Dialect Identification on Twitter

A research about the detection of the Limburgian dialect from Twitter messages

Paul van Cann

Anr 649040

Master's Thesis

Communication and Information sciences

Specialization Human Aspects of Information Technology

Faculty of Humanities

University of Tilburg, Tilburg

Supervisor: Dr. G.A. Chrupala

Second Reader: Dr. P.H.M. Spronck

January 19, 2015

Abstract

Dialect identification is a recent area in Machine learning. In the current study has been investigated if it possible to build a system that identifies the Limburgian dialect, and three different algorithms have been used for the classification between Dutch and Limburgian Twitter messages. That is, a Logistic Regression, Linear Support Vector Machine and a Multinomial Naïve Bayes classifier. The data used for this classification has been collected using the Twitter API, and was divided into a training, development, and test set. Secondly, the models were trained on the training data, and different tests were conducted on the development data by adjusting the parameters of the different classifiers. Finally, the parameters which performed best on the development set were used for a final prediction on the test set. The results showed that the Multinomial Naïve Bayes performed the task of identifying the Limburgian dialect best, and was able to achieve an accuracy of 94 percent on fivegrams.

Content

1. Introduction

Social networking is an old mechanism for mediating interactions between people, and has become more important in the age of the web, or how it is also referred to, the information age. The internet has become an important component in life, and probably the largest shift has taken place in how people socialize and seek-out and spread information. Specifically, online information transfer and social interaction sites are the fastest growing types. For example, Facebook's popularity has grown exponentially from 5.5 million active users in 2005 to around 500 million in 2011 (Hughes, Rowe, Batey & Lee, 2012), becoming one of the largest social networking applications in the world. Furthermore, another familiar social networking site Twitter, consisted of around 200 million registered accounts in 2011 (Hughes et al., 2012).

Academically, with the current spread of such global communication possibilities on the internet, text is available in a large number of languages, and automatic treatments of these texts have become increasingly important.

According to Ljubešić, Mikelić and Boras (2007) it is not possible to use data for information retrieval without knowledge about the language a message is written in. This problem of research has been part of research for a long time, and several feature-based models for automatic language identification have been brought up. For example, the use of syllable characteristics, morphology and syntax, information about short words, the frequency of n-grams of characters, Markov models, and more recently Support Vector Machines (Ljubešić et al., 2007). After the text has been represented as features, it is possible to classify or label a written text as being a specific language. To perform this task, most often a corpus of written text is used on which some kind of machine learning model or classifier is trained to identify the language.

According to Biadsy, Hirschberg and Habash (2009) a lot of research has been performed on automatic language identification, but to lesser extent to dialect identification, which is the task of recognizing the regional dialect of a speaker within a known language. Although, dialect identification has been receiving more interest recently. Dialect classification is not an easy task, because dialects within the same language are in nature similar. However, dialect identification is a useful task, because it is important for improving speech recognition, to guess the regional origin of the speaker, or to detect when speakers 'switch code' (Biadsy et al., 2009). Therefore, this research will focus on the identification of a language that has not yet been considered in other research efforts, namely the Limburgian dialect.

In the Netherlands, Dutch is the native language spoken by almost the entire population. However, in some parts of the Netherlands a variation or dialect is spoken (Swanenberg, 2013). For example, in Friesland and Limburg. Limburgian is the regional dialect that is used in the most southern province of the Netherlands. Limburgian is spoken by approximately 75% of the population of Limburg, which is about 900.000 inhabitants of the total population of Limburg (Driessen, 2008).

The Limburgian dialect shows a lot of similarities with the Dutch language. However, there are also a lot of differences. The similarities are to be found mainly in the construction of sentences, while the differences are to be found in the construction of words. For example, 'is that true', means 'is dat waar' in Dutch, and is in the Limburgian dialect written as 'is det waor'.

In this research, we want to find these differences and build a system that automatically identifies the dialect of Limburg from Twitter messages. In general, we want to find answer to the following research question (RQ):

RQ: Is it possible to build a system that detects the Limburgian dialect from written Twitter messages?

To be able to find an answer to the research question some research in advance is necessary. For example, some research has to be performed considering the different systems which we can apply. Therefore, the first sub question is:

SQ1: What systems are commonly used for dialect identification, and how do they perform?

Furthermore, it is the task in this research to apply these systems to see how they perform on the identification of the Limburgian dialect. Therefore, the second sub question is:

SQ2: How do these systems perform in the classification of Dutch and Limburgian tweets?

After we explored previous research about dialect identification and commonly used classification algorithms, it was decided to use a Logistic Regression, Linear Support Vector Machine, and Multinomial Naïve Bayes for the final classification between Dutch and Limburgian twitter messages, because they were successfully applied in previous research. The Multinomial Naïve Bayes performed best, and achieved an accuracy of 94 percent on fivegrams.

2. Background

2.1 Twitter

Twitter is a microblogging application in which users are allowed to post messages of no longer than 140 characters. The messages on Twitter are also referred to as 'tweets'.

Recent figures show that Twitter had around 284 million active users in the third quarter of the year 2014 (NU, 2014). According to Bergsma, McNamee, Badouri, Fink and Wilson (2012) daily, more than 50 million users log in to their account and billions of tweets are posted every month. These tweets are publicly available and provide an enormous resource of unedited text produced by ordinary people (Bergsma et al., 2012).

Furthermore, Twitter users are able to post two different kinds of updates, namely direct updates (i.e. updates directed to a specific person), and indirect updates (i.e. updates meant for everyone that is interested to read it). The direct updates are also publicly available for everyone to read. However, most often a reference to a specific person is characteristic for these kind of tweets. The directed updates are recognized by the '@name' structure, and about 25% of all Twitter posts are directed (Huberman, Romero & Wu, 2008).

Unlike news articles, tweets vary widely in style and contain misspellings or grammatical errors. Furthermore, people switch often between the languages they use based on their target audience and the text contains special characters and abbreviations. Therefore, text from Twitter is probably one of the most challenging texts to process in a language identification task.

2.2 The Limburgian dialect

Limburg is an outstretched province adjacent to Belgium and Germany, in which the narrowest part is only a few kilometers wide. The Limburgian dialect consists of several diverse dialects brought collectively under the name Limburgian (Swanenberg, 2013). It is

possible to divide these varieties into five isoglosses, which are basically the different language areas (Heeringa, 2007).

The area in which the Limburgian dialect is spoken ranges from Venlo in the Netherlands in the north, to Dusseldorf in Germany in the east, to Maastricht in the Netherlands in the south, and to Tienen in Belgium in the west (Peters, 2007). Geographically, Limburg is divided between Dutch Limburg, and Belgium Limburg.

Limburgian is one of the three regional languages (i.e. Frisian, Low Saxon, and Limburgian) recognized by the European Charter for Regional or Minority languages, and is used by all classes of society, both in formal and informal situations. Furthermore, Limburgian enjoys high prestige by the inhabitants, and is not only used for interaction in public places but also on school (Driessen, 2006). In addition, many Limburgian speakers also interact in Limburgian online, on social media applications such as Facebook, Twitter, and instant messaging applications such as WhatsApp.

The variety that is found in the dialect, is the consequence of several historical factors. In history, there was a lot of fragmentation due to occupations and different political rulers (Ubachs, 2000). However, the most important reason is to be found in the High German Consonant Shift (HGCS), which is one of the most familiar sound shifts. The HGCS characterizes a sound shift of the stops *p, t,* and *k* towards affricates at the beginning of words and when doubled (e.g. *pf*und), and after a vowel as long fricatives. The shift spread from south to north, affecting the language in one part of the area more than the other. Its reach can still be noticed, as it is possible to draw a border between the Dutch use of the *k* and the High German use of *ch* after a vowel (Britannica, 2010).

The Uerdinger line characterizes the border (Heeringa, 2007). The city of Venlo lies just above this line, which is the reason for using the *k* instead of *ch* after a vowel. For

example, in Venlo they say au*k* instead of au*ch* (also). The rest of the Dutch Limburgian area lies south of this line and uses *ch* after a vowel.

2.3 Language Identification

Language identification is the identification of a language that is commonly used by people in communication. To perform automatic language identification on text, it is necessary to categorize a text. This task of automatically placing pre-defined labels on a previously unknown text is called text classification (Scott & Matwin, 1999).

In text classification the text is most often divided into a training and test set, and the training data is used to predict the label of the test data (i.e. the unseen data set). The training and test data needs to be balanced, to prevent overfitting (i.e. the data is too much adapted to the training data, which affects the performance on the test data) (Sriram, Fuhry, Demir, Ferhatosmanoglu & Demirbas, 2010). After the division, a text presentation is chosen as input for the classification task (Rogati & Yang, 2002).

To use machine learning algorithms it is important to find the best presentation for the text that needs to be classified (Houvardas & Stamatatos, 2006). Most often a 'bag of words' model or technique is used to represent the text, in which the text is tokenized (i.e. each word corresponds to a feature). The features extracted form a feature vector for the text (Sriram et al., 2010). In most bag of words presentations punctuation and frequent words such as stop words are removed from the vector, or a stemming or lemmatization algorithm is used to reduce the complexity of the text (Scott & Matwin, 1999).

Another approach to represent text is to use character n-grams. The extraction of n-grams is a task that is language independent. Therefore, this approach is useful for different categorization problems (Houvardas and Stamatatos, 2006). According to Martins and Silva (2005) the key benefit of n-gram based modelling is that because every string is decomposed into smaller parts, errors tend to affect only a limited number of these parts. This is especially

convenient for language identification on the web, since the level of texts differ considerably (Martins & Silva, 2005).

Often the representation approach of Cavnar and Trenkle (1994) is used, in which the frequency of occurrences of character n-grams is used as a representation. They appended blanks to the beginning and ending of each string and used n-grams of different lengths simultaneously (i.e. bi-grams, tri-grams and quad-grams). Furthermore, the approach from Cavnar and Trenkle (1994) is based on Zipf's law. Zipf's law implies that when texts from the same category are compared to each other, these texts should have quite similar n-gram frequency distributions.

The problem of using n-gram character representations is that when the training set is large, the feature space can exceed large proportions based on the n-gram size that is used (Houvardas & Stamatatos, 2006). Therefore, often an automatic feature selection algorithm is used to remove redundant features from the feature space. The most frequently used feature selection algorithms are: (1) Information Gain, which measures the number of bits of information that is gained about a category, (2) Document Frequency Thresholding, which calculates the frequency of each feature in the training set, and removes terms that appear less than some predefined threshold, and the (3) x^2 statistic, which measures the lack of independence between a term and a category (Houvardas & Stamatatos, 2006).

According to Rogati and Yang (2002) reduction of the feature space by feature selection has repeatedly shown to have little accuracy loss, and performance gain in many cases. Specifically, information gain and x^2 statistics are effective for optimizing the results of a classification task, and document frequency is better for efficiency and scalability. After the text is represented, the machine learning algorithm is able to categorize the text (Sriram et al., 2010).

Several machine learning algorithms can be used for this categorization task. For example, the k-Nearest Neighbor or Naïve Bayes approaches, or Support Vector Machines. The k-Nearest Neighbor decides whether a document or text belongs to a category by checking whether the *k* training examples most similar to the document or text are also in the category. When the proportion that belongs to the category is large enough, a positive or negative decision is taken (Houvardas & Stamatatos, 2006). Similarity can be measured by using different metrics. The most popular metric is the Cosine Similarity measure, in which is measured if two documents correspond to the correlation between the vectors. However there are also measures such as Euclidean Distance (i.e. the distance between two points in an n-dimensional space), or Pearson Correlation Coefficient (i.e. the measure of the correlation between two variables) (Huang, 2008). The k-Nearest Neighbor algorithm is a lazy learning model, which means that all computation takes place at the classification stage (Houvardas & Stamatatos, 2006).

Support Vector Machines are the most dominant method used to date. A Support Vector Machine is a classifier that tries to find a linear model, also referred to as the maximum margin hyperplane (Houvardas & Stamatatos, 2006). The maximum margin hyperplane gives the greatest separation between the categories. The instances that are closest to the hyperplane are referred to as 'support vectors' (Houvardas & Stamatatos, 2006).

Support Vector Machines are based on a principle called 'Structural Risk Minimization', which finds a hypothesis for which the lowest true error can be guaranteed. The true error of the hypothesis is the probability that an error will occur on an unseen randomly selected test example (Joachims, 2005).

Furthermore, Support Vector Machines are known to deal with large feature spaces, because they use overfitting protection, and are able to deal with the few irrelevant features

texts provide. In addition, Support Vector Machines are known for dealing with document vectors which are sparse (i.e. it is able to deal with document vectors which only contain a few entries which are not zero). Finally, Support Vector Machines are able to deal with high dimensional feature spaces. Therefore, feature selection is not necessary (Joachims, 2005).

According to Joachims (2005) experimental results show that Support Vector Machines achieve good performance on text categorization tasks, and it is a promising and easy-to-use approach.

Furthermore, Naïve Bayes probabilistic classifiers are also commonly used in text classification. The Naïve part of this algorithm is that all attributes of the examples are independent of each other. The idea behind the algorithm is that joint probabilities of words and categories estimate the probabilities of categories of a text. There are different variants of the Naïve Bayes classifiers. However, the multinomial model is often the best choice (Houvardas & Stamatatos, 2006). McCallum and Nigam (1998) used this model and concluded that this model works well. In the multinomial model probability is calculated by multiplying the probability of only the words that occur in the text (McCallum & Nigam, 1998).

Gottron and Lipka (2010) also used a multinomial version of the Naïve Bayes algorithm and compared the results to the results of three other approaches. They concluded that on short texts and single texts, an accuracy of more than 80 percent was achieved, and on longer texts close to a 100 percent. (Gottron & Lipka, 2010).

The Naïve Bayes classifier is based on a theorem of probability, known as 'Bayes rule'. Bayes rule calculates a posterior probability $P(x|y)$ (i.e. the chance that x occurs, given that y occurs). When we have certain texts that have not yet been classified, a Naïve Bayesian

classification can be used to determine to which class the text belongs. Formally, the Naïve Bayes classifier is described as:

$$P(y|\mathrm{x}) \propto P(y) \prod_{i=1}^{N} P(x_i|y)$$

The probability $P(y)$ is the proportion of the examples labeled with y. $P(x_i|y)$ is the number of times word x_i occurs in texts labeled with y, divided by the total number of words in texts labeled with y. The label with the highest score is the label the word belongs to (van Hertum, 2007).

2.4 Dialect Identification

Distinguishing between Dutch and Limburgian is the task of automatic dialect identification. According to Zaidan and Callison-Burch (2012) dialect identification is the task of building a learner that is able to determine whether or not a sentence contains dialectal content. Furthermore, Zaidan and Callison-Burch (2012) say that it is in many ways equivalent to language identification. However, dialect identification is a difficult case of Language identification, because it is applied to a closely related group of languages, which typically share the same script, and have similar spelling conventions.

In their research of identifying Arabic dialects Zaidan and Callison-Burch (2012) used the Arabic Online Commentary Dataset (Zaidan and Callison-Burch, 2011), and crowd-sourced Arabic dialect annotation. They performed an automatic classification task by using n-gram features to classify sentences in classes. The classification method was based on language modeling scoring. Finally, the results were compared to the results of the human annotation, which showed that there was a small difference between the result of the model and the result of the annotators (Zaidan & Callison-Burch, 2012).

Similarly to previous research Biadsy et al. (2009) also performed a classification task on Arabic dialects. In this research the task was to identify regional dialects within Modern Standard Arabic (MSA). A Logistic Regression was used as a classifier, because from their experience it worked better than Support Vector Machines, and the classifier was given tri-gram models as input. The algorithm proved to be quite accurate in the classification task with an accuracy of approximately 80 percent (Biadsy et al., 2009).

Finally, Sadar, Kazermi and Farzindar (2014) also tried to identify Arabic dialects from MSA. In this research social media texts from different sources were used, because these texts contain the '3V's' of big data (i.e. volume, velocity, and variety), which are basically the challenges that data from these sources entail. Furthermore, the Arabic language was chosen because it is morphologically rich, complex, and spoken by more than 350 million people around the world. A N-gram language model and a Naïve Bayes classifier were used for the classification, in which the N-gram model was used to calculate the probability of the text being derived from a given language model that was built from training data, and the Naïve Bayes classifier for applying Bayes rule. The results show that the Naïve Bayes algorithm identified more than fifteen dialects with an accuracy of almost 100 percent (Sadar, Kazermi & Farzindar, 2014).

2.5 Identification of Twitter messages

According to Bergsma et al. (2012) language identification can be quite difficult by choosing domains that contain, informal writing, a great diversity of languages, short texts, and unbalanced data. Twitter has all these characteristic, which makes a classification task challenging.

Bergsma et al. (2012) collected data from Twitter using the Twitter API (i.e. Application Programming Interface). Using the API's geotag method they collected tweets

within a specific radius, by giving the coordinates in latitude and longitude. Texts were collected from nine languages, namely Arabic, Farsi, Urdu, Hindi, Nepali, Marathi, Russian, Bulgarian, and Ukrainian. In the next step, the data was annotated via human annotation. The annotators were accessed via Mechanical Turk, which is an online marketplace in which people can post tasks that people perform against a payment. When the annotation was completed, a Logistic Regression and partial matching algorithm were used for the classification task. The feature values were calculated as log(1 + count), in which count refers to the number of times a feature is in the given example. A Logistic Regression is used to predict the language with the highest probability. A partial matching algorithm minimizes cross-entropy and selects the language that most compactly encodes the text. Both algorithms seemed to be very accurate in the task. Even with a few hundred annotated tweets in the training data, the Logistic Regression was able to get an accuracy over 90 percent (Bergsma et al., 2012).

In research from Tratz (2014) they performed two experiments, in which Twitter messages were also used. In the first experiment they used a Maximum Entropy classifier to distinguish between three languages, namely Arabic, Farsi, and Urdu. A Maximum Entropy classifier is another name for a Multinomial Logistic Regression. Feature values were calculated similarly to Bergsma et al. (2012), and log(1+count) was used to represent the features. Furthermore, the data from the research of Bergsma et al. (2012) was used, or at least to the extent that it was still available. The results showed that the classifiers accuracy was quite similar to Bergsma et al. (2012) with almost a 100 percent. They used almost 1100 tweets divided over training and testing, of which almost the entire amount of tweets was correctly classified.

In the second experiment the dataset of Zaidan and Callison-Burch (2011) was used, including about 3000 tweets of another Arabic dialect, Darija, and about 3000 tweets

collected from Twitter users with other dialects. The results from this experiment were worse, with a result of about 80 percent accuracy maximal.

Other than the works discussed, there is little research about language identification of Twitter messages available. Most research efforts focus on the classification of sentiment from Twitter messages.

However, from the different research efforts it is possible to conclude, that it is harder to identify dialects in comparison to languages. The accuracy scores achieved in language identification tasks are most often above 90 percent, while the accuracy scores obtained in dialect identification tasks are around 80 percent. A possible explanation could be, that there is a lot of similarity between MSA and its dialects. Therefore, it may be expected that the identification of the Limburgian dialect will also be a difficult task.

2.6 Logistic regression

From the researches from Biadsy et al. (2009), Bergsma et al. (2012) and Tratz (2014) it is possible to conclude that the Logistic Regression classifier is an appropriate algorithm for the task of identifying the Limburgian dialect from Twitter messages. Therefore, in this chapter, this model will be explained in addition to the Multinomial Naïve Bayes.

In its simplest form, the Logistic Regression determines if tweet x belongs to the Dutch language category or the Limburgian dialect category. To be able to predict to which language a message belongs, features of the message are used to predict the likelihood of the message being in a certain category. When there are only two categorical outcomes the regression is known as a binary logistic regression. However, when the prediction concerns more than two categories a Multinomial Logistic Regression is used.

Linear regression is based on the assumption that the relationship between variables is linear. However, when the outcome variable is categorical, this assumption is not met.

Therefore, a logarithmic transformation 'logit' is applied to the outcome variable to make the relation linear.

When the assumption is met, it is possible to perform the Logistic Regression. In the formula below, the binary logistic regression is shown. The probability that the text with features *X* is labeled with category *Y* is predicted given known values of X_i.

$$P(Y|X) = \frac{1}{1 + e^{-(\beta_0 + \sum_{i=1}^{N} \beta_i X_i)}}$$

e is the base of natural logarithms. The other coefficients are much the same as in a simple regression. β_0 is a bias, X_i is the predictor variable, and β_i is the coefficient attached to the predictor variable. The resulting value $P(Y|X)$ varies between 0 and 1, and expresses the probability that the example *X* belongs to category *Y*. A value that is close to 1 indicates that Y is very likely to have occurred (Field, 2013).

3. Methods

3.1 Materials

The materials have been collected using the Twitter API. A program has been written to collect tweets, and a query of approximately 820 words is used to find the Limburgian tweets. Furthermore, the geographical coordinates represented by the latitude and longitude are used to find messages from the Limburgian area, and within a radius of 25 kilometers outside of this border. The Dutch tweets were collected only based on these coordinates.

The full texts are used as input, and no techniques such as stop-words removal, stemming, or low-frequency words removal are used to optimize the input. Only punctuation is removed from the text, and text is made lower-case. The date of the messages is randomly chosen.

Before processing, the Limburgian messages were given the label ‘li’, and the Dutch messages were given the label ‘nl’. The Limburgian labels were annotated into ‘li’ by a Limburgian speaker, because Twitter does not recognize the Limburgian language. Twitter was able to identify the Dutch language. Therefore, there was no need to change the labels of the Dutch messages.

In this study, a 1000 messages were collected, of which 500 Limburgian and 500 Dutch messages. Because the program gave a Unicode error in the beginning of the experiment, which could not be solved easily, 13 Limburgian tweets were replaced by new tweets. As a result, the amount of tweets was still equally distributed.

3.2 Procedure

A corpus of Limburgian and Dutch text is collected from Twitter using the Twitter API, and divided into three sets of data files, a training set, a development set and a test set. The training set is for fitting the model, while the development set is for improving the model. The

test set is for testing the model on unseen data. Precisely 40 percent of the corpus was used as training classes and the remaining 60 percent was divided over the development and test sets for classification. Each dataset consists of seven columns, namely the ID of the User, the user name, the location of which the user posted the tweet, the date the tweet was placed, the text of the tweet itself, and its label Dutch or Limburgian.

In the next step, the text was preprocessed, in which punctuation was removed, and text was made lowercase. Then, for each text a vector was created by splitting the text into n-grams of different lengths and count the occurrences. This step provided a vector for each language label (Dutch or Limburgian).

Then the vectors were given to the classifier as examples, and it learned which variables correlate with which labels. The classifier was trained (i.e. finding patterns, and associating these patterns to certain outcomes) on the training set and the outcome of unseen examples were predicted from the test set. For the classification task, a Logistic Regression, Linear Support Vector Machine and Multinomial Naïve Bayes were used, because these classifiers were successfully applied to classification tasks in several previous research efforts.

Finally, the model was evaluated by using the language labels of the messages by comparing the results of the classification with the initial labels of the message. In addition, an error analysis has been conducted of the misclassified tweets.

4. Experimental setting

To be able to predict the probability of a tweet being Dutch or Limburgian, a script in the Python language is written to perform the classification task automatically. The Python program represents the experimental setting, and shows the different steps that were discussed in the method section in more detail. It is possible to assess the code through the following link: https://www.dropbox.com/s/ij6mt8tz1vzjq2i/Classifiers.py?n=50589251

4.1 Labelling of data

As mentioned earlier, the first step considered was the processing of the data. In this step, the script reads into the csv file, returns the input, and labels the Limburgian texts as '0' and the Dutch texts as '1' to be used in the training phase.

Furthermore, a baseline was created. In the baseline, the query terms used to retrieve the Limburgian tweets were matched with the text. The basic rule that was applied to this baseline was simply that when a query term was in the text, the label 'li' was given to the tweet. If the query term was not in the text, the tweet was given the label 'nl'. The query only consisted of Limburgian words, which gave the advantage that we were able to program this simple rule. The query can be found in the appendices (table 2).

The baseline was run on the development and test data, and an accuracy score was calculated of the baseline. This accuracy score was then used to see if the other models improved the model.

In the final step of processing the data, the data was preprocessed for classification according to the procedure in previous chapter.

4.2 Development of features

As mentioned earlier, to use machine learning algorithms it is important to find the best presentation for the text that needs to be classified (Houvardas and Stamatatos, 2006). Most

often a model or technique is chosen to create features. These features form a feature vector for the text (Sriram et al., 2010). After preprocessing the data, the features were created by splitting the text into character n-grams. During the classification task, different n-gram lengths were considered to see whether the accuracy of the classification task increased or decreased when the size of the n-gram characters got larger.

4.3 Parameter tuning

The goal of this research was to find the highest accuracy using different classifiers. The accuracy is the proportion of texts in Limburgian that actually have been classified as Limburgian, and the actual texts in Dutch that have been classified as Dutch.

In the first step the evaluation considers the different character n-grams lengths, and sees which length shows the highest accuracy score using default settings of the Logistic Regression, Linear Support Vector Machine, and Multinomial Naïve Bayes.

In the second step, the parameters were adjusted to see if there were improvements, and to what extent. Different settings are used, which are provided by the classifier itself. These different settings are also known as hyperparameters. In the Logistic Regression and the Linear Support Vector Machine it is possible to use L1 and L2 regularization, which can be added to the algorithm to ensure that the models do not overfit its data. The L1 regularization norm is the sum of the absolute differences between the estimated and target values, while the L2 regularization norm is the sum of square of the differences between estimated and target values.

Furthermore there are also other options. For example, in Linear Support Vector Machine classification it is possible to adjust the multi-class strategy. This option, which ignores penalty, uses contrary to the default strategy a one versus the rest scheme. However, because there are only two classes in this experiment this function is not relevant, and is not

considered. In addition, it is possible to turn on class weighting. However, because the data is balanced this is also not relevant, and class weighting had no effect on the results.

Therefore, only different penalty norms are considered, using different regularization values. Specifically, the default regularization values of 1.0, with in addition values of 0.1, 10, 50, and 100 have been used. Class weighing was set on 'auto'.

Finally, a Multinomial Naïve Bayes classifier was used to predict the language labels of the texts. The Multinomial Naïve Bayes also possesses a regularization method, called Laplace smoothing. When a class and feature never occur together, the probability estimate is zero. Laplace smoothing solves this problem by giving a small-sample correction to the probability estimates. Similarly to the other classifiers, different smoothing values are used. Specifically we used the default smoothing value of 1.0, with in addition, 0.1, 0.5, 2.0, and 5.0.

Finally the results are compared to see, which of the classifiers is most accurate in the task, and which settings are most effective to perform the classification task. Furthermore, an analysis of the absolute coefficients has been performed to see which of the features contribute most to the predictions.

Error analysis

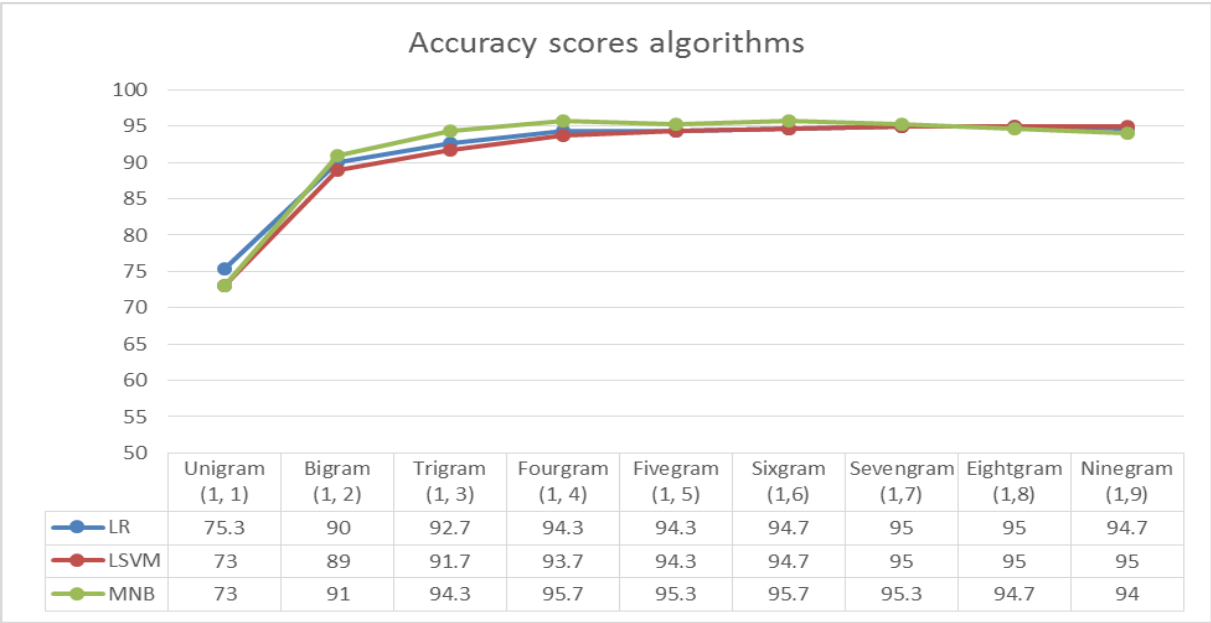
In addition to the comparison of the accuracy scores with the different settings, also an error analysis has been conducted. In the error analysis first, the misclassified tweets have been extracted by comparing the initial labels to the labels predicted by the model. The numbers that came out this comparison were sought in the dataset to find out why they could have been misclassified.

5. Results

As mentioned earlier, in the first step the accuracy score of the baseline was calculated on the test file (i.e. the development set). The result of the baseline gave an accuracy score of 53.3%. Then, the other models were subjected to the classification task using default settings.

In figure 1 the accuracy scores of the Logistic Regression (LR), Linear Support Vector Machine (LSVM) and Multinomial Naïve Bayes (MNB) are shown using default settings. In the default settings of the Logistic Regression and Linear Support Vector Machine class weighting is turned off, and regularization (C) is 1.0. Smaller values of C indicate higher regularization. Furthermore, the penalty norm is L2 for both classifiers, and the default strategy is the one versus the rest scheme for the Linear Support Vector Machine.

Furthermore, the Multinomial Naïve Bayes has a Laplace smoothing (α) parameter of 1.0. When α=0 no smoothing is applied, which indicates that higher values introduce higher smoothing. The accuracy scores are the percentages that the classifier has categorized correctly.



	Unigram (1, 1)	Bigram (1, 2)	Trigram (1, 3)	Fourgram (1, 4)	Fivegram (1, 5)	Sixgram (1,6)	Sevengram (1,7)	Eightgram (1,8)	Ninegram (1,9)
LR	75.3	90	92.7	94.3	94.3	94.7	95	95	94.7
LSVM	73	89	91.7	93.7	94.3	94.7	95	95	95
MNB	73	91	94.3	95.7	95.3	95.7	95.3	94.7	94

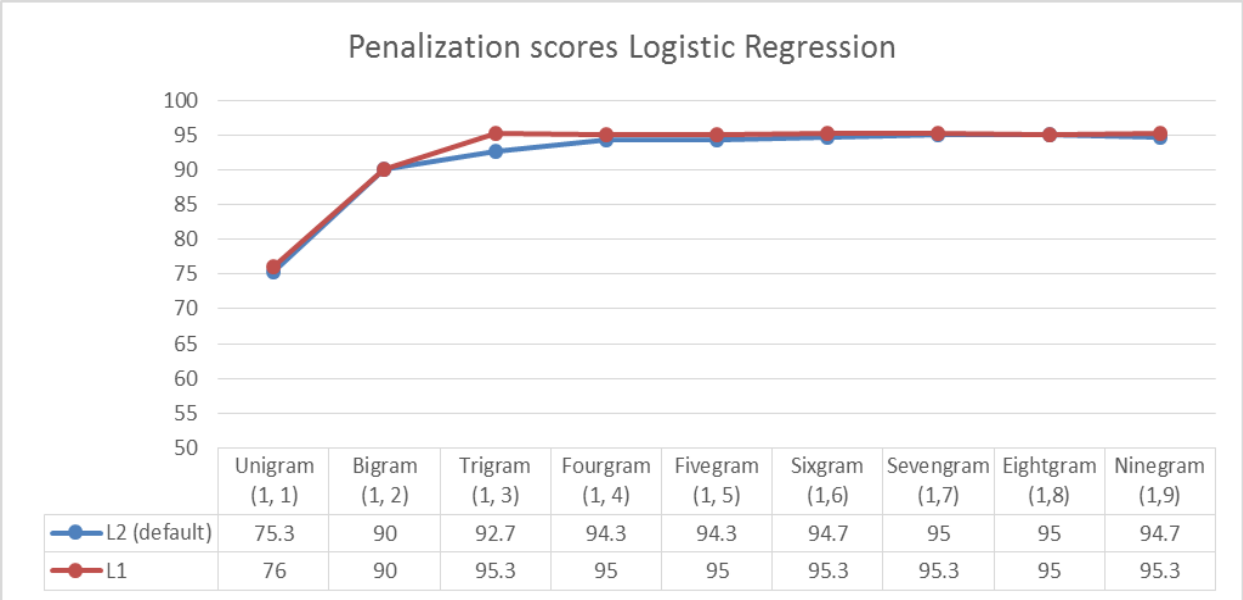
Figure 1: Results of the classifiers using default settings

The results show that while the n-gram lengths increase, the accuracy scores increase. At a certain length, the Linear Support Vector Machine finds a steady state, in which the accuracy score does not change anymore. The Multinomial Naïve Bayes achieves the best accuracy scores in comparison to the Logistic Regression and the Linear Support Vector Machine. The accuracy scores achieved are high, with a max of 95.7 percent. The Logistic Regression and Linear Support Vector Machine are very close to each other on all n-gram lengths. Only on the smaller lengths the Logistic Regression provides better predictions than the Linear Support Vector Machine.

5.1 Parameters of the Logistic Regression

Penalty

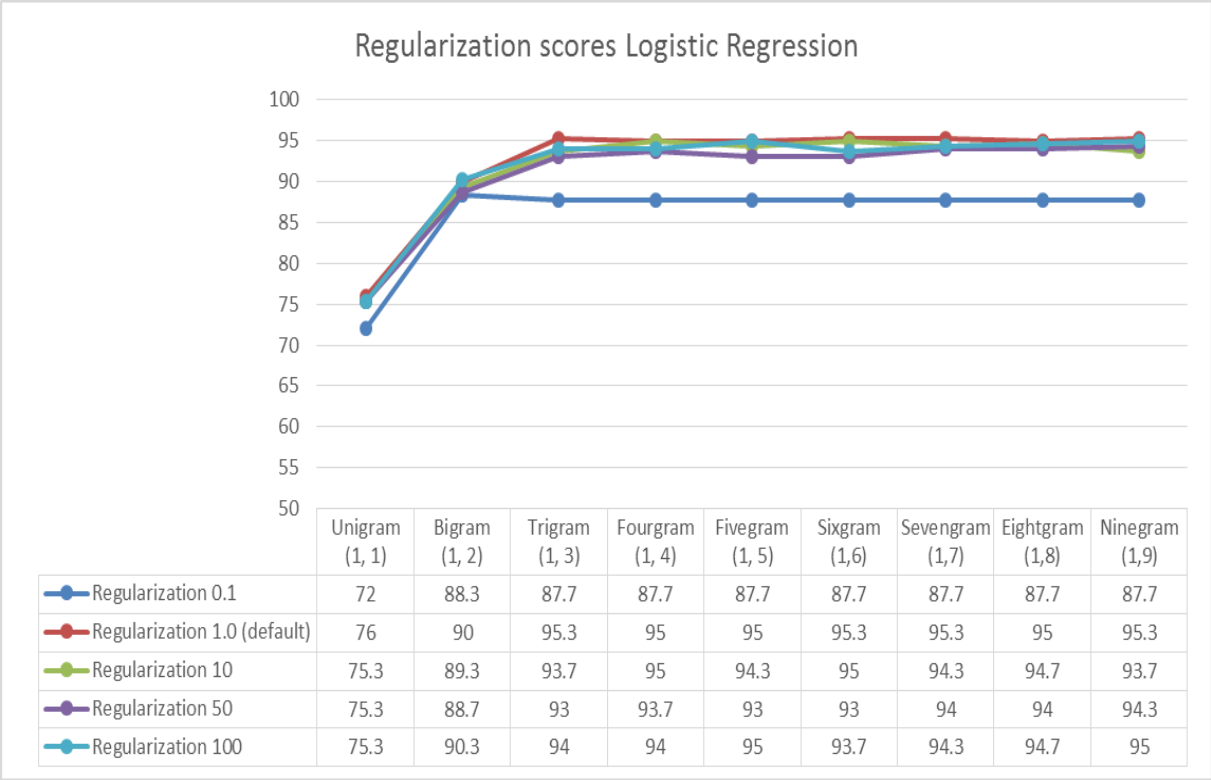
After the default classification was performed, the penalty parameter was first changed for the logistic regression. The results are shown in figure 2.



	Unigram (1, 1)	Bigram (1, 2)	Trigram (1, 3)	Fourgram (1, 4)	Fivegram (1, 5)	Sixgram (1,6)	Sevengram (1,7)	Eightgram (1,8)	Ninegram (1,9)
L2 (default)	75.3	90	92.7	94.3	94.3	94.7	95	95	94.7
L1	76	90	95.3	95	95	95.3	95.3	95	95.3

Figure 2: Results of the LR classification with the different penalization norms

The results show that the classification with penalty norm L1 improved the results. The results of the penalty norm L1 show that an accuracy score of 95 percent is achieved rapidly. Therefore, we only consider the different regularization values for the L1 norm (figure 3).



	Unigram (1, 1)	Bigram (1, 2)	Trigram (1, 3)	Fourgram (1, 4)	Fivegram (1, 5)	Sixgram (1,6)	Sevengram (1,7)	Eightgram (1,8)	Ninegram (1,9)
Regularization 0.1	72	88.3	87.7	87.7	87.7	87.7	87.7	87.7	87.7
Regularization 1.0 (default)	76	90	95.3	95	95	95.3	95.3	95	95.3
Regularization 10	75.3	89.3	93.7	95	94.3	95	94.3	94.7	93.7
Regularization 50	75.3	88.7	93	93.7	93	93	94	94	94.3
Regularization 100	75.3	90.3	94	94	95	93.7	94.3	94.7	95

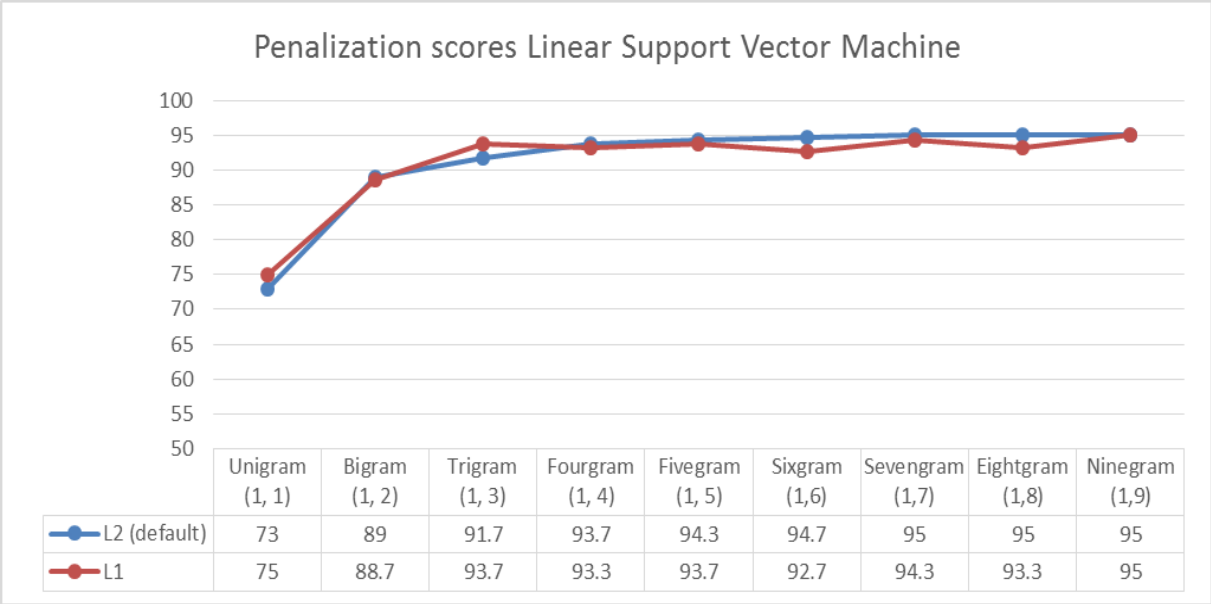
Figure 3: Results of the Logistic Regression with norm L2 and regularization

The results in figure 3 show that when the regularization value increases, and thus regularization decreases, no better accuracy scores are achieved. When regularization is 1.0, the classifier shows the best results, which is the default setting. The lowest regularization value shows the worst performance, and achieves a steady state very rapidly. Therefore, it is better to use less regularization (i.e. C=1.0).

5.2 Parameters of the Linear Support Vector Machine

Penalty

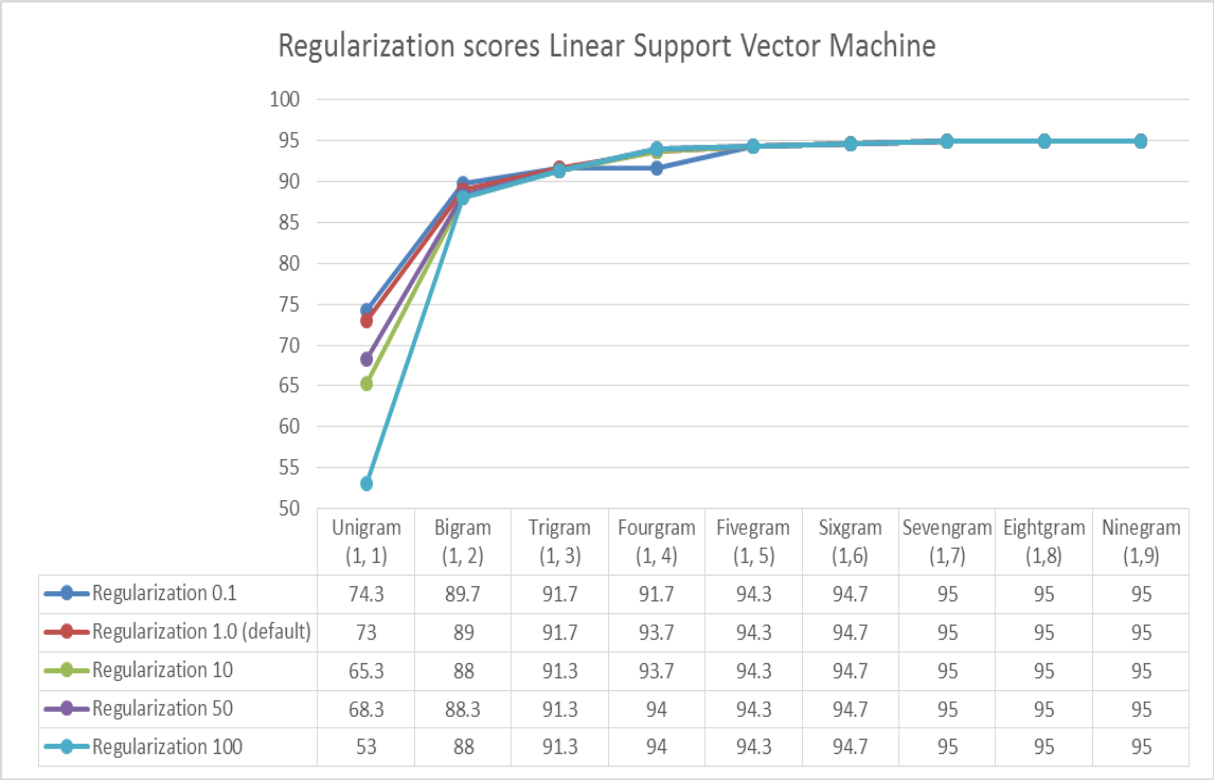
When changing the parameters of the Linear Support Vector Machine, first the penalty norm was changed similar to the Logistic Regression.



	Unigram (1, 1)	Bigram (1, 2)	Trigram (1, 3)	Fourgram (1, 4)	Fivegram (1, 5)	Sixgram (1,6)	Sevengram (1,7)	Eightgram (1,8)	Ninegram (1,9)
L2 (default)	73	89	91.7	93.7	94.3	94.7	95	95	95
L1	75	88.7	93.7	93.3	93.7	92.7	94.3	93.3	95

Figure 4: Results of the LSVM classification with the different penalization norms

The penalty L1 setting did not improve the results of the categorization task overall (figure 4). Only on trigrams the penalty L1 norm did improve the results. In comparison to the Logistic Regression, the Linear Support Vector Machine achieves better accuracy scores when the L2 norm is considered. The L2 norm finds a steady state, and does not fluctuate from this value anymore when n-gram lengths become larger. The L1 norm shows an unsteady performance, as it fluctuates over the different n-gram lengths. However, because the scores remain lower in comparison to the L2 norm, we will only consider regularization for the L2 norm (figure 5).



	Unigram (1, 1)	Bigram (1, 2)	Trigram (1, 3)	Fourgram (1, 4)	Fivegram (1, 5)	Sixgram (1,6)	Sevengram (1,7)	Eightgram (1,8)	Ninegram (1,9)
Regularization 0.1	74.3	89.7	91.7	91.7	94.3	94.7	95	95	95
Regularization 1.0 (default)	73	89	91.7	93.7	94.3	94.7	95	95	95
Regularization 10	65.3	88	91.3	93.7	94.3	94.7	95	95	95
Regularization 50	68.3	88.3	91.3	94	94.3	94.7	95	95	95
Regularization 100	53	88	91.3	94	94.3	94.7	95	95	95

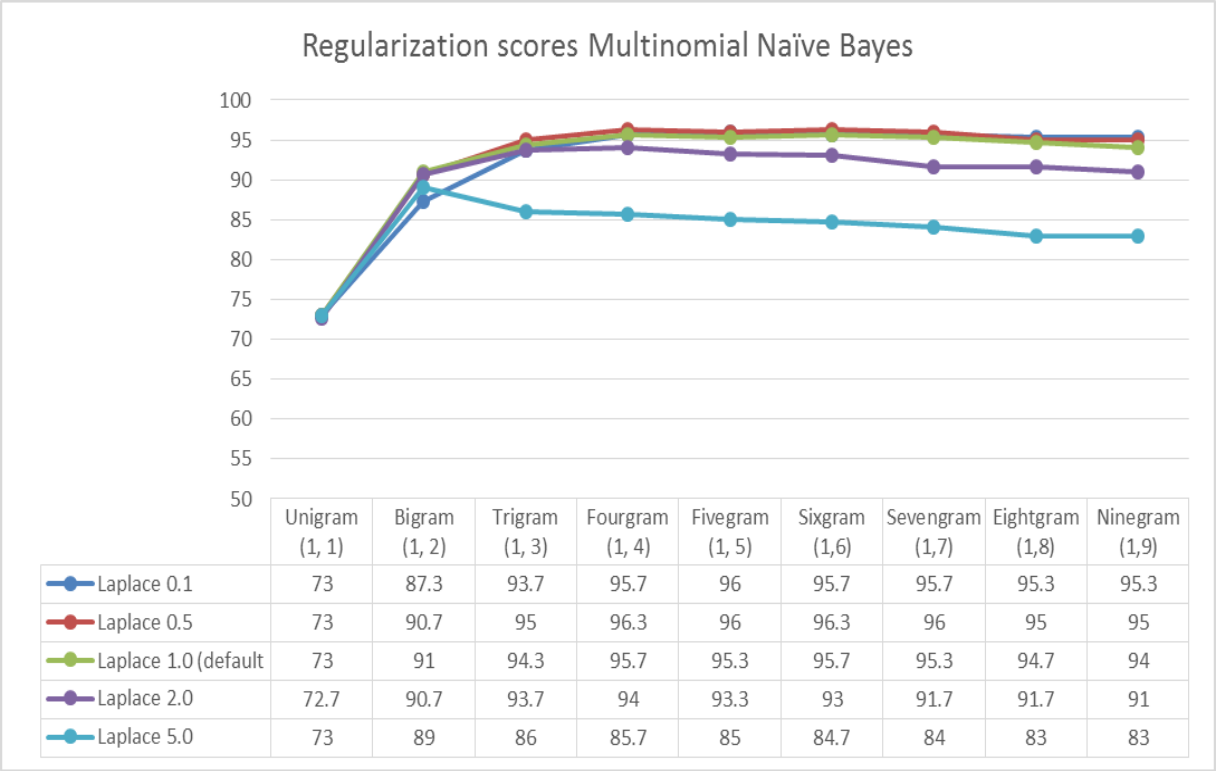
Figure 5: Results of the Linear Support Vector Machine with norm L2 and regularization

The results in figure 5 show that when the regularization parameter (C) increases and thus regularization decreases there is large decrease of the accuracy score for unigrams. When regularization is highest (i.e. C=0.1) the highest accuracy score is achieved on unigrams and bigrams. However, when C=1.0 a higher accuracy of two percent is achieved on fourgrams. Furthermore, when the n-gram length increases for all the lengths a steady state of 95 percent is achieved. Although, for the final classification the highest regularization setting will be used, as it achieves the best accuracy scores on unigrams and bigrams.

5.3 Parameters of the Multinomial Naïve Bayes

Laplace smoothing

In the final step Laplace smoothing has been introduced to the Multinomial Naïve Bayes. The results are shown in figure 6.



	Unigram (1, 1)	Bigram (1, 2)	Trigram (1, 3)	Fourgram (1, 4)	Fivegram (1, 5)	Sixgram (1,6)	Sevengram (1,7)	Eightgram (1,8)	Ninegram (1,9)
Laplace 0.1	73	87.3	93.7	95.7	96	95.7	95.7	95.3	95.3
Laplace 0.5	73	90.7	95	96.3	96	96.3	96	95	95
Laplace 1.0 (default	73	91	94.3	95.7	95.3	95.7	95.3	94.7	94
Laplace 2.0	72.7	90.7	93.7	94	93.3	93	91.7	91.7	91
Laplace 5.0	73	89	86	85.7	85	84.7	84	83	83

Figure 6: Results of the Multinomial Naïve Bayes with different Laplace smoothing values

The result in figure 6 show that the Multinomial Naïve Bayes performs best when smoothing is small. When smoothing is 0.5 the highest accuracy scores are achieved. Although the differences are small in comparison to a smoothing value of 0.1. When smoothing is high (i.e. α=5.0), the accuracy scores achieved are worst in comparison to the other smoothing values.

5.4 Summary of parameters changes

The classifiers contain different parameters of which some have been adjusted. With adding these different settings, we obtained different effects.

First, the penalty norm of the Logistic Regression was adjusted. The L1 norm performed best, while the L2 norm led to lower accuracy scores on the predictions. Therefore, for the final test, the L1 norm will be used for the prediction.

Secondly, different regularization values were used for the prediction. It seemed that the default setting performed best (i.e. C=1.0). Furthermore, high regularization gave

substantial lower accuracy scores in comparison to the default setting, and lower regularization than the default setting also gave lower accuracy scores. Therefore, the regularization parameter will be set on 1.0 for the final test.

Furthermore, the same steps have been performed for the Linear Support Vector Machine. The penalty parameter was adjusted from the default L2 norm to the L1 norm, and different regularizations were used for the prediction. Contrary to the Logistic Regression the L1 norm did not lead to higher accuracy scores in comparison to the L2 norm. In addition, regularization led to a large increase on unigrams, but not to large increases on other lengths. The Linear Support Vector Machine found a steady state for all n-gram lengths after the length of four, and did not differ from this value anymore. However overall, when regularization was highest (i.e. C=0.1) the best accuracy scores were achieved. Therefore, this setting will be used for the final classification.

When the Multinomial Naïve Bayes was considered, different Laplace smoothing values were used to improve accuracy. In this case a value of 0.5 performed best, and therefore this setting will be used for the final classification.

5.5 Most important features

As mentioned earlier, an analysis on the absolute coefficients has been performed, to determine which features contribute most to the predictions. The most important features have been taken randomly from one of the classifications of the Logistic Regression, which achieved an accuracy of 95 percent. In figure 7 below, the 20 most important features are shown for the classification of the Limburgian tweets, and the 20 most important features are shown for the classification of the Dutch tweets. The bars represent the coefficients for each feature.

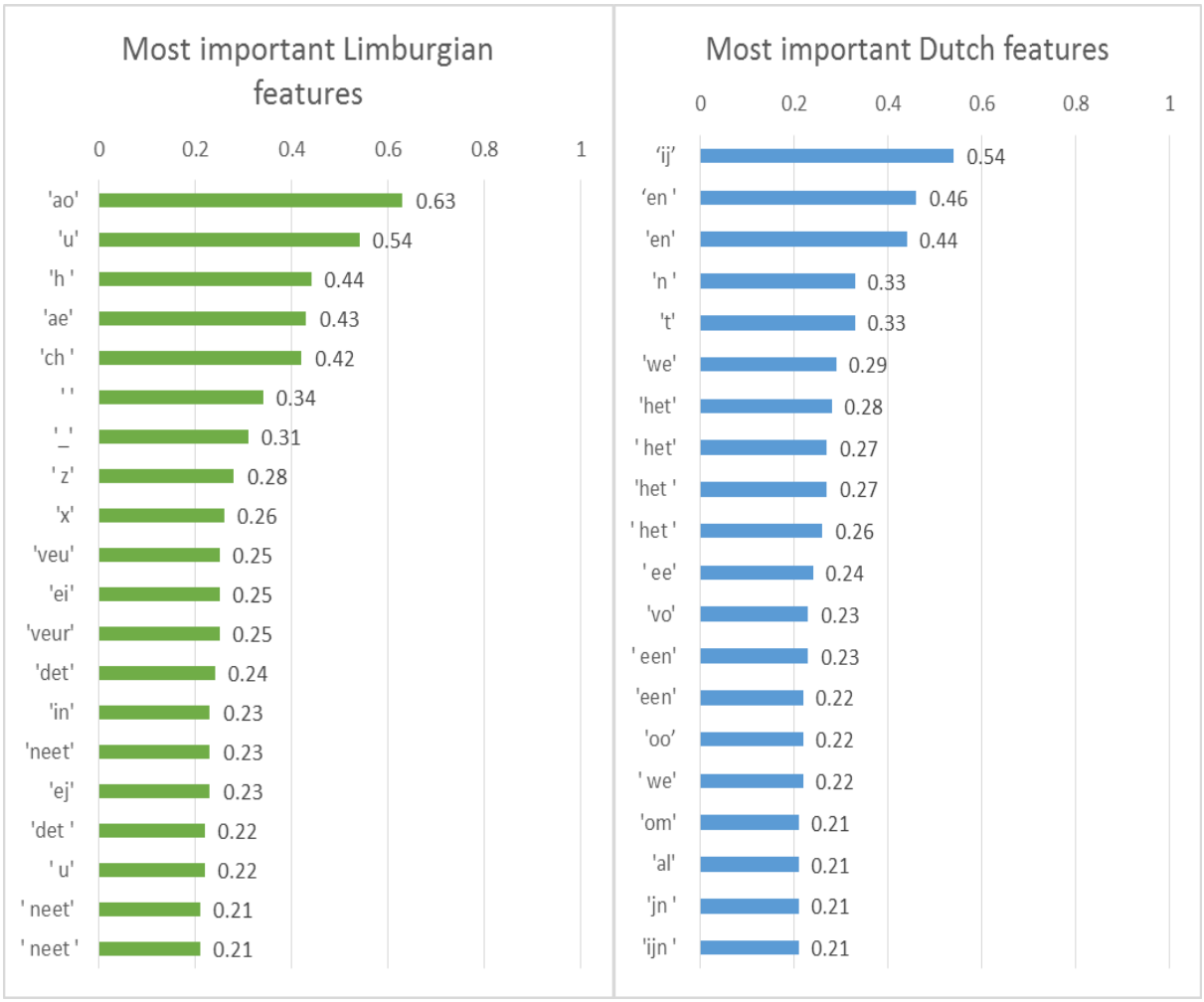


Figure 7: Top twenty most important features of the classification of Dutch and Limburgian tweets

When the most important features are studied, it is possible to conclude several things from these features. First, the most important feature for the Limburgian tweets is ‘ao’, which is common for the Limburgian language. For example, the words ‘jao’ (yes), ‘dao’ (there), ‘straot’ (street), ‘bienao’ (almost), ‘hienao’ (after this), and many more use this combination. Secondly, it is possible to say the same about ‘ae’, which is used in words such as ‘nae’ (no), ‘dae’ (him), ‘kaerel’ (man), ‘vlaegel’ (bully), ‘aek’ (vinegar), and more. Another important feature is ‘ch’, which is also commonly used. As discussed in the background section ‘ch’ is used commonly after vowels south of the Uerdinger line, and also for this feature it is possible to give some examples, such as ‘ich’ (I), ‘mich’ (me), ‘dich’ (you), and many more.

When the lesser important features are considered, there are also a couple of interesting features among them. For example, ‘veur’, which refers to different things. That is, ‘we’, ‘in front’, or ‘for’ when translated to English. Furthermore, ‘det’ refers to ‘that’, and ‘neet’ refers to ‘not’. Finally, ‘ei’ is also commonly used in Limburgian words. For example, ‘steit’ (stands), ‘geit’ (going to), bleik (pale), ‘dreij’ (turn) and more.

The most important feature of the classification of the Dutch tweets is ‘ij’. This combination is commonly used in words such as ‘wij’ (we), ‘zij’ (she), ‘blij’ (happy), ‘tijd’ (time), and many more. The second most important feature is ‘en’, which refers to ‘and’. Furthermore, ‘het’ (it), ‘we’ (we), ‘een’ (a), ‘om’ (at), and ‘al’ (already) are commonly used in Dutch tweets. Several of these combinations can also be part of other words. For example, the feature ‘we’ can also be part of ‘welke’ (which), or ‘al’ can also be part of ‘alles’ (everything) or ‘alleen’ (alone). Therefore, they are probably important features. Finally, the combination ‘vo’ and ‘oo’ are important, which possibly refers to words such as ‘ook’ (also), ‘voor’ (for), ‘door’ (by or through), and more.

Comparing these results to each other it is noticed that there is a similarity in the common features for the Limburgian and Dutch languages, namely the word for ‘we’ is one of the most important features in both languages, ‘veur’ in Limburgian and ‘we’ or ‘wij’ in Dutch.

5.6 Misclassifications

Finally we performed an analysis on the misclassified tweets. The misclassified tweets have been taken from the setting that achieved the highest accuracy score, namely the Multinomial Naïve Bayes with a smoothing value of 0.5 and an n-gram length of four. These misclassified tweets have been taken, because these were also in the classifications with lower accuracy

scores, and therefore occurred most. The tweets and their numbers are shown in table 1 below.

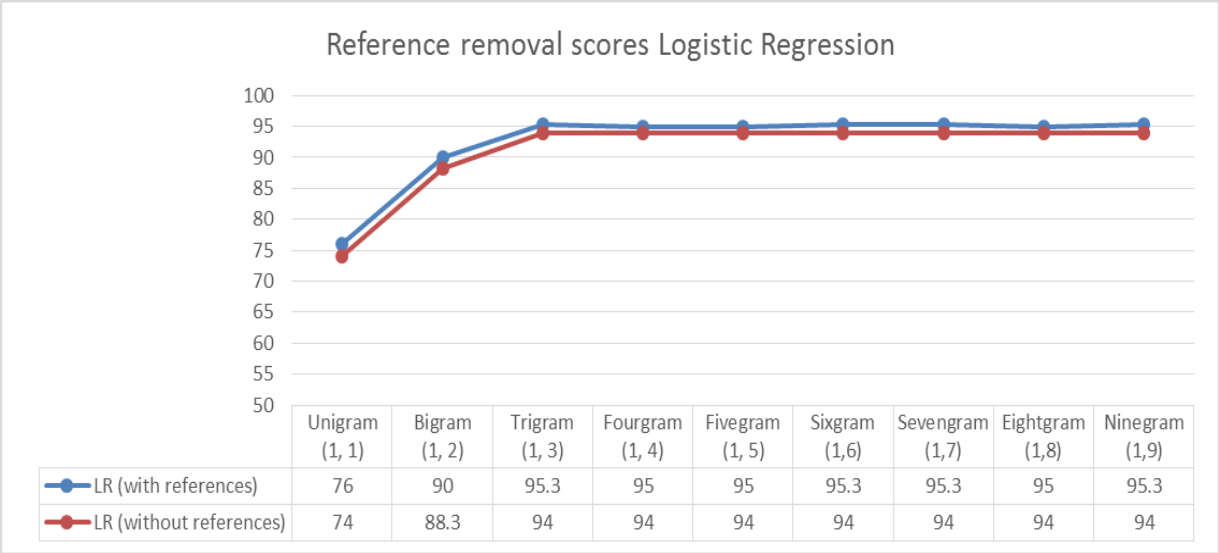
TweetNR	TweetText	True label	Classified label
1	@BernardGepken Bedaankt veur je RT, Bernard! :-) #OpStreek	li	nl
42	@LinksVlaams wa's da na wÃ¨r veur 'n toaltje? :o) :o) @EddyVanDaele59	li	nl
63	@MickWaajen du kins toch zo lekker kaoke!ðŸ˜‰ (ich weit waatse zou's aete.) en angers bel lekker veur win pizzaatje! Sjmakelik in eeder geval.ðŸ˜˜	li	nl
68	@MaudVanDijkk hiel vÃ¤Ã¶l plezeer veur uuch alle veer. Geneet d'r vÃ¢n!	li	nl
93	@jansenantonius dat is get anges es de koempels oet kirchroa	li	nl
101	morgevruug vertreke weej weer um 9 oor op de parade	li	nl
124	Lekker weekend gehad, auk al hebben we now vekansie... Toch maar ens mien hoeshald lieske aafgaon ðŸ˜Š	li	nl
129	En dit is woavuur datste ut dees! Wauw, wat han ich genoate.. WÃ¡t ing premiÃ¨re, wÃ¡t ee sjun publiek. Teamwork op,... http://t.co/Qtpn0059yk	li	nl
130	Altied weer det brandalarm as weej pauze hebben. Um gek van te waere.	li	nl
134	Poeh poeh! Bietje te gezellig gister!	li	nl
140	Effe in Mestreech , rootsgevoel http://t.co/WUA9wy2jqh	li	nl

Table 1: Misclassified tweets

The misclassified tweets vary among each other, as some of them are just plain text, and some contain references to names or places referred to by hashtags, @ signs, and links. Furthermore, it is noticed that some of the tweets have some errors in them, of which some of have been caused during the retrieval of the tweets.

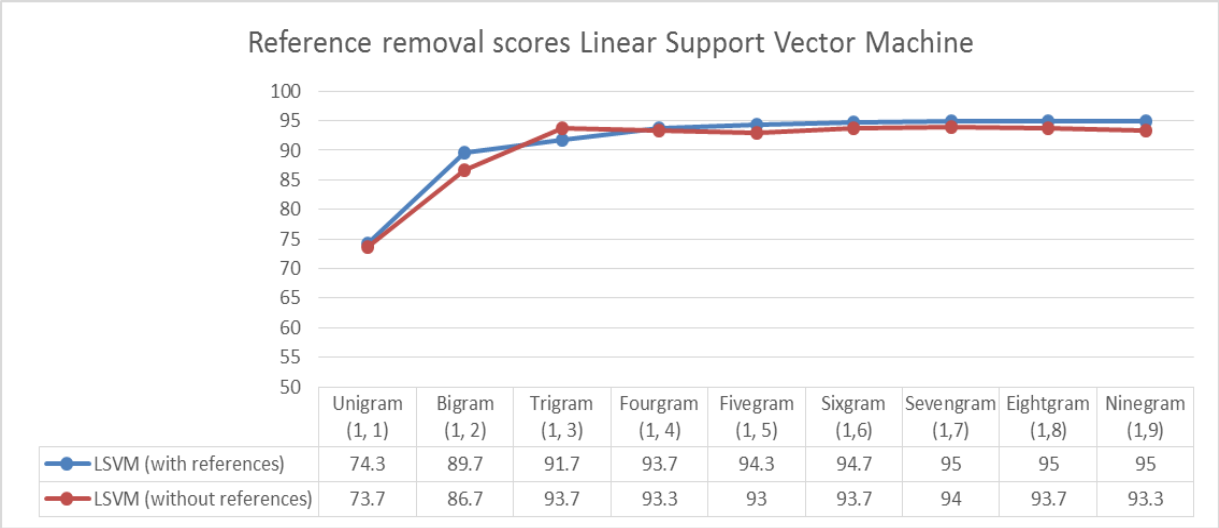
When we search for a pattern, it is noticed, that all of the misclassified tweets are Limburgian, and that the algorithm classified these tweets as Dutch instead. It is unknown why these tweets are classified wrong. However, it is possible that it is because of the references, which are present in the text of the tweets. Therefore, the pieces of text referred to by links, hashtags, and @signs have been removed from the tweet to see if this leads to any improvements. In general, extra preprocessing steps have been added to the code to achieve this.

To test if there is an improvement, for all the classifiers the best configuration has been taken and run again, only without references. The results are shown in the figures 8, 9, and 10. The references have been removed in both the training and test data.



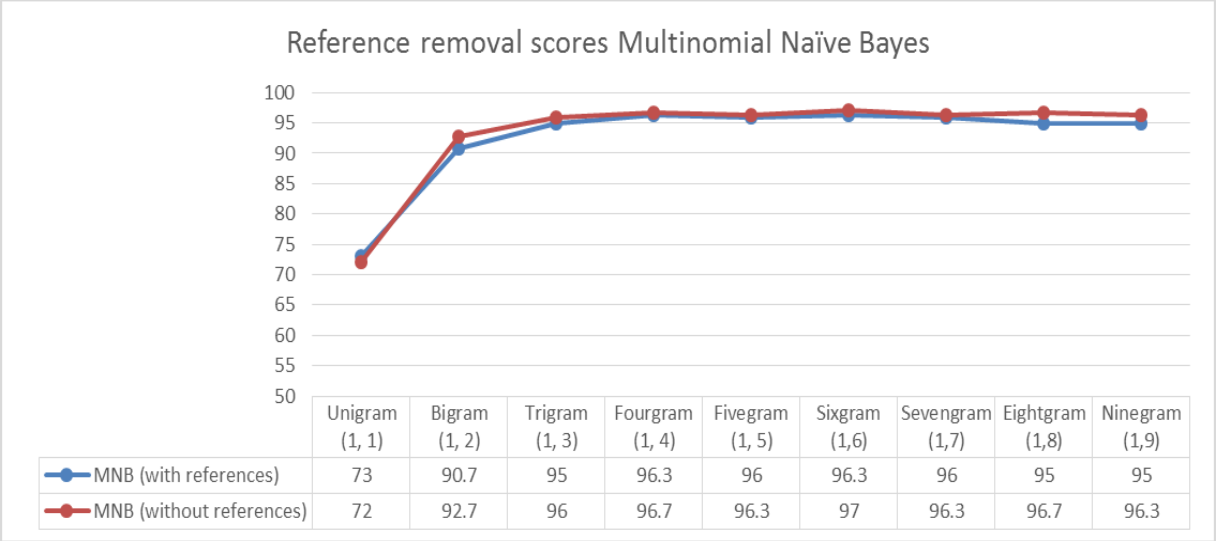
	Unigram (1, 1)	Bigram (1, 2)	Trigram (1, 3)	Fourgram (1, 4)	Fivegram (1, 5)	Sixgram (1,6)	Sevengram (1,7)	Eightgram (1,8)	Ninegram (1,9)
LR (with references)	76	90	95.3	95	95	95.3	95.3	95	95.3
LR (without references)	74	88.3	94	94	94	94	94	94	94

Figure 8: Results of