	0.812	0.821	16.144	2.868	82%	36%
Count with LEX-NORM	0,811	0.821	16.487	3.056	83%	36%
TF	0.846	0.665	12.821	4.716	54%	45%
TF with LEX	0.847	0.688	13.732	5.009	55%	46%
TF with LEX-NORM	0.847	0.693	14.087	5.364	55%	48%
TF-IDF	0.847	0.822	11.888	4.574	51%	44%
TF-IDF with LEX	0.849	0.756	13.004	4.89	51%	45%
TF-IDF with LEX-NORM	0.849	0.788	13.376	5.086	53%	46%

5.1.3 Naïve Bayes

Naïve Bayes is our third classifier in the experiments. This classifier treats each feature as independent. This is different than the random forest where each feature has dependency on each other. The results of Figure 13 Accuracy of Naïve Bayes with unigrams and Table 6 Performance of Naïve Bayes with unigrams show that the binary baseline for n-grams gives the best performance.

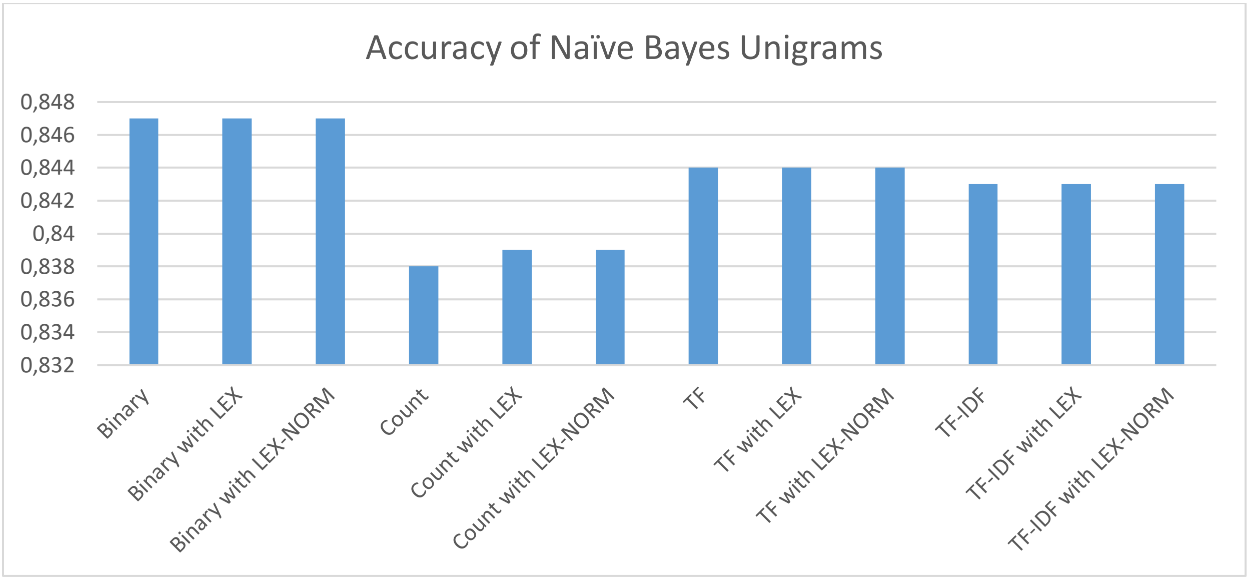

Figure 13 Accuracy of Naïve Bayes with unigrams

Because Naïve Bayes treats every feature individually, it sees the lexicon feature as a single word rather than an overall score of the review. That is the reason that it does not increase the performance of the classifier and is not useful to use for predictions of the reviews.

The behaviour of Naïve Bayes is comparable to the function of the lexicon. Each word in the lexicon has a value of sentiment. The sentiment of the review is the sum of sentiment values of the words. Words with a more extreme sentiment values could dominate the sum of all sentiment values of the other words. Appearance of a word is more important than repeating of a particular word.

For the classification it makes no difference how extreme the sum of the sentiment values is. It is only looking for the positive or negative outcome. If negative the sentiment of the review is negative and else vice versa. The classification of a review can be manipulated if certain kinds of words are used.

This behaviour is also the case in Naïve Bayes, because it predicts the classification based on probability. If certain word is used, it will have higher probability to negative or positive outcome. It is possible that other words can't change this prediction, because their probability to given outcome has a lower influence.

TF and TF-IDF have higher accuracy then Count. The reason is that TF and TF-IDF are normalized and this is visible in the results. We can conclude that normalization improves the performance of the classifier.

The highest accuracy of 84.7% is achieved by n-grams with the binary baseline.

Table 6 Performance of Naïve Bayes with unigrams

Name	Accuracy of Naïve Bayes Unigrams	Train time	Predict time
Binary	0.847	1.016	0.4
Binary with LEX	0.847	1.76	0.767
Binary with LEX-NORM	0.847	2.29	1.252
Count	0.838	1.096	0.429
Count with LEX	0.839	1.797	0.749
Count with LEX-NORM	0.839	2.4	1.124
TF	0.844	0.993	0.456
TF with LEX	0.844	1.933	0.892
TF with LEX-NORM	0.844	2.627	1.195
TF-IDF	0.843	1.156	0.435
TF-IDF with LEX	0.843	1.784	0.781
TF-IDF with LEX-NORM	0.843	2.445	1.14

5.1.4 Comparison between classifiers

In the results of unigrams the use of the lexicon was not impressive. In Random Forest it gave only a performance boost of 1%. This boost almost vanished in SVM, there was maximum performance boost of 0.2% and in some cases the performance decreased by -0.1%. The lexicon gave Naïve Bayes no extra performance and in all the classifiers LEX shown a better performance than LEX-NORM. The reason for this is that Naïve Bayes treats all the features as words and sees the lexicon feature as a word rather than information about the sentiment value of the instance.

All the classifiers had trends in the baselines of the unigrams. Random Forest had the best performance with Count and good results in TF and TF-IDF. SVM did not need a particular baseline but had slightly more performance with TF and TF-IDF. Naïve Bayes was functionally different, it gained more performance with the Binary baseline. In our research we found that for Naïve Bayes, the presence of a word is more important than the number of occurrences of a word.

SVM and Naïve Bayes are similar in accuracy, but the compute-time of SVM is several times larger than that of Naïve Bayes. This makes Naïve Bayes the best classifier of the unigrams. The best results of SVM and Naïve Bayes are gained by the configurations without the lexicon features.

Table 7 Comparison of compute-time and accuracy of the classifiers with unigrams

Classifier	Best Accuracy	Avg. train time	Avg. predict time	Max. train time	Max. predict time
Random Forest	0.836	7,002*	1,288*	14,199*	2,778*
SVM (linear)	0.849	14.151	3.933	16.602	5.364
Naïve Bayes	0.847	1.775	0.802	2.627	1.252

* The time of random forest should be multiple by almost four. The reason is that all the other algorithms are trained and predicted by single CPU thread and in case of random Forest were that four threads.

5.2 Bigram

5.2.1 Random Forest

5.2.1.1 Compute time

The train time has the same behaviour as the unigram, only the offset the graphs is increased by approx. two seconds. The explanation could be that the internal memory copy is bigger because the amount of words is doubled. A unigram only holds one word and bigram holds two words. This increases the amount of characters that are stored into the memory of the machine.

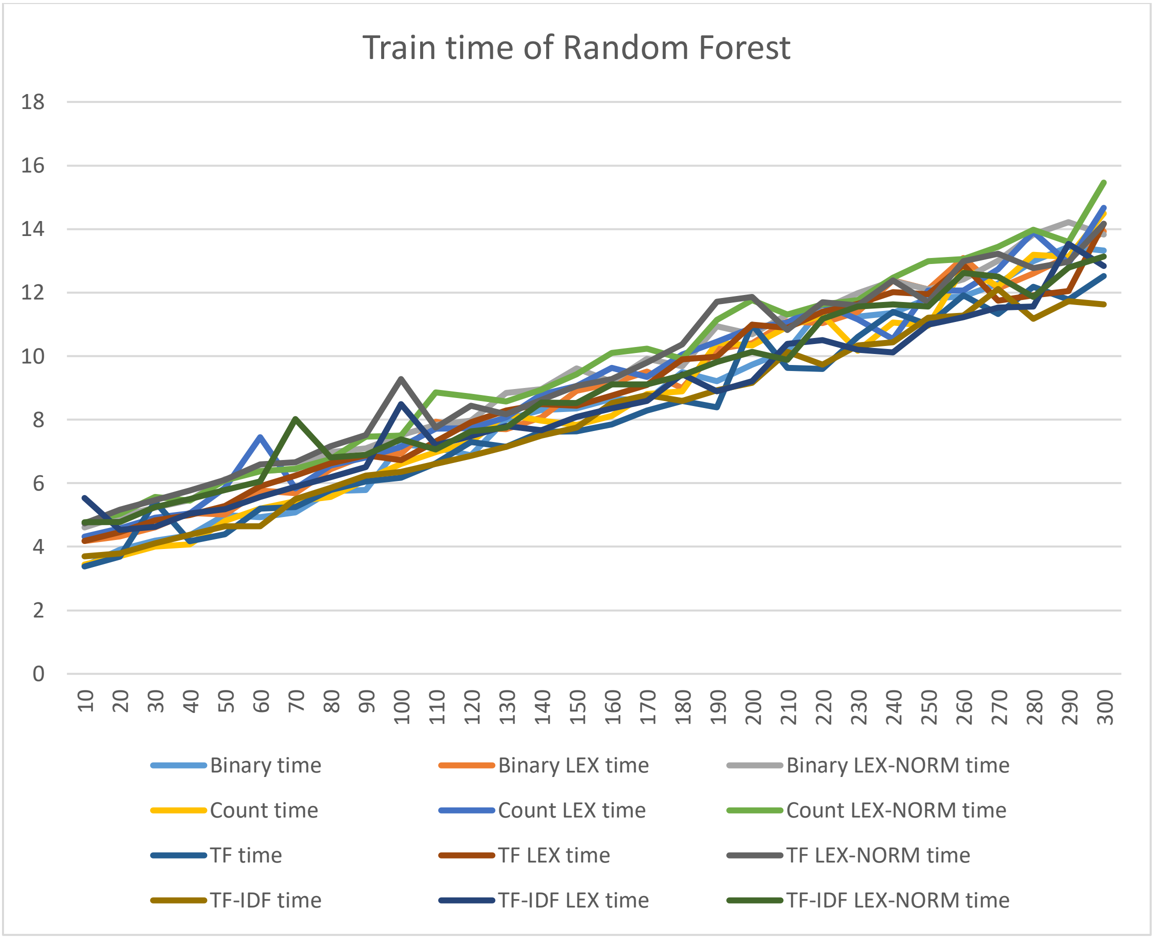

Figure 14 Random Forest compute time increase by different amount of estimators (bigrams)

5.2.1.2 Performance

The diagrams show clear margins between non-lexicon, LEX and LEX-NORM results across all baselines. In every baseline, the diagram shows almost a consistent margin between them as if they are dependent on each other based on the amount of estimators.

The reason could be the way that internal trees of the Random Forest are built for the given amount of estimators. The features and samples are equally distributed over the internal trees. Each random forest get same 2001 features and samples. They are not different among non-lexicon, LEX and LEX-NORM, besides the value of lexicon feature (see "**4.3.3 Lexicon**").

If we compare the pattern of bigrams to unigrams, it is almost the same. The only difference is that the general performance of bigrams are lower. Also the LEX-NORM performance behaviour the same as that from the unigrams. The behaviour by increasing the estimators is good visible and following the same assumptions as by the unigrams.

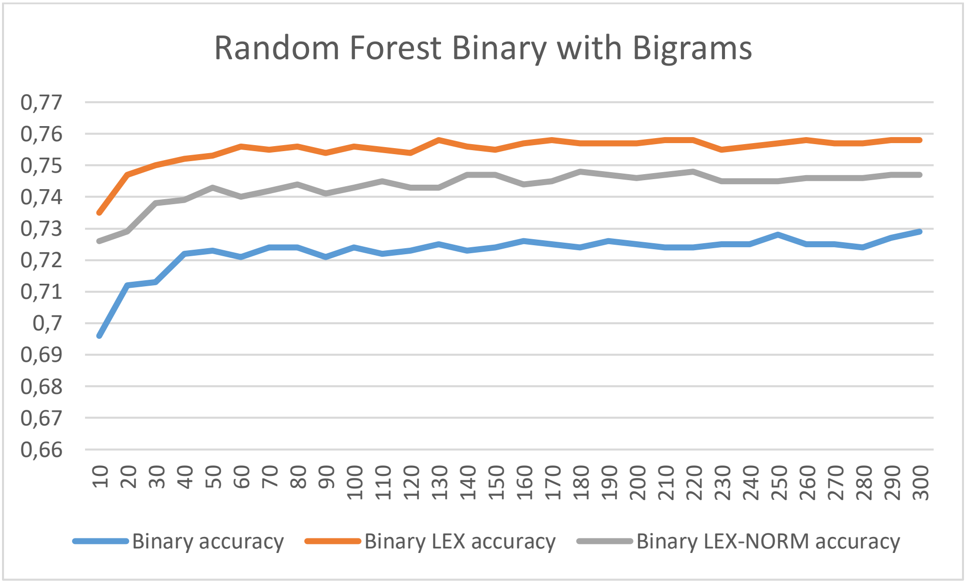

Figure 15 Accuracy of Random forest with Binary bigrams

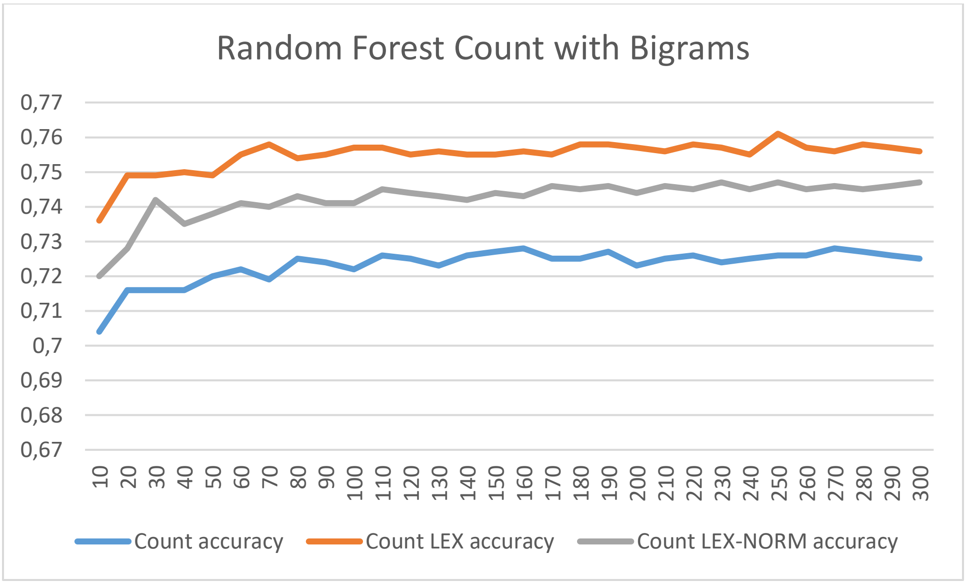

Figure 16 Accuracy of Random forest with Count Bigrams

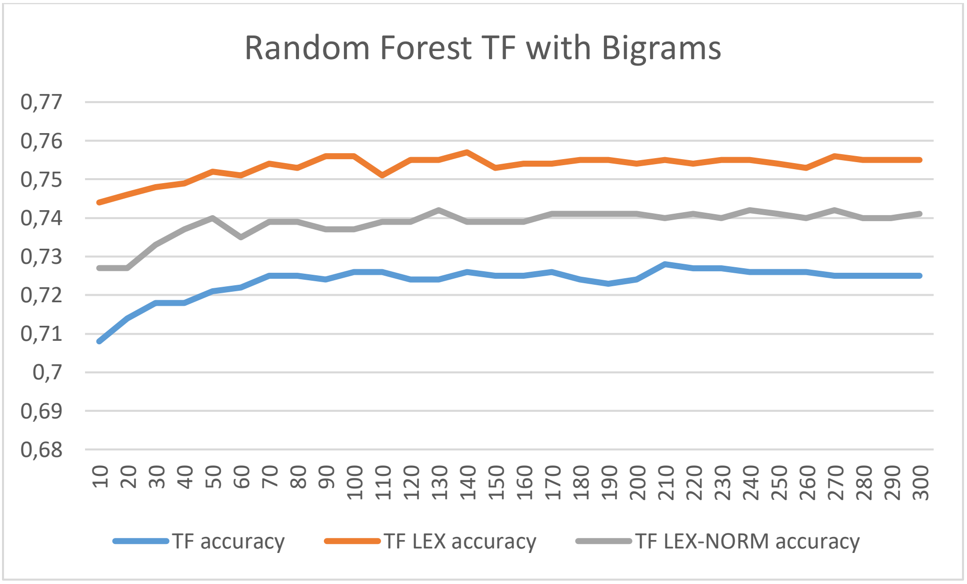

Figure 17 Accuracy of Random forest with TF bigrams

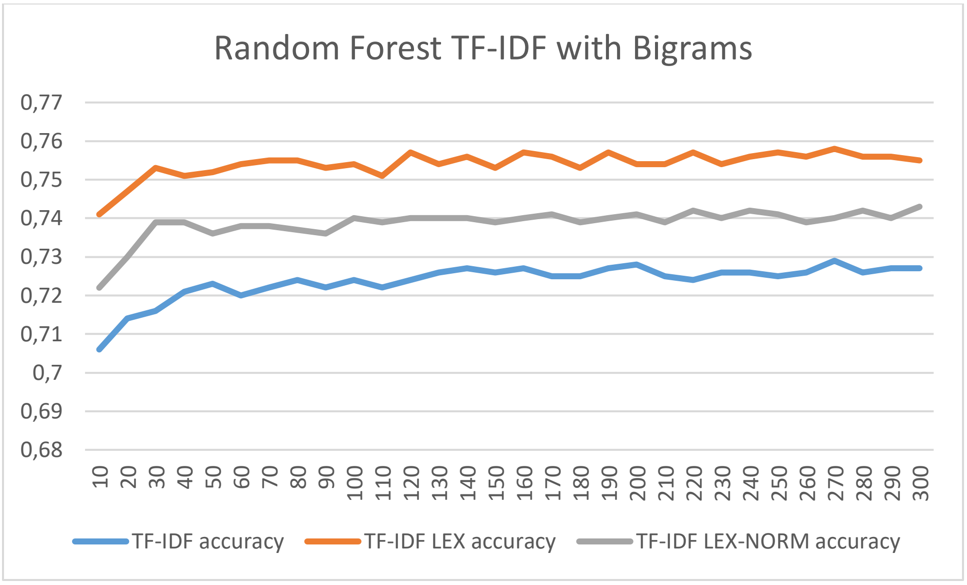

Figure 18 Accuracy of Random forest with TF-IDF bigrams

5.2.1.3 Best performance among the different baselines

The results shows similarity to unigrams with in general lower performance. The LEX-NORM has lesser penalty compared to the unigram with LEX-NORM.

The results show a similar trend to the performance of the unigrams classified by Random Forest. The main differences are that bigrams have a lower performance in general and the accuracy of LEX and LEX-NORM are closer together. LEX still outperforms LEX-NORM in all the cases.

Different between the different baselines is 3.4%. The improvement of LEX against non-lexicon is 2.8% to 3.4%. The use of a lexicon with bigrams is more reasonable than the use of a lexicon with unigrams because the difference between non-lexicon and LEX results are larger and the compute time overhead is lower.

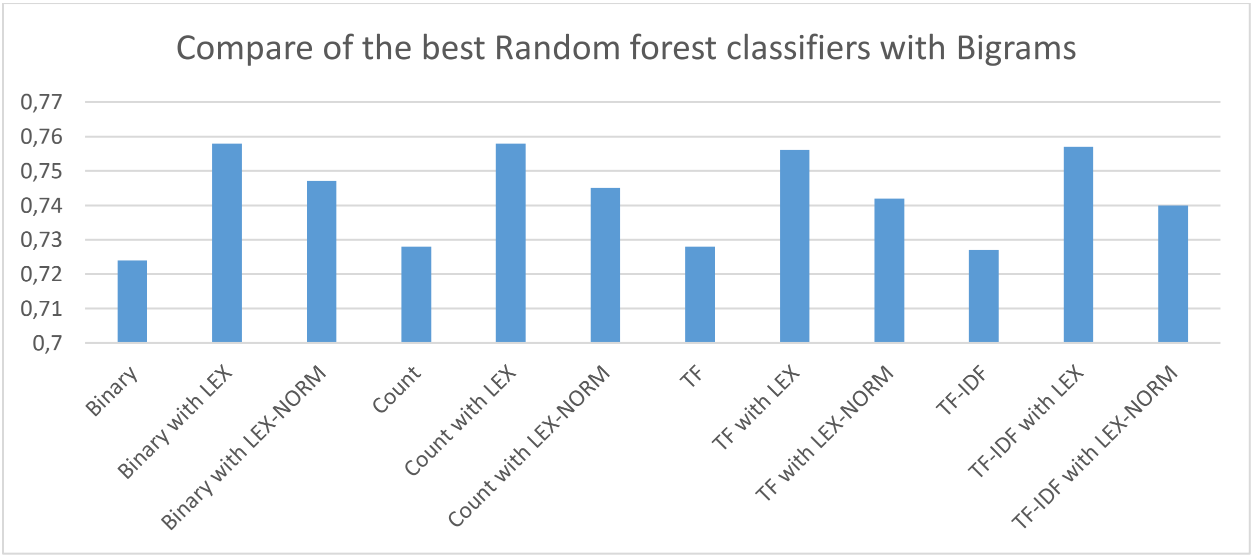

Figure 19 Accuracy of Random forest with bigrams

In the Table 8 Performance of Random Forest with bigrams and Figure 19 Accuracy of Random forest with bigrams, 75.8% is the best accuracy and is achieved by Count with LEX at 160 estimators and Binary with LEX at 130 estimators. The Binary baseline without the lexicon had the worst results, with only 72.4% accuracy. The best performance configurations is 7.8% lower than the best configuration of unigrams with Random Forest. These results are in line with the research of (Pang, et al., 2002; Greaves, et al., 2013). Which baseline is used makes almost no difference to the result, the main difference is in LEX and non-lexicon configurations.

The results are listed in the Table 8 Performance of Random Forest with bigrams.

Table 8 Performance of Random Forest with bigrams

Name	Estimators	Accuracy	Train time	Predict time
Binary	70	0.724	5.084	1.07
Binary with LEX	130	0.758	7.702	1.477
Binary with LEX-NORM	140	0.747	8.965	1.851
Count	160	0.728	8.127	1.16
Count with LEX	70	0.758	5.836	1.395
Count with LEX-NORM	110	0.745	8.864	1.557
TF	160	0.728	7.857	1.121
TF with LEX	100	0.756	6.733	1.416
TF with LEX-NORM	130	0.742	8.175	2.086
TF-IDF	140	0.727	7.494	1.089
TF-IDF with LEX	120	0.757	7.487	1.148
TF-IDF with LEX-NORM	100	0.74	7.379	1.655

5.2.2 SVM

The results shows that the behaviour of linear against non-linear is the same than it was with unigrams. The compute-time of the classification of bigrams for linear and non-linear are closer together, but the accuracy difference between them has increased.

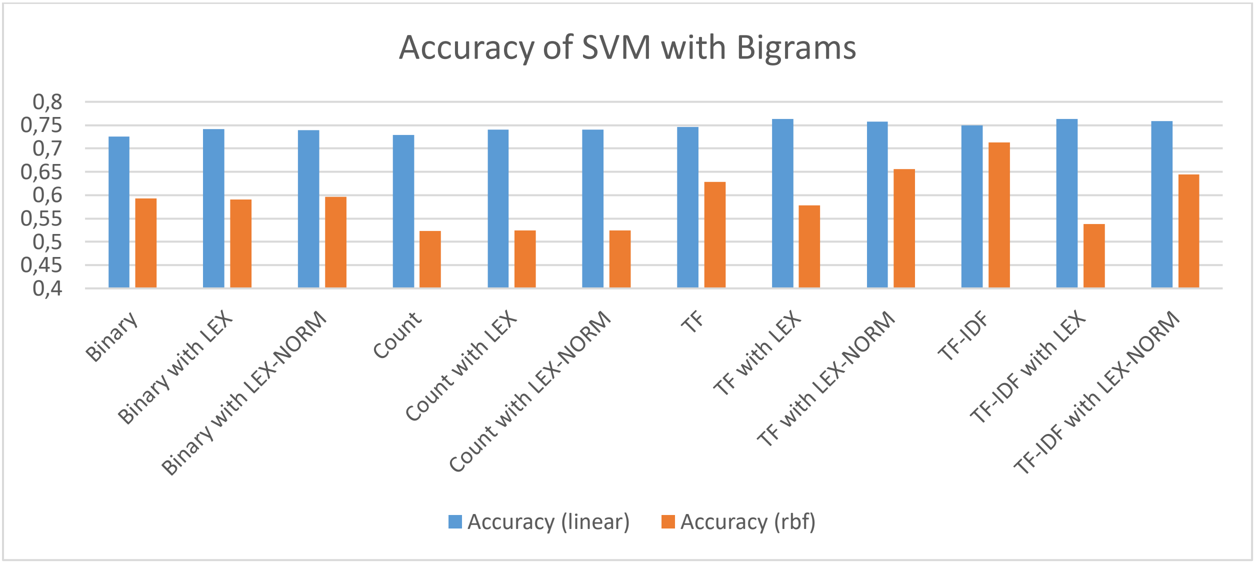

Figure 20 Accuracy of SVM with bigrams

As the earlier observations by the SVM unigram classification shows that the TF and TF-IDF have the best scores on accuracy, this is also the case for classification of bigrams. Using bigrams the use of the lexicon feature does increase the performance of the classifier. That is different than the classification of unigram with SVM.

A possible reason is that when using unigrams all the words are treated as individual once and when using bigrams they are treated as phrases. The value of occurrence of phases is lower than the occurrence of a single word. In bigrams the values are closer together than unigrams, therefore it is more difficult to discriminate the importance of a given occurrence. This not the case if binary is the baseline of the n-grams because the value is either zero or one.

In our experiments we reduce the total amount of n-grams to 2000 and it is possible that important phrases for classification are removed. The reason that it removes important phrases is because it ranks the phrases on occurrence. Only the 2000 most frequently occurring phrases are taken.

If the lexicon is used it adds a sentimental value to the instance that gives more specific information about the sentiment of the review. It recovers the disadvantage of the results of the bigrams slightly, but it still scores lower than the unigrams.

The best results of bigrams is 76.3% with the n-gram baseline TF or TF-IDF and the use of LEX. In their research (Kessler, et al., 2015) the LEX-NORM did decrease the performance in compare to LEX. In our observations was this also the case.

Table 9 Performance of SVM with bigrams

Name	Accuracy (linear)	Accuracy (rbf)	Train time (linear)	Predict time (linear)	Train time % (linear of rbf)	Predict time % (linear of rbf)
Binary	0.725	0.593	7.12	1.151	101%	50%
Binary with LEX	0.741	0.591	7.94	1.643	92%	57%
Binary with LEX-NORM	0.739	0.596	8.773	1.945	90%	62%
Count	0.729	0.523	7.901	1.166	111%	51%
Count with LEX	0.74	0.524	8.952	1.601	105%	57%
Count with LEX-NORM	0.74	0.524	9.281	1.922	102%	61%
TF	0.746	0.628	5.699	1.477	79%	62%
TF with LEX	0.763	0.578	6.577	1.958	77%	66%
TF with LEX-NORM	0.758	0.656	7.211	2.243	79%	72%
TF-IDF	0.749	0.713	5.695	1.468	78%	59%
TF-IDF with LEX	0.763	0.538	6.695	1.865	79%	64%
TF-IDF with LEX-NORM	0,759	0,644	7.296	2.208	80%	71%

5.2.3 Naïve Bayes

The accuracy of Naïve Bayes with bigrams is lower than the results with unigrams. The maximum difference of accuracy between the different baselines for n-grams is 0.5%. The normalization of counting the presence of phrases does increase the performance and this is the same as we saw in the unigrams.

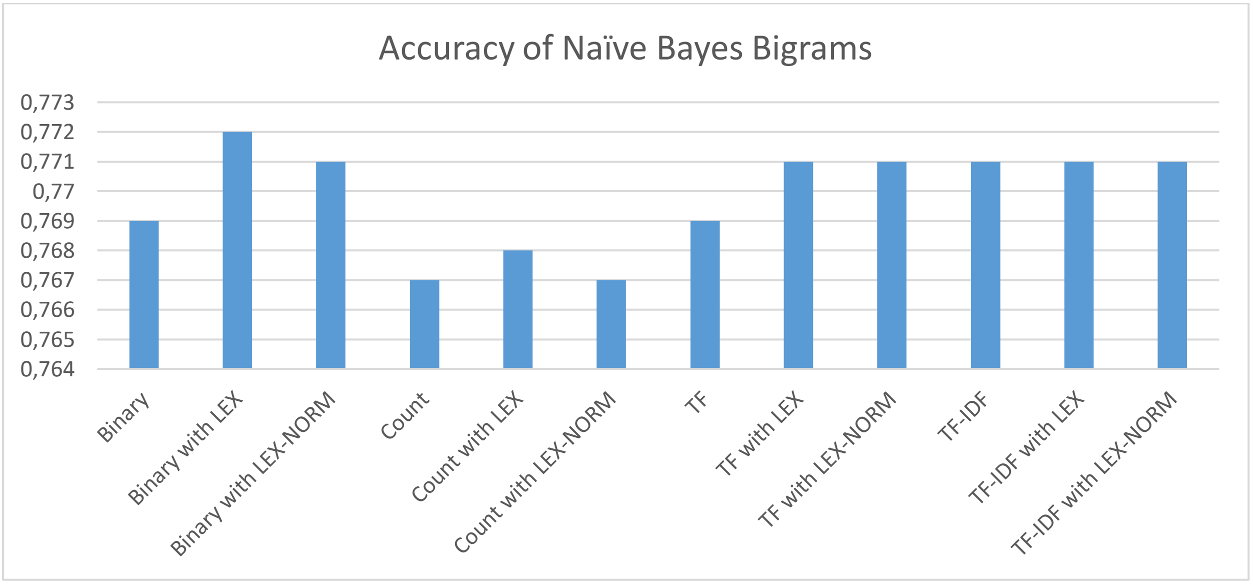

Figure 21 Accuracy of Naïve Bayes with bigrams

The use of a lexicon improves the performance with average of 0.125% across all n-gram baselines. The makes the use of them almost useless and only increase the needed compute-time. The highest accuracy is 77.1% with use of TF-IDF as baseline for n-grams.

Table 10 Performance of Naïve Bayes with bigrams

Name	Accuracy	Train time	Predict time
Binary	0.769	3.199	0.573
Binary with LEX	0.772	3.845	0.847
Binary with LEX-NORM	0.771	4.322	1.199
Count	0.767	3.101	0.558
Count with LEX	0.768	3.903	0.877
Count with LEX-NORM	0.767	4.473	1.207
TF	0.769	3.027	0.597
TF with LEX	0.771	4.083	0.965
TF with LEX-NORM	0.771	4.402	1.223
TF-IDF	0.771	3.224	0.579
TF-IDF with LEX	0.771	3.813	0.987
TF-IDF with LEX-NORM	0.771	4.31	1.243

5.2.4 Comparison between classifiers

The lexicon improves the performance of classification when bigrams are used. With Random Forest it showed an improvement of 2.8% to 3.4%. This improvement is also visible in SVM, but here the average improvement is 1.45%. With Naïve Bayes, the improvement barely visible at only 0.125%. The LEX outperformed the LEX-NORM in all the classifiers.

The results are close together because the n-gram baselines had small role in the classification with bigrams and lexicon feature. Based on the accuracy and compute-time Naïve Bayes with the LEX was the best classifier.

Table 11 Comparison of compute-time and accuracy of the classifiers with bigrams

Classifier	Best Accuracy	Avg. train time	Avg. predict time	Max. train time	Max. predict time
Random Forest	0.761	8.799*	1.534*	15.462*	3.084*
SVM (linear)	0.763	7.428	1.721	9.281	2.243
Naïve Bayes	0.772	3.809	0.905	4.473	1.243

* The time of random forest should be multiple by almost four. The reason is that all the other algorithms are trained and predicted by single CPU thread and in case of random Forest were that four threads.

5.3 Unigrams and bigrams

5.3.1 Random Forest

5.3.1.1 Compute time

The train time is inherited the characteristics of the unigrams and bigrams, because this approve is a combinations of unigrams and bigrams. Our assumption about compute increase still applies through the use of bigrams. The amount of bigrams and unigrams is dependent on external factors. We only know that the total amount of n-grams is 2000. The external factors are the reviews and their content. For our research we use only the 2000 most suitable n-grams, which are chosen by the machine learning system. The result will be varied if other reviews are used, but will still follow the same trend as the unigrams and bigrams do. The reason for varied results is that in other corpuses the distribution between unigrams and bigrams may be different.

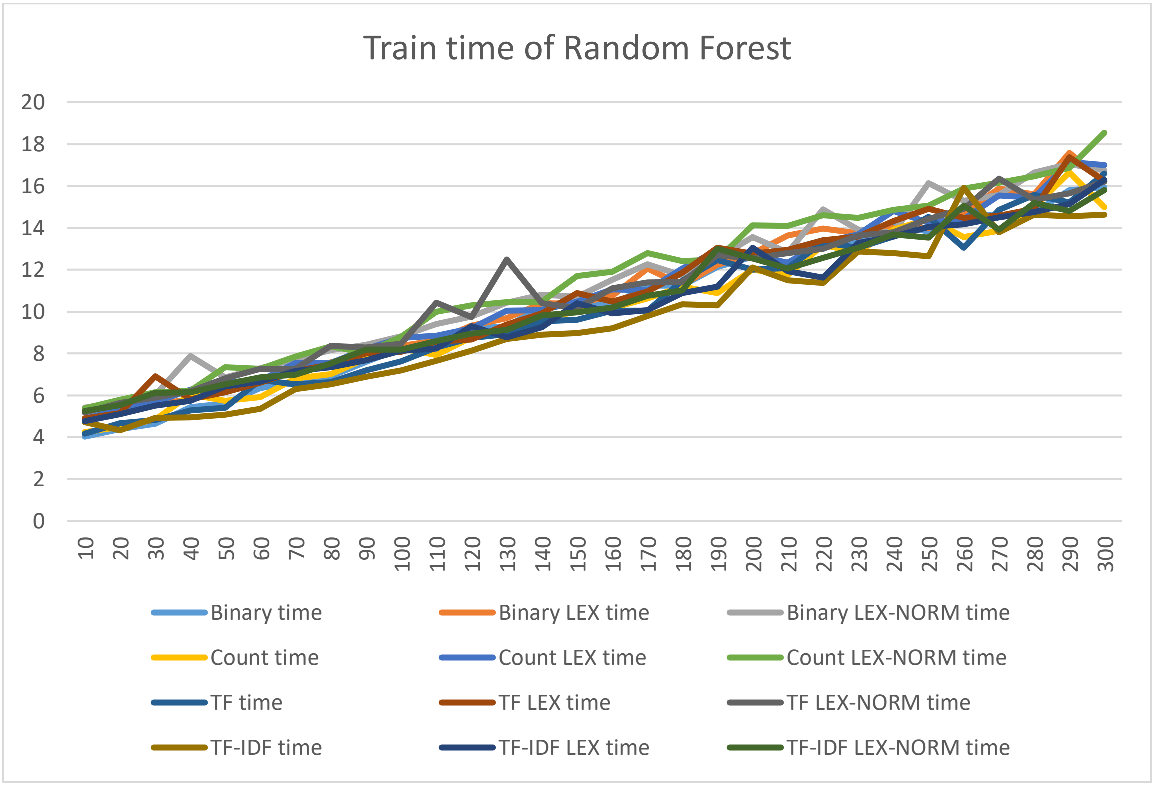

Figure 22 Random Forest compute time increase by different amount of estimators (unigrams, bigrams)

5.3.1.2 Performance

The results showing that this a hybrid between unigrams only and bigrams only. It needs less estimators the produce the same results as unigrams only. We researched that total amount of estimators is reduced by an average of 23.33 in comparison to unigrams only. The compute time is equal to unigrams only with larger amount of estimators, in this area the unigram bigram combination has no improvement.

It never performs better than the unigrams only and it doesn't matter which baseline is used. The non-lexicon and LEX-NORM have almost the same performance. LEX baseline based configurations have slightly better results, which was also the case with unigrams only and bigrams only.

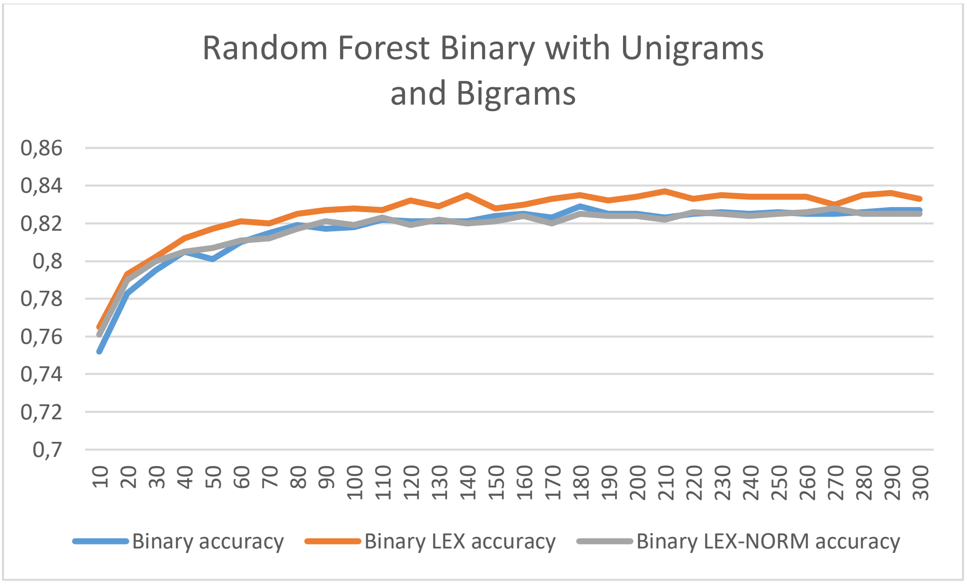

Figure 23 Accuracy of Random forest with unigrams and bigrams (Binary)

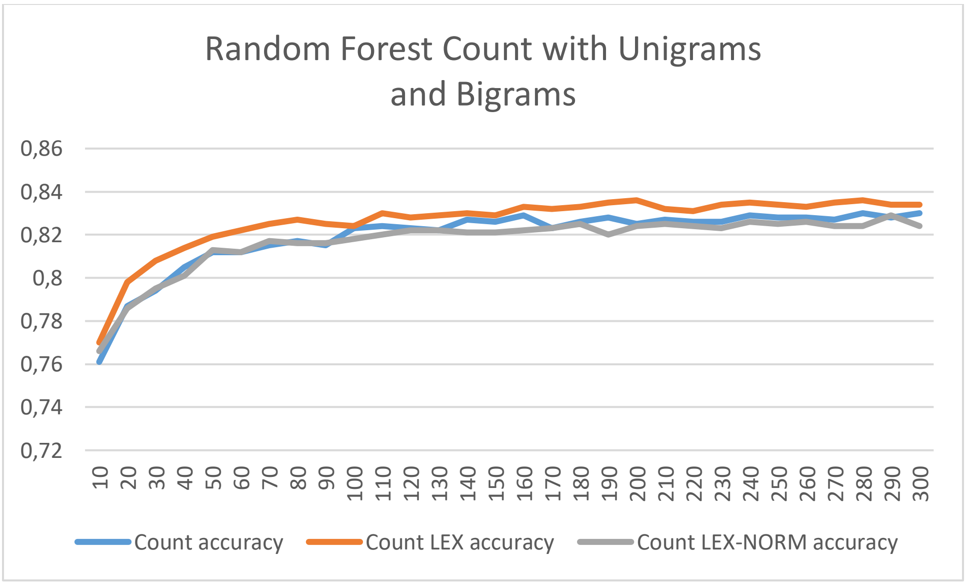

Figure 24 Accuracy of Random forest with unigrams and bigrams (Count)

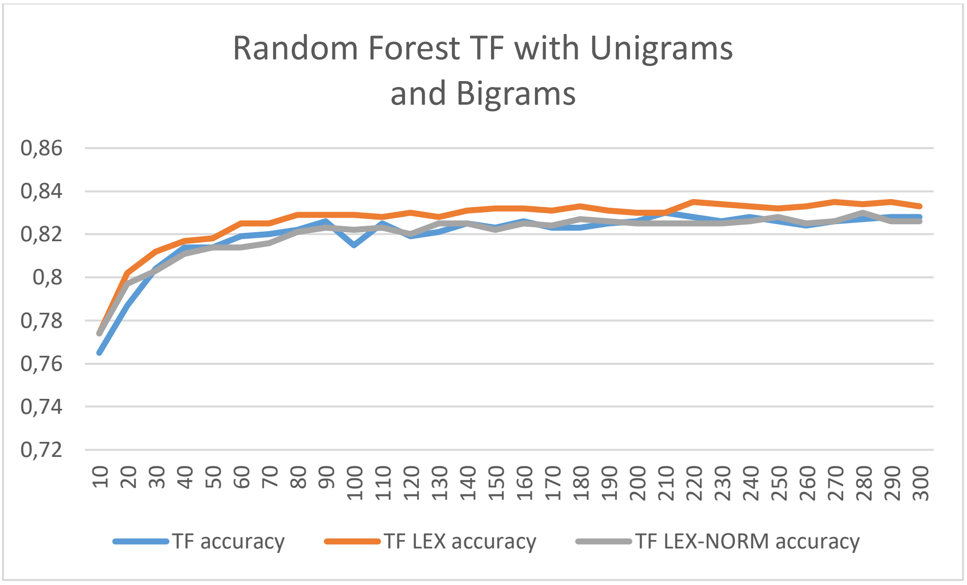

Figure 25 Accuracy of Random forest with unigrams and bigrams (TF)

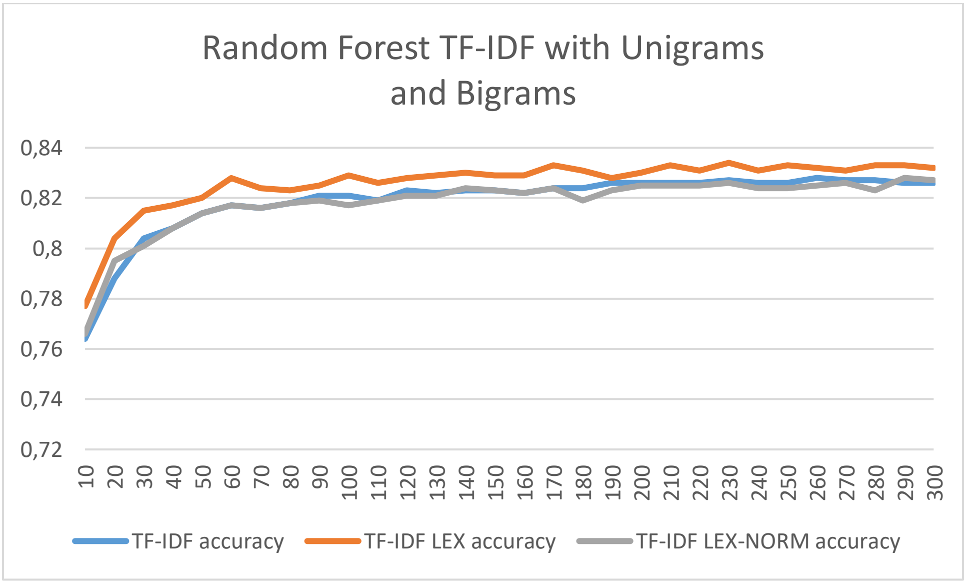

Figure 26 Accuracy of Random forest with unigrams and bigrams (TF-IDF)

5.3.1.3 Best performance among the different baselines

Figure 27 Accuracy of Random forest with unigrams and bigrams shows similarity to unigrams only. As earlier explained the unigram and bigram is a hybrid version from unigram only and bigram only. This is observable in the results of Figure 27 and Table 12.

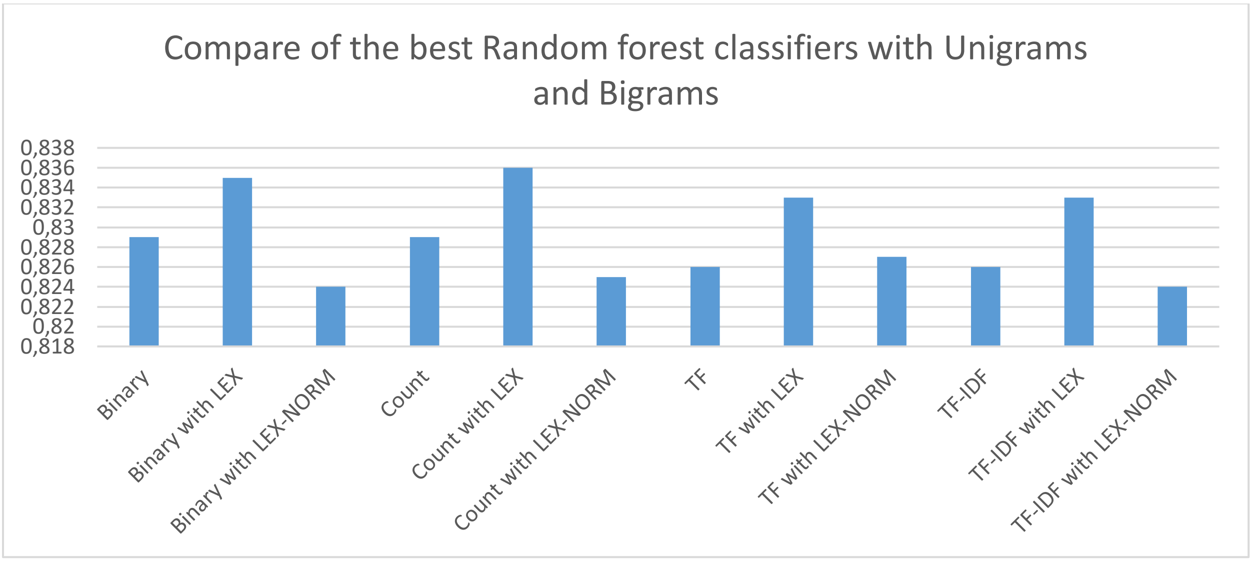

Figure 27 Accuracy of Random forest with unigrams and bigrams

The correlation between Table 4 Performance of Random Forest with unigrams and Table 12 Performance of Random Forest with unigrams and bigrams is very high. This demonstrated in

Table 13 Performance difference between (unigram and bigram) and unigram only. This table shows that the increase and decrease of the accuracy is between -0.3% and 0.2%. As described earlier is the average used amount of estimators reduced by 23.33 in comparison to unigrams only.

We didn’t get performance increase as the research of (Pang, et al., 2002) and in our case the results were similar to unigrams only classified by Random Forest. Therefore there is no reason to apply unigrams and bigrams instead of unigrams only in our case. Beside the missing performance increase, the compute-time did slightly increase.

Table 12 Performance of Random Forest with unigrams and bigrams

Name	Estimators	Accuracy	Train time	Predict time
Binary	180	0.829	11.23	1.047
Binary with LEX	140	0.835	10.408	1.503
Binary with LEX-NORM	160	0.824	11.513	1.905
Count	160	0.829	10.17	1.105
Count with LEX	120	0.836	9.2	1.696
Count with LEX-NORM	180	0.825	11.911	1.988
TF	160	0.826	10.052	1.252
TF with LEX	180	0.833	11.847	1.538
TF with LEX-NORM	180	0.827	11.418	2.042
TF-IDF	190	0.826	10.301	1.914
TF-IDF with LEX	170	0.833	10.059	2.017
TF-IDF with LEX-NORM	140	0.824	9.822	2.115

Table 13 Performance difference between (unigram and bigram) and unigram only

Name	Estimators	Accuracy	Increase/decrease number of estimators	Increase/decrease of accuracy
Binary	180	0.829	+ 10	+ 0.02
Binary with LEX	140	0.835	+ 0	+ 0.01
Binary with LEX-NORM	160	0.824	+ 10	- 0.03
Count	160	0.829	- 30	- 0.01
Count with LEX	120	0.836	- 60	- 0.00
Count with LEX-NORM	180	0.825	- 10	- 0.03
TF	160	0.826	- 50	- 0.01
TF with LEX	180	0.833	- 30	- 0.02
TF with LEX-NORM	180	0.827	- 20	- 0.01
TF-IDF	190	0.826	- 10	- 0.00
TF-IDF with LEX	170	0.833	- 30	- 0.03
TF-IDF with LEX-NORM	140	0.824	- 80	- 0.02

5.3.2 SVM

The classifications of unigrams and bigrams have lower accuracy than the unigrams only, but the behaviour of the two are the same. TF and TF-IDF has the best scores with accuracy of 84.7% and 84.6%.

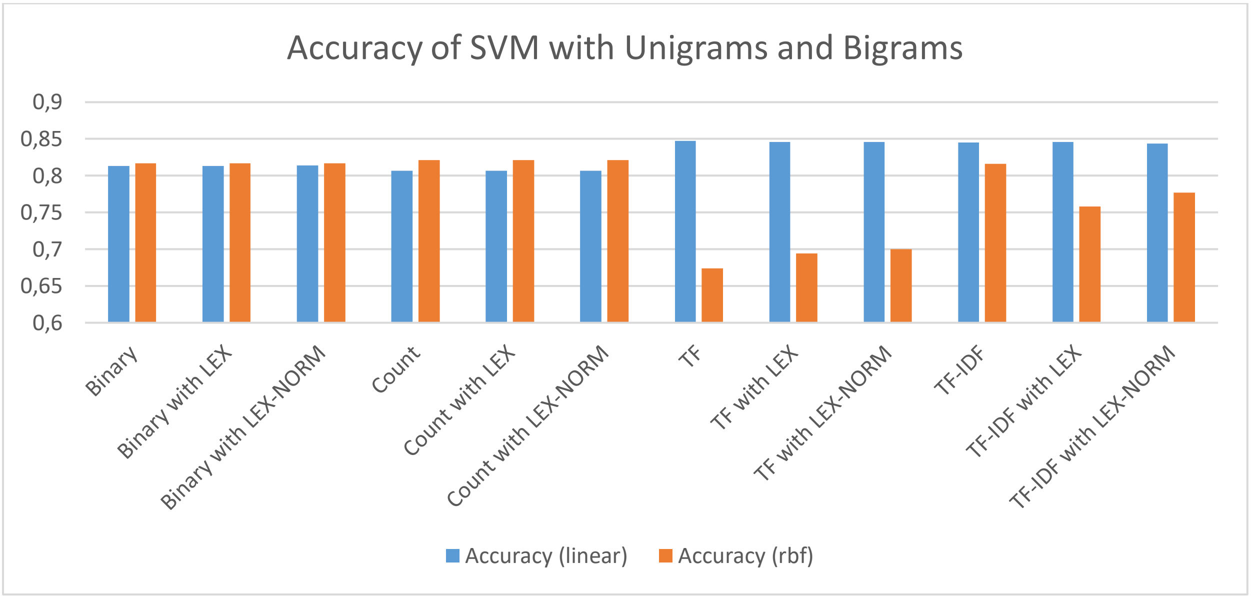

Figure 28 Performance of SVM with unigrams and bigrams

There is no reason to use this way of classification, because all results are lower than the unigrams only. The results of unigrams and bigrams are listed in **Fout! Ongeldige bladwijzerverwijzing.**.

Table 14 Performance of SVM with unigrams and bigrams

Name	Accuracy (linear)	Accuracy (rbf)	Train time (linear)	Predict time (linear)	Train time % (linear of rbf)	Predict time % (linear of rbf)
Binary	0.813	0.817	15.853	2.802	68%	29%
Binary with LEX	0.813	0.817	16.557	3.374	68%	33%
Binary with LEX-NORM	0.814	0.817	17.615	3.543	70%	32%
Count	0.807	0.821	18.411	2.71	87%	33%
Count with LEX	0.807	0.821	21.594	3.133	97%	35%
Count with LEX-NORM	0.807	0.821	20.493	3.467	89%	36%
TF	0.847	0.674	15.358	5.092	58%	48%
TF with LEX	0.846	0.694	15.873	5.483	58%	46%
TF with LEX-NORM	0.846	0.7	16.602	5.654	60%	47%
TF-IDF	0.845	0.816	15.207	4.888	56%	45%
TF-IDF with LEX	0.846	0.758	15.855	5.239	56%	45%
TF-IDF with LEX-NORM	0.844	0.777	16.31	5.51	57%	47%

5.3.3 Naïve Bayes

The accuracy of every baseline is comparable to the results of unigrams only. The overall accuracy is 0.4% lower than that of unigrams only and the compute-time for training unigrams and bigrams increases by an average of 2.6 times. The best results are given by n-grams with binary baseline. Their accuracy was 84.6%. If we compare this results to the unigrams only, there is no reason to use these kind of configurations for predicting sentiment of reviews.

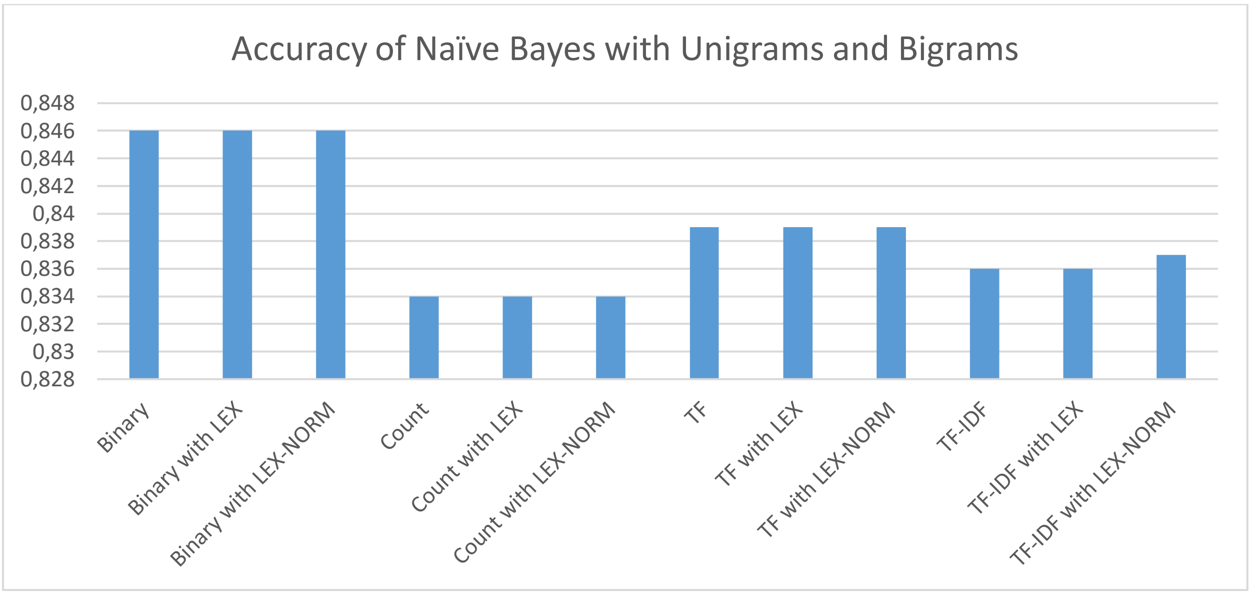

Figure 29 Accuracy of Naïve Bayes with unigrams and bigrams

Table 15 Performance of Naïve Bayes with unigrams and bigrams

Name	Accuracy	Train time	Predict time
Binary	0.846	3.577	0.672
Binary with LEX	0.846	4.391	1.085
Binary with LEX-NORM	0.846	4.832	1.294
Count	0.834	3.686	0.663
Count with LEX	0.834	4.493	1.107
Count with LEX-NORM	0.834	4.905	1.351
TF	0.839	3.955	0.677
TF with LEX	0.839	4.516	1.066
TF with LEX-NORM	0.839	4.961	1.329
TF-IDF	0.836	3.755	0.714
TF-IDF with LEX	0.836	4.537	1.033
TF-IDF with LEX-NORM	0.837	4.905	1.308

5.3.4 Comparison between classifiers

All the results are familiar to unigrams only and bigrams only. The results shown that this is a hybrid between unigrams only and bigrams only, this is because it has inherited the properties of both. The best results based on accuracy and compute-time was scored by the Naïve Bayes.

Table 16 Comparison of compute-time and accuracy of the classifiers with unigrams and bigrams

Classifier	Best Accuracy	Avg. train time	Avg. predict time	Max. train time	Max. predict time
Random Forest	0.837	10.527*	1.651*	4.622*	0.837*
SVM (linear)	0.847	17.144	4.241	21.594	5.654
Naïve Bayes	0.846	4.376	1.025	4.961	1.351

* The time of random forest should be multiple by almost four. The reason is that all the other algorithms are trained and predicted by single CPU thread and in case of random Forest were that four threads.

5.4 Comparison between the different n-gram types

Naïve Bayes outperforms all the other classifier and its performance is highest with a Binary baseline for n-grams. From our observations, the hybrid of unigrams and bigrams is capped to the maximum performance of the unigrams only. The Table 17 Comparison of the most accurate classifiers from the different n-grams shows that the unigrams has the best performance.

When the n-grams contains bigrams then the lexicon starts to be useful, but the maximum improvement is achieved by Random Forest with bigrams with an improvement of 3.4%. In most cases was this much lower and sometimes it gave decrement in performance.

The use of bigrams increases the compute-time of the classifiers dramatically.

Table 17 Comparison of the most accurate classifiers from the different n-grams

N-gram	Best Classifier	Best type	Best accuracy	Worst Classifier*	Worst accuracy*
Unigram	Naïve Bayes	Binary (None)	0.847	Random Forest (10)	0.754
Bigram	Naïve Bayes	Binary (LEX)	0.772	Random Forest (10)	0.696
Unigram and Bigram	Naïve Bayes	Binary (None, LEX, LEX-NORM)	0.846	Random Forest (10)	0.752

* SVM (rbf) is in all cases outperformed by SVM (linear) on speed and accuracy that is the reason that it is not taken in the above table for comparison.

6. Conclusions

Our research shows that using bigrams in n-grams reduce the accuracy for classifying reviews on positive or negative sentiment. The best scores were scored by unigrams. The behaviour that Pak and Paroubek (2010) described about bigrams was not visible in our results, the highest accuracy with bigrams was only 0.772. If the bigrams were involved in the n-grams it always limit the performance of the classifier. Bigrams did not improve performance so it didn't help disambiguation of words, which was described by Pedersen (2001). Pang et al. (2002) wrote that bigrams had lower accuracy then unigrams and the combination of unigrams and bigrams had the highest accuracy. The first part of this statement was also apparent in our research, but the combination had a worse accuracy than the unigrams only. Based on our results, we agree with the statement of Pang et al. (2002) concerning Naïve Bayes and bigrams. His statement was that the use of bigrams in combination of Naïve Bayes goes against the Naïve Bayes principals.

Regardless of the accuracy of bigrams, the compute-time increases several times making bigrams an even worse choice to consider for sentiment analysis. We believe that the reason could lay in the extra memory use to store all the information.

Changing the baseline of an n-gram influence the outcome of the classification. Random Forest and SVM scores the best accuracy on Count, TF and TF-IDF. The normalization by TF and TF-IDF helps to increase performance and advisable for other researches concerning sentiment analysis. Naïve Bayes had the worst scores with non-normalization.

Naïve Bayes scored the best accuracy out of all the classifiers with Binary n-gram baseline. Our theory is that the presence of a word is more important than amount of times it appears. This behaviour is comparable to using a sentiment lexicon where each word has a value. This means that an appearance of a given word can alter the entire outcome.

Our conducted experiments shows similarities to the research of Kessler et al. (2015), where the normalization of a lexicon decreased the performance in comparison to the non-normalized lexicon. We find the use of the lexicon questionable, because the performance boost is small and it does not increase the accuracy of our best classifier.

Random Forest had in research of Fang and Zhan (2015) mixture results, this results were also observed in our research with 83.6% as the highest accuracy. SVM was very competitive to the Naïve Bayes, but the Naïve Bayes had better performance with 84.7% accuracy. The best SVM accuracy was 84.9%, but its compute-time took several time longer than of Naïve Bayes. SVM proved that the dataset of the Amazon game reviews is linear. In our research we did not modify the C penalty parameter of SVM, doing this could have produced better results than Naïve Bayes and justify the extra compute-time.

6.1 Reflection

The relation between our work and that from others is difficult because the sources and goals are different. Also that diagrams can give a false impression of results which can be caused by scales, labels and size. This can make it difficult to understand the context of the diagram.

Compute power plays big role in machine learning and can limit the scope of the experiments. We experienced this when trying to predict all the data. We planned to perform 10 fold cross validation, instead we had to settle for 3 fold cross validation. If we had more compute power we could also have taken the C penalty parameter of SVM in account.

6.2 Future works

In the future we could apply negation to verbs and adjectives as the research of Fang and Zhan (2015) did or we could deploy this in another way. We believe that if we would do it, the classifier needs to understand the context of the text. As part of understanding the context, we could implement methods to detect sarcasm and unrelated parts to improve the classification accuracy. The understanding of context could be next step in sentiment analysis to improve the predictions and its accuracy.

7. References

Bonaccorso, G., 2017. Naive Bayes. In: *Machine Learning Algorithms.* Birmingham: Packt Publishing Limited, pp. 120 - 123.

Bonaccorso, G., 2017. Random Forest. In: *Machine Learning Algorithms.* Bimingham: Pack Publishing ltd., pp. 167-169.

Bonaccorso, G., 2017. ROC Curve. In: *Machine Learning Algorithms.* Birmingham: Packt Publishing Ltd., pp. 115-118.

Boschetti, A. & Massaron, L., 2015. *Python Data Science Essentials.* Second Edition ed. Birmingham: Packt Publishing Ltd..

Breiman, L., 1996. Bagging Predictors. *Machine Learning,* 24(2), p. 123–140.

Cang, S., 2011. *A mutual information based feature selection algorithm.* Shanghai, IEEE.

Dangeti, P., 2017. AdaBoost classifier. In: *Statistics for Machine Learning.* Birmingham: Packt Publishing Ltd., pp. 158-159.

Dangeti, P., 2017. Random forest classifier. In: *Statistics for Machine Learning.* Birmingham: Packt Publishing Ltd., p. 150.

Dangeti, P., 2017. Support Vector Machines and Neural Networks. In: *Statistics for Machine Learning.* Birmingham: Packt Publishing Ltd., pp. 220-227.

Fang, X. & Zhan, J., 2015. Sentiment analysis using product review data. *Journal of Big Data,* 2(5), pp. 1-14.

Greaves, F. et al., 2013. Use of sentiment analysis for capturing patient experience from free-text comments posted online. *Journal of medical Internet research,* 15(11).

Hai Nguyen, T., Shirai, K. & Velcin, J., 2015. Sentiment analysis on social media for stock movement prediction. *Expert Systems with Applications,* 42(24), pp. 9603-9611.

Joshi, P., 2017. What are Decision Trees?. In: *Artificial Intelligence with Python.* Birmingham: Packt Publishing Ltd., pp. 64-65.

Joyce, B. & Deng, J., 2017. *Sentiment analysis of tweets for the 2016 US presidential election.* Cambridge, IEEE, pp. 1-4.

Kessler, W., Klinger, R. & Kuhn, J., 2015. *Towards Opinion Mining from Reviews for the Prediction of Product Rankings.* Lisbon, The Association for Computational Linguistics, pp. 51-57.

Kessler, W. & Kuhn, J., 2013. *Detection of product comparisons - How far does an out-of-the-box semantic role labeling system take you?.* Stuttgart, Universität Stuttgart, pp. 1892-1897.

Klinger, R. & Cimiano, P., 2013. *Bidirecional inter-dependencies of subjective expressions and targets and their value for a joint model.* s.l., ACL, pp. 848-854.

Knispelis, A., 2016. *LDA Topic Models.* [Online]
Available at: https://www.youtube.com/watch?v=3mHy4OSyRf0
[Accessed 26 April 2018].

Kouloumpis, E., Wilson, T. & Moore, J., 2011. *Twitter Sentiment Analysis: The Good the Bad and the OMG!.* Barcelona, AAAI Press, pp. 538-541.

Lei, S., 2012. *A Feature Selection Method Based on Information Gain and Genetic Algorithm.* Hangzhou, IEEE, p. 356.

Lin, C., He, Y., Everson, R. & Ruger, S., 2012. Weakly Supervised Joint Sentiment-Topic Detection from Text. *IEEE Transactions on Knowledge and Data Engineering,* 24(6), pp. 1135-1137.

Manning, C. D., Raghavan, P. & Schütze, H., 2008. *Introduction to Information Retrieval.* s.l.:Cambridge University Press.

Pak, A. & Paroubek, P., 2010. *Twitter as a Corpus for Sentiment Analysis and Opinion Mining.* Valletta, European Language Resources Association, pp. 1320-1326.

Pang, B., Lillian, L. & Vaithyanathan, S., 2002. *Thumbs up?: sentiment classification using machine learning techniques.* Philadelphia, Association for Computational Linguistics, pp. 79-86.

Pedersen, T., 2001. *A decision tree of bigrams is an accurate predictor of word sense.* Pittsburgh, Association for Computational Linguistics, pp. 1-8.

Pedregosa, F. et al., 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research,* Issue 12, pp. 2825-2830.

Platt, J. C., 1998. *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines,* s.l.: Microsoft Research.

Richardson, L., 2017. *Beautiful Soup.* [Online]
Available at: https://www.crummy.com/software/BeautifulSoup/
[Accessed 24 April 2018].

Schapire, R. E. & Singer, Y., 2000. BoosTexter: A Boosting-based System for Text Categorization. *Machine Learning,* 39(2-3), pp. 135-168.

Singh, J., Signh, G. & Singh, R., 2017. Optimization of sentiment analysis using machine learning classifiers. *Human-centric Computing and Information Sciences,* 7(1), pp. 1-12.

Singla, Z., Randhawa, S. & Jain, S., 2017. *Sentiment Analysis of Customer Product Reviews Using Machine Learning.* Coimbatore, IEEE, pp. 1-5.

Stone, P. J., Dunphy, D. C., Smith, M. S. & Ogilvie, D. M., 1996. *General Inquier: A Computer Approach to Content Analysis.* Oxford: M.I.T. Press.

Wasden, L., 2009. *Internet Lingo Dictionary: A Parent's Guide to Codes Used in Chat Rooms, Instant Messaging, Text Messaging, and Blogs,* Idaho: Idaho Commission for Libraries.

Whitelaw, C., Garg, N. & Argamon, S., 2005. *Using appraisal groups for sentiment analysis.* s.l., ACM, pp. 625-631.

Wilson, T., Wiebe, J. & Hoffmann, P., 2005. *Recognizing contextual polarity in phraselevel sentiment analysis.* Vancouver, Association for Computational Linguistics, pp. 347-354.

Wilson, T., Wiebe, J. & Hoffmann, P., 2009. Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis. *Computational Linguistics,* 35(3), pp. 399-433.

8. Appendix

8.1 Source Code

This file contains our source code, but it can also be viewed on the Git-hub page.

coursework2.py

https://github.com/andrevdl/intelligent-systems-EHU/blob/master/intelligent%20Systems%20EHU/coursework2.py

8.2 Results

This file is an Excel file containing the experiments' results.

results.xlsx