UMEÅ UNIVERSITY

EVALUATION OF MACHINE LEARNING ALGORITHMS FOR SMS SPAM FILTERING

David Bäckman

Bachelor Thesis, 15 credits
Bachelor Of Science Programme in Computing Science

2019

Abstract

The purpose of this thesis is to evaluate different machine learning algorithms and methods for text representation in order to determine what is best suited to use to distinguish between spam SMS and legitimate SMS. A data set that contains 5573 real SMS has been used to train the algorithms K-Nearest Neighbor, Support Vector Machine, Naive Bayes and Logistic Regression. The different methods that have been used to represent text are Bag of Words, Bigram and Word2Vec. In particular, it has been investigated if semantic text representations can improve the performance of classification. A total of 12 combinations have been evaluated with help of the metrics accuracy and F1-score.

The results shows that Logistic Regression together with Bag of Words reach the highest accuracy and F1-score. Bigram as text representation seems to work worse then the others methods. Word2Vec can increase the performnce for K-Nearst Neigbor but not for the other algorithms.

Acknowledgements

I would like to thank my supervisor Kai-Florian Richter for all good advice and guidance through the project. I would also like to thank all my classmates for help and support during the education, you have made it possible for me to reach this day.

Contents

1 Introduction

1.1 Background

An SMS spam message is any unwanted message that is sent to users' mobile phone. Spam messages include advertisements, promotions, awards and SMS based phishing. SMS phishing are messages which are sent with the intention to steal sensitive information. SMS spam can come from less serious companies that people have handed out their phone numbers to. For exemple when ordering food and leaving a phone number. It is also common that people by mistake agree to receiving SMS and starts to get spammed, for example when they shop online. Spammers collect the telephone numbers in different ways. They steal, buy, or in other ways get the client database from the operators. Some databases with phone numbers may even be used from the Internet.

The growth of the number of mobile phone users has led to a dramatic increase in SMS spam messages. According to a report that was published by Mobile Ecosystem Forum[1], 28% of the surveyed said that they recieved spam every day. 58% said that they recieved spam at least ones a week and only 16% replied that they never received spam. But it can also vary widely between countries. Some countries have legislation that makes it more difficult for spammers. One of those countries which has really big problems with SMS spam is India. 96% of the people in India recieved SMS spam every day, according to a survey of over 12 000 people by online community platform LocalCircles[2].

Most people experience spam as very annoying and something that draws attention from other things. But some SMS spam can also be more dangerous, when users read the messages and click on a link in the content, it may happen that they will download malware onto their devices or are directed to malicious websites.

To reduce the risk of people being affected by this, it is important to be able to distinguish between spam SMS and legitimate SMS. In this project, different methods of classifying spam SMS will be evaluated in order to determine which one is best suited for use in a spam filter for mobile phones.

Previously, there are many studies done on E-mail spam filtering[3]. E-mail consists of considerably more text than a SMS message, which makes it uncertain whether they can be directly applied to detect SMS spam.

1.2 Purpose and Research Questions

The study will focus on answering which combination of text representation and classification algorithms that gives the highest accuracy and F1-score in the problem of SMS spam classification. Especially, it will be studied if semantic text representation can improve the performance of classification. The various method of text representations that will be analyzed are Bag of words, Bigram and Word2Vec. The classification algorithms that will be evaluated are K-Nearest Neighbor, Naive Bayes, Support Vector Machine and Logistic Regression.

2 Related Work

Over the past 15 years, various methods have been used to detect SMS spam. A number of studies with different focus have been done, in this chapter some of these are presented.

In 2006, Gomez et al[4] investigated how Bayesian filtering techniques used to block E-mail spam could be applied to the problem of detecting mobile spam. Naive Bayes, C4.5 and Support Vector Machine were evaluated on a data set of SMS from Spanish Vodafone users as well as an English SMS database from the Internet. There conclusion was that Support Vector Machine perform better then the other algorithms most of the times, and that Bayesian filtering techniques can be effectively transferred from E-mail to SMS spam even though the messages are shorter in length. Since E-mail has been used longer than SMS, it is no wonder that one of the first studies in the field used methods for classifying E-mail spam on SMS spam. But other recent studies have also tested other methods. In 2009, Sohn, Lee, and Rim[5] investigated stylistic features based on shallow linguistic analysis for learning mobile spam filters. The stylistic features used were word and sentence lengths, frequencies of function and part-of-speech tags. They did an experiment when a maximum entropy model were trained on 30 000 Korean SMS. Their experiments showed that the newly added stylistic features effectively contributes to statistically significant improvement on the performance of mobile spam filters.

Obtaining good data to evaluate methods can always be difficult. Almeida et al[6] compared the performance of several established machine learning methods. They have produced a new SMS spam collection, that according to them is the largest one ever produced. They use the same data set as in this project. Their conclusion was that Support Vector Machine outperformed other classifiers such as Naive Bayes, C4.5 and K-Nearest Neighbors.

Choosing the features that characterize spam is probably at least as important as the choice of algorithm. Uysal et al[7] used feature selection methods based on Information gain and Chi-square test to find out which features that were most important to classify SMS spam. Two different Bayesian-based classifiers were tested with the selected feature subsets with varying sizes ranging from 10 to 50. A mobile application for Android was also developed in order to use the spam filter in real-time.

A study that differs from the others was made in 2014 by Karami and Zhou[8]. They proposed a framework for detecting SMS spam. They investigated two categories of features, SMS Specific features and Linguistic Inquiry and Word Count. SMS specific feature included feautures like number of capital words, number of spam words, number of URLs etc. Linguistic Inquiry and Word Count is an analysis tool for analyzing 80 different features including linguistic processes and psychological processes. They used the classifiers Support Vector Machine and Random Forest to evaluate the different categories. Their study showed that SMS Specific and Linguistic Inquiry and Word Count could improve the classification performance.

3 Theoretical Background

3.1 The Choice of Algorithms

Previous studies in the field have mainly focus on the choice of classifying algorithms and methods for feature selection[4][6][7][8]. But it would also be of great interest to investigate how different methods for text representation affect the classification. Therefore, this study will also investigate semantic methods, such as Bigram and Word2Vec. The machine learning classifiers Naive Bayes, Support Vector Machine and K-Nearest Neigbor were successfully used in [4][6][8]. Because they previously worked well and to be able to make a good comparison with other types of text representations, they will also be used in this study.

3.2 Classification Algorithms

Machine learning algorithms can be divided into three main categories. Supervised learning, unsupervised learning and reinforcement learning. Supervised learning is when a mapping function between input variables X and output variables Y is going to be found. The goal is to approximate the mapping function so well that new data can be predicted by using the mapping function. All the algorithms used and described in this project to classify SMS spam are supervised learning algorithms. Some of the algorithms in this chapter are illustrated in two dimension but work in the same way when they are used in multi dimensional space.

3.2.1 Logistic Regression

Linear regression is used for finding a relationship between a dependent and an explanatory variable. In machine learning it is used to predict a continuous quantity. The problem of categorize SMS as spam or not spam is a binary classification problem which means that linear regression cannot be used directly. That is because it is not possible to decide a point on the x-axis from where all the values lie to its left belongs to one class and all the values lie to its right belongs to another class. Logistic regression is a linear method, based on linear regression, but the predictions are transformed using a Sigmoid function (3.1). The Sigmoid function have the domain of all real numbers, with return value from 0 to 1 and is used to transform a continous value to a probability value that can be mapped to a discrete class. If observed data have been approximated to the the linear function $y(x) = kx + m$, the logistic equation will become as in 3.2.

$$f(t) = \frac{1}{1 + e^{-t}} \tag{3.1}$$

$$f(t) = \frac{1}{1 + e^{-(kx+m)}} \tag{3.2}$$

3.2.2 K-Nearest Neighbor

K-Nearest Neighbor algorithm is one of the simplest classification algorithms to understand because no advanced calculations are done. K-Nearest Neighbor algorithm uses the neighbors data points to predict the target class. The training phase just consist of placing all the training data in a multidimensional space. In the classification phase, the euclidean distance to all the data points in the training set is calculated. New data is then classified by assigning the label which is most frequent among the K nearest neighbors. In general the boundary between classes becomes smoother with an increasing value of K.

Figure 1 shows an example of how K-Nearest Neighbor algorithm does the classification. The black test sample should be classified either to the first class of blue dots or to the second class of red dots. If $K = 3$ it is assigned to the first class because there are 2 blue dots and only 1 red dot inside the inner circle. If $K = 6$ it is assigned to the second class because there are 4 red dots and 2 blue dots in the outer circle.

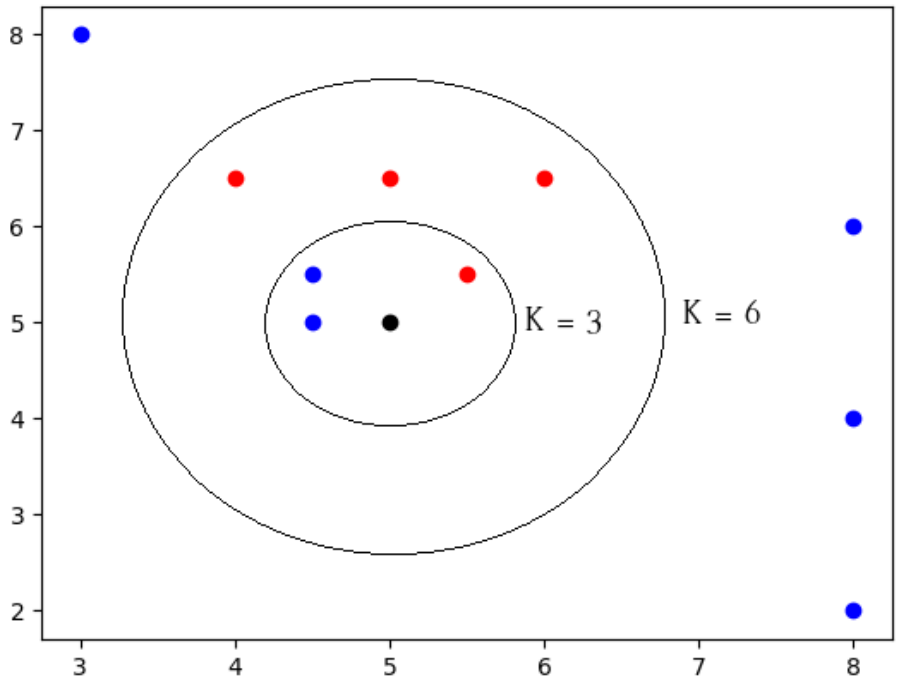

Figure 1: Example of K-Nearest Neighbor classification.

3.2.3 Support Vector Machine

The simplest kind of Support Vector Machine is the linear classifier. Support Vector Machine use the training set to find a hyper plane as a decision boundary between classes. This is shown in Figure 2. Data points closest to the hyper plane are called support vectors. The margin of the classifier is the distance between support vectors. The optimal separating hyper plane is the one that maximizes the margin. The problem of finding that hyper plane can be formulated as a standard quadratic programming problem[9]. In the test phase, data points will be assigned a label depending on which side of the hyper plane they are located.

If the the two classes are not linear separable, instead of trying to fit a non-linear hyper plane, the data can be transformed into a higher dimension that has a clear dividing margin between classes. This is called the kernel trick and relies on using a suitably chosen basis function in order to do a non-linear transformation[9].

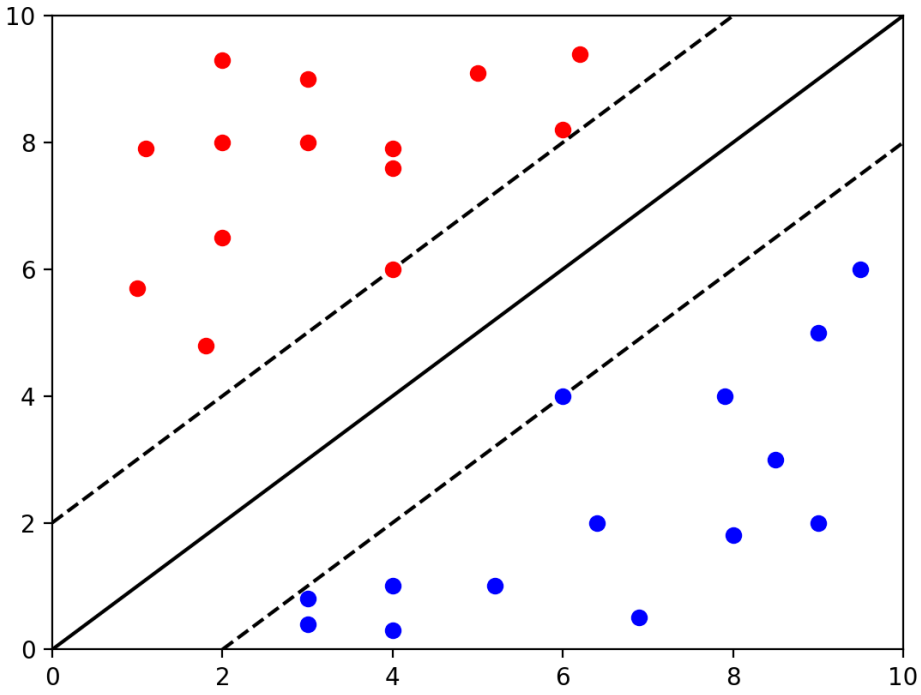

Figure 2: Example of a linear Support Vector Machine. The solid line is the separating hyper plane and the dashed line are the support vectors.

3.2.4 Naive Bayes

Naive Bayes is a machine learning classifier based on Bayes' theorem (3.1), where y is the class variable and $X = (x_1, x_2, x_3, ..., x_n)$ is the feature vector. Naive Bayes classifier often perform well in text classification and is widely used because of their simplicity. Training and prediction times are very fast, which makes it suitable for high dimensional data sets.

The probability of a class given a certain feature is called the conditional probability. Naive Bayes assume conditional independence between every pair of features. Naive Bayes aims to find the probability of a class given some observed features. By calculating the probability of each feature belonging to each class and then multiplying the conditional probabilities together for each feature for a given class (3.2), the probability of a new data point belonging to that class can be found. To make a prediction, the probability of a data point belonging to each class is calculated and then the class with the highest probability is selected.

$$P(y|X) = \frac{P(X|y) \cdot P(y)}{P(X)} \tag{3.3}$$

$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)} \tag{3.4}$$

How the algorithm works will be described by a simple example. Table 1 shows a fictional data set of tomatoes and capsicums. To use the formula 3.2, $P(x_i|y)$ has to be calculated for each x_i in X, all these calculations can be seen i Table 2.

Table 1 Shows a data set of 10 samples, describing the features for tomatoes and capsicums.

Color	Taste	Size	Class
Red	Sweet	Medium	Tomato
Red	Sweet	Medium	Tomato
Red	Sweet	Small	Tomato
Green	Sour	Medium	Tomato
Yellow	Sweet	Large	Tomato
Green	Bitter	Small	Capsicum
Yellow	Sweet	Medium	Capsicum
Yellow	Sweet	Medium	Capsicum
Red	Sweet	Large	Capsicum
Red	Sweet	Medium	Capsicum

Table 2 Conditional probabilities.

Color		
	P(Tomato)	P(Capsicum)
Red	3/5	2/5
Yellow	1/5	2/5
Green	1/5	1/5

Taste		
	P(Tomato)	P(Capsicum)
Sweet	3/5	4/5
Sour	1/5	0/5
Bitter	1/5	1/5

Size		
	P(Tomato)	P(Capsicum)
Small	1/5	1/5
Medium	3/5	3/5
Large	1/5	1/5

Total	
Vegetable	Probability
Tomato	5/10
Capsicum	5/10

Now it is possible to classify a new vegetable, based on color, taste and size. For example, if a vegetable are yellow, sweet and large, the feature vector is $X = (yellow, sweet, large)$ and the probability of it being a tomato is calculated in 3.3. The probability of it being a capsicum is calculated in 3.5.

$$P(Tomato|X) = P(Yellow|Tomato) \cdot P(Sweet|Tomato) \cdot P(Large|Tomato) \cdot P(Tomato) \quad (3.5)$$

$$P(Tomato|X) = \frac{1}{5} \cdot \frac{3}{5} \cdot \frac{1}{5} \cdot \frac{5}{10} = 0.012 \quad (3.6)$$

$$P(Capsisum|X) = P(Yellow|Capsicum) \cdot P(Sweet|Capsicum) \cdot P(Large|Capsicum) \cdot P(Capsicum) \quad (3.7)$$

$$P(Capsicum|X) = \frac{2}{5} \cdot \frac{4}{5} \cdot \frac{1}{5} \cdot \frac{5}{10} = 0.032 \quad (3.8)$$

To get the probability value, these numbers have to be normalized.

$$P(Tomato|X) = \frac{0.012}{0.012 + 0.032} \approx 27.3\% \quad (3.9)$$

$$P(Capsicum|X) = \frac{0.032}{0.012 + 0.032} \approx 72.7\% \quad (3.10)$$

So in this example, the new vegetable had be classified as a capsicum because $72.7 > 27.3$. The same principle applies when SMS is classified as spam or not spam. The only difference is that there are more data and much more features.

3.3 Feature Selection with Chi-square Test

Feature selection is the process of selecting the most relevant features from a data set. If irrelevant and redundant information is removed and the right subset of features are chosen, the accuracy of a model can often be improved[10]. A large number of irrelevant features also increases the training time and increase the risk of overfitting. The two most common classes of feature selection algorithms are filter methods and wrapper methods.

When using filter methods the selection of features is independent of the machine learning algorithms used. Filter methods using various statistical tests for calculating features correlation with the outcome variable. The features are then ranked by a calculated score and either selected to be kept or removed from the data set. Filter methods often considers all features independently of each other. Example of filter methods are Chi-square, Pearson's correlation and ANOVA.

Wrapper methods consider the selection of features as a search problem. Different combinations of features are evaluated and compared with other combinations. These methods are usually computationally expensive since a new model has to be trained for every combination that going to be evaluated. Example of wrapper methods are Forward selection, Backward elimination and Recursive feature elimination.

In this project Chi-square test has been used to reduce the number of features when the text is represented as Bag of Words or Bigram. Chi-square test is a well proven method for feature selection and has performed well in previous text classification tasks[11]. Chi-square test is used in statistics to test the independence of two events. In feature selection, the two events are occurrence of the feature and occurrence of the class. Chi-square test measures how much the observed occurrence of a feature, and the expected occurrence of a feature for a certain class differs from each other. The Chi-square score is calculated by the the formula 3.3 and the expected frequency is the number of expected observations of a class if there was no relationship between the feature and the class. When the two events are independent, the observed count is close to the expected count and gives a small Chi-square score. When a feature is correlated with a class, Chi-square test gives a high score. The Chi-square score is calculated between each feature and the target class. The features with the highest Chi-square score is then selected.

$$\chi^2 = \frac{(Observed\,frequency - Expected\,frequency)^2}{Expected\,frequency} \tag{3.11}$$

3.4 Text Representation

Machine learning algorithms cannot work with raw text documents directly. Documents have to be represented as vectors of numbers with equal size. The technique of transforming text into vectors is called feature extraction. There are different ways to do this, all with their advantage and disadvantage. Three different methods described here are Bag of Words, Bigrams and Word2Vec.

3.4.1 Bag of Words

One the most popular method for feature extraction from a corpus of documents is called Bag of Words. It is based on the idea of representing each document as a vector. All the vectors have the length that is equal to the number of words in the vocabulary. Each index in the vector describes occurrence of a specific word. The idea behind this representation is that documents that have significant similarities also have similar content. The advantage of this representation is mainly that it is simple to understand and easy to implement. If the data set is small and context is domain specific, Bag of Words may work better than more advanced methods based on semantic text representation[12]. But there are also disadvantages, the length of the vectors will be as long as the number of words in the vocabulary. It is not uncommon with a vocabulary with 100 000 words. It provides large vectors that are heavy to work with in terms of computational complexity. Another thing to consider is that the information about the order of the words in a document is discarded, which means that the semantics of a document is lost.

In Listing 3.1 there is an example of a corpus which contains four documents. The first step is to make a vocabulary of unique words. If case and punctuation are ignored, the unique words from the corpus are shown in Listing 3.2. Listing 3.3 shows the occurrence of every word in the first document. The vector representation of the first document will then be like in equation 3.12. And the representation of the whole corpus is shown in 3.13.

Listing 3.1: Example of a corpus.

```
Today it is sunny and tomorrow it will be rain.
Today it is rain.
Tomorrow it will be sunny.
Tomorrow it will be rain.
```

Listing 3.2: Vocabulary of unique words.

```
today
it
is
sunny
and
tomorrow
will
be
rain
```

Listing 3.3: Words in the first document.

```
today = 1
it = 2
is = 1
sunny = 2
and = 1
tomorrow = 1
will = 1
be = 1
rain = 1
```

$$\begin{bmatrix} 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{3.12}$$

$$\begin{bmatrix} 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{3.13}$$

When the text is represented in this format, the machine learning classifiers, together with a label, can be used to train a model.

3.4.2 Bigram

Bigram is a text representation method similar to Bag of Words but with the the difference that Bigram counting sequences of two words when building the vocabulary. By counting sequences of two words instead of one, the idea is to catch a deeper meaning of a text. If the same example is used as for Bag of Words (3.1), the vocabulary is shown in Listing 3.4. Then the occurrence of every word in the first document is counted in the same way as for Bag of Words, which is shown in Listing 3.5. The vector representation of the first document will then be like in equation 3.14. And the representation of the whole corpus is shown in 3.15.

Listing 3.4: Vocabulary for Bigram.

```
today it
it is
is sunny
sunny and
and tomorrow
tomorrow it
it will
will be
be rain
is rain
be sunny
```

Listing 3.5: Words in the first document.

```
today it = 1
it is = 1
is sunny = 1
sunny and = 1
and tomorrow = = 1
tomorrow it = 1
it will = 1
will be = 1
be rain  = 1
is rain = 0
be sunny = 0
```

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{3.14}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{3.15}$$

3.4.3 Word2Vec

Word2Vec is a group of models used for learning vector representations of words. Word2Vec was devoloped by Mikolov[13] in 2013 at Google. The purpose of Word2Vec is to group vectors of similar words together in a vector space. The meaning of that is to capture the context of a word in a document, semantic similarity and relation with other words. The idea behind Word2Vec is that the meaning of a word can be inferred by the surrounding words. This is based on the Distributional Hypothesis, which states that words that appear in the same contexts share semantic meaning and was first formulated by Harris[14]. Word2Vec is a two-layer neural network that processes text. The input is a text corpus and the output are vectors, typically of several hundred dimensions, for every word in the vocabulary.

If Word2Vec is trained on big enough data sets, like billions of words, Word2Vec can make highly accurate estimation about a word's meaning based on past appearances. Not only that similar words are grouped together in the same direction, algebraic operation can also be applied, for example that vector("King") - vector("Man") + vector("Woman") gives a vector very close to the vector representation of "Queen"[15].

When Word2Vec require huge data sets and a lot of training to perform well, it is common to use pre trained models. Word2Vec can mainly use two different types of architectures, both using neural networks. One architecture is called Common Bag of Words and takes the context of each word as the input and tries to predict the word corresponding to the context. Another architecture is called Skip-gram and takes the target word as input and tries to predict the context and produce the representation in that way. Common Bag of Words is faster but Skip-gram is better for infrequent words.

In order to represent a document with help of Word2Vec, the vectors for all the words in the document is summed to one vector, which is normalized. The final vector then represent the document.

4 Method

To answer the research question, a labeled data set with SMS messages have been used to train different classifiers with help of a machine learning library called Scikit-learn[16]. Scikit-learn is a library for Python and provide all the needed algorithms, such as K-Nearest Neighbor, Naive Bayes, Support Vector Machine and Logistic Regression. All combinations of text representation and classifications algorithms have been tested. A total of 12 different combinations.

4.1 Data Set

The data set which will be used in the study contains of 5574 SMS messages in English, labeled being a legitimate message or a spam message. 4827 are labeled as legitimate messages and 747 are labeled as spam messages. The data set has been collected by Tiago Agostinho de Almeida and José María Gómez Hidalgo[17]. It originates from various sources at the internet, such as Grumbletext web site[18], which a UK forum in which mobile phone users make public claims about SMS spam messages, NUS SMS Corpus[19], which is a corpus of SMS messages collected for research at the Department of Computer Science at the National University of Singapore, messages collected from Caroline Tag's PhD Thesis[20] and messages collected from SMS Spam Corpus v.0.1 Big[17]. All SMS are completely anonymous which means that no sensitive information is spread.

4.2 Feature Extraction and Text Preprocessing

The data set is representated as a text file that contain one message per line. Each line is composed by two columns, the first with a label and the second with raw text. To be able to use the machine learning classifiers, the data has to be transformed into vectors. The method of transforming the text into vectors differs a lot depending on the text representation. More can be read below. Text preprocessing is an important task in the field of text classification and can really improve the accuracy if done properly[21]. Before the messages were transformed into vectors all characters were lowercased and all non-alphabetic characters were removed.

4.3 Bag of Words and Bigram

When the text is to be represented as Bag of Words or Bigram a class called CountVectorizer from the Sckikit-learn library, has been used. CountVectorizer contains functions in order to learn a vocabulary, encode a document as vectors and much more. To reduce the number of features the Chi-squared statistics test has been used with help from the class Chi2 from Sckikit-learn library. The number of features for representing a text messages has been set to 2500. That is because experiments made showed a good results with that number of features.

4.4 Word2Vec

In order to represent the messages with help of Word2Vec, Google's pre-trained model has been used. It includes word vectors for a vocabulary of 3 million words and phrases that were trained on more then 100 billion of words from a Google News dataset. The vector length is 300 features. To convert a message into a vector a function has been written. The function reads a word from the message, look it up in a table and get the pre-trained vector, then the vectors for every word are summed and finally normalized to get a vector that represent that particular message.

4.5 Evaluation measure and cross validation

In order to evaluate which combination gives the best results, the accuracy has been calculated. This metric is calculated as percentage correct classifications of the total number of classifications. This is intuitive and easy to understand but also has its disadvantages. When working with a unbalanced data set like this, where one class constitutes 87% of the data, there is a risk that the result is perceived as better than it actually is. Another drawback is that false positive is not seen as the problem it really is. For example when important messages is classified as spam and these messages become lost.

With that in mind a metric called F1-score has also been used. F1-score combines precision and recall relative to a specific positive class. The F1-score can be seen as a weighted average of the precision and recall. Precision tells how many of the predicted positive actual are positive and recall tells how many of the actual positives capture through classify it as positive. F1-score reaches its best value at 1 and worst at 0. F1-score is good to use when a balance between precision and recall is needed. Equation 4.1 shows how precision is calculated. Equation 4.2 shows how recall is calculated and in equation 4.3 the F1-score is calculated.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{4.1}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{4.2}$$

$$F1 - score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{4.3}$$

All measurements were made with cross validation. Cross validation is used in machine learning to evaluate the performance of a machine learning model to a data set that is independent of the data that were used to train the model. Cross validation is good to use because it generally results in a less biased estimation[22] and gives a less optimistic estimate of the model performance then a simple train/test split.

To perform cross validation, the first step is to shuffle the data set randomly. Then the data set is split into k equal groups. One group is then selected to be the test data and the remaining groups are used for training. This procedure is repeated k times, so every group have been used as the test set. The result of the cross validation is the mean score of all trained models. The number of groups have been selected to 10 because it has proved to work well in previous studies[22].

5 Result and Analysis

In this chapter each subsection shows the result for a certain algorithm. The highest score for each text representation is written in bold and the highest score for each algorithm is written in red. According to equation 4.3, the F1-score can not be calculated if the denominator is zero. If that is the case, it has been indicated by 0. The chapter ends with an overall analysis.

5.1 K-Nearest Neighbor

Table 3 Shows the scores for K-Nearest Neighbor for different values of K.

K-Nearest Neighbor						
	Bag of Words		Bigram		Word2Vec	
	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score
K = 3	**0.918**	**0.583**	**0.920**	**0.591**	**0.978**	**0.921**
K = 5	0.905	0.478	0.899	0.426	**0.978**	**0.921**
K = 10	0.879	0.223	0.865	0.053	0.974	0.904
K = 20	0.863	0.023	0.861	0	0.964	0.862
K = 30	0.861	0	0.861	0	0.960	0.844
K = 40	0.861	0	0.861	0	0.959	0.839

When looking at Table 1, the accuracy and F1-score decrease when the number of neighbors increase. It applies to all types of text representations. It is not that surprising, because when the data set is so unbalanced, the frequency of not spam is much higher than spam and it makes that all data points will be classified as not spam when checking at many neighbors. It can also be concluded that Word2Vec has a significant higher accuracy and F1-score then the other methods of text representations. The difference in score between Bag of Words and Bigram are generally low and both perform quite bad together with K-Nearest Neighbor. When Bag of Words and Bigram looking at 30 and 40 neighbors, the algorithm will classify everything as not spam.

5.2 Support Vector Machine

Table 4 Shows the scores for Support Vector Machine for different kinds of kernel functions.

Support Vector Machine						
	Bag of Words		Bigram		Word2Vec	
	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score
Linear	**0.982**	**0.929**	**0.971**	**0.884**	**0.978**	**0.919**
Rbf	0.98	0.922	0.963	0.846	0.962	0.851
Sigmoid	0.965	0.857	0.899	0.431	0.956	0.8

Table 2 shows that the Linear kernel always gives the best accuracy and F1-score, no matter which method of text representation is chosen. It is also clear that Bag of Words perform better then Bigram and Word2Vec, no matter which kernel function is used. Also note that Bigram together with the Sigmoid kernel function works quite bad.

5.3 Naive Bayes

Table 5 Shows the scores for Naive Bayes.

Naive Bayes					
Bag of Words		Bigram		Word2Vec	
Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score
0.978	**0.918**	**0.974**	**0.909**	**0.961**	**0.866**

Naive Bayes does not have any parameters that can be changed. But quickly it can be concluded that Bag of Words outperform Bigram and Word2Vec, when measuring accuracy and F1-score.

5.4 Logistic Regression

Table 6 Shows the scores for Logistic Regression for different solvers.

Logistic Regression						
	Bag of Words		Bigram		Word2Vec	
	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score
Liblinear	**0.985**	**0.944**	**0.974**	**0.899**	**0.983**	0.936
Lbfgs	**0.985**	0.942	**0.974**	**0.899**	**0.983**	**0.938**
Saga	0.980	0.928	0.972	**0.889**	**0.983**	**0.938**

Table 4 shows the scores for Logistic Regression. Only marginal differences can be seen when accuracy and F1-score are compared between different solvers. But once again, Bag of Words perform a little bit better then Bigram and Word2Vec when talking about accuracy and F1-score.

5.5 Overall Analysis

One thing that stands out when the algorithms are compared is that Bag of Words is the best way of representing text for all algorithms except for K-Nearest Neighbor. For K-Nearest Neighbor, Word2Vec is clearly much better than Bag of Words and Bigram. When it comes to Bag of Words and Bigram, Support Vector Machine, Naive Bayes, and Logistic Regression performs quite equally. It can also be concluded that Bigram usually results in a worse result then Bag of Words and Word2Vec, although the difference is small. The overall winner is Logistic Regression, when uses a Liblinear solver together with Bag of Words. It reach both the highest accuracy and the highest F1-score.

6 Discussion

The main question to answer in this section is whether semantic text representation can improve the performance of classification. But also compare the result in a larger context and with other studies in the field of spam SMS classification.

As previously stated, there is nothing that can prove that the use of Bigram as text representation can improve the performance of classification. Rather, it can be concluded that it works worse. Among the papers listed in related work, Bigram has not been investigated in field of spam SMS classification but it is a well studied method of text representation. In [23] Bekkerman and Allan writes about unsuccessful attempts to improve text categorization by applying Bigrams and that there might be certain limitation in usability of Bigrams. It may well be that it also applies to classification of SMS messages. Since Bigram theoretically increases the number of words in the vocabulary and SMS contains very little text, maybe Bigram, just making it harder to find what is characteristic of a spam SMS.

Looking at Word2Vec, it is remarkable that it works much better then Bag of Words together with K-Nearest Neighbor but worse when used with other algorithms. What that depends on is hard to say but Word2Vec and Bag of Words uses completely different methods when converting a text messages to a data point in a vector space. Probably Word2Vec group the data points in a way that is favourably for the algorithm K-Nearest Neighbor. If the accuracy is compared with Almeida et al.[6] study, it is clear that the use of Word2Vec can improve the performance for K-Nearest Neighbor. In this study, an accuracy of 97,8% was reached, compared with 91,0% in their study. But then it should be mentioned that this study used 67% of the data set as training data and their study used 30%.

In this study a Chi-square test was used to select the 2500 most relevant features. It seems like it had an impact on the result. If the accuracy for Support Vector Machine and Naive Bayes together with Bag of Words are compared with Almeida et al.[6] study, who did not use any method for feature selection, the accuracy are better for Support Vector Machine and much better for Naive Bayes. But ones again, this study used more data as training data and also a different method for text pre processing so a comparison can be little difficult to do.

In 2014 Karami and Zhou[8] also used a Support Vector Machine to evaluate their framework for detecting spam SMS. The accuracy in their study is quite equal to best accuracy for Support Vector Machine in this study. But when looking at their F1-score, it is considerably better than in this project. They used completely different methods to produce features than in this project and obviously they were successful with their SMS-specific and Linguistic Inquiry and Word Count methods for feature selection. Logistic Regression together with Bag of Words had a F1-score of 0.944, which was the best in this study. But it was still worse than their best F1-score that reached 0.982. With that in mind, it may also be necessary to consider other methods than statistical when selecting features.

6.1 Conclusion and Recommendations

The conclusion regarding Bigram as text representation is that it does not work well enough and therefore should not be used in the field of spam SMS classification. Word2Vec has shown that it can improve the performance of K-Nearest Neighbor but not for other algorithms. The recommendation is not to use this method because Bag of Words seems to work better in general. Word2Vec is a really interesting way of representing text and maybe it is possible to make it work better, but in this study that is note the case. Another drawback with Word2Vec is that it requires a lot of memory to store all words and phrases. It can be a problem when an application should be developed for a mobile phone with a limited memory. Bag of Words is an old proven method of text representation and in this study it has also proved to work best. If a spam filter going to be used in reality on a mobile phone it is important that the classification goes fast. Logistic Regression, Support Vector Machine and Naive Bayes have all a very fast test phase. So the recommendation, based on this study is to use Logistic Regression together with Bag of Words.

6.2 Future Work

One type of machine learning algorithms that has not been investigated in this study is neural networks. It is in itself a very large area to investigate and probably need a much bigger set of data to really perform. But it would have been interesting to investigate how they should perform in the field of spam SMS classification.

As previously said Karami and Zhou[8] were successful with their SMS-specific and Linguistic Inquiry and Word Count methods for feature selection. This is certainly something that can be developed further, by identify what really is characteristic of a spam SMS and selecting features in that way in order to increase performance.

This study has just focused on the theory behind spam SMS filtering but it would also be interesting to develop a mobile application that is based on the conclusions from this study.

This project is all about text classification methods to detect spam SMS, but of course, there are completely different methods that can handle the problem. There is a popular mobile application that uses a database containing black listed phone numbers, when a new SMS is received, it is controlled against this database to make a classification. It does not have anything to do with text classification but it shows that there are other ways to go to solve the problem.

References

[1] Mobile messaging fraud report 2016. Technical report.

[2] Survey by online community platform LocalCircles. Technical report.

[3] Enrico Blanzieri and Anton Bryl. A survey of learning-based techniques of email spam filtering. *Artificial Intelligence Review*, 29(1):63–92, 2008.

[4] José María Gómez Hidalgo, Guillermo Cajigas Bringas, Enrique Puertas Sánz, and Francisco Carrero García. Content based SMS spam filtering. In *Proceedings of the 2006 ACM symposium on Document engineering*, pages 107–114. ACM, 2006.

[5] Dae-Neung Sohn, Jung-Tae Lee, and Hae-Chang Rim. The contribution of stylistic information to content-based mobile spam filtering. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 321–324. Association for Computational Linguistics, 2009.

[6] Tiago A Almeida, José María G Hidalgo, and Akebo Yamakami. Contributions to the study of SMS spam filtering: new collection and results. In *Proceedings of the 11th ACM symposium on Document engineering*, pages 259–262. ACM, 2011.

[7] Alper Kursat Uysal, Serkan Gunal, Semih Ergin, and Efnan Sora Gunal. A novel framework for SMS spam filtering. In *2012 International Symposium on Innovations in Intelligent Systems and Applications*, pages 1–4. IEEE, 2012.

[8] Amir Karami and Lina Zhou. Improving static SMS spam detection by using new content-based features. 2014.

[9] Ethem Alpaydin. *Introduction to machine learning*. The MIT Press.

[10] Avrim L Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial intelligence*, 97(1-2):245–271, 1997.

[11] George Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of machine learning research*, 3(Mar):1289–1305, 2003.

[12] 3 basic approaches in bag of words which are better than word embeddings. https://towardsdatascience.com/3-basic-approaches-in-bag-of-words-which-are-better-than-word-embeddings-c2cbc7398016. Accessed: 2019-04-08.

[13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[14] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[15] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013.

[16] Scikit learn. https://scikit-learn.org/. Accessed: 2019-06-05.

[17] Sms spam collection. http://www.dt.fee.unicamp.br/ tiago/smsspamcollection/. Accessed: 2019-04-20.

[18] Grumbletext web site. http://www.grumbletext.co.uk/. Accessed: 2019-04-20.

[19] Nus sms corpus. http://www.comp.nus.edu.sg/ rpnlpir/downloads/corpora/smsCorpus/. Accessed: 2019-04-20.

[20] Caroline Tagg. *A corpus linguistics study of SMS text messaging*. PhD thesis, University of Birmingham, 2009.

[21] Alper Kursat Uysal and Serkan Gunal. The impact of preprocessing on text classification. *Information Processing & Management*, 50(1):104–112, 2014.

[22] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[23] Ron Bekkerman and James Allan. Using bigrams in text categorization. Technical report, Technical Report IR-408, Center of Intelligent Information Retrieval, UMass ..., 2004.

UMEÅ UNIVERSITY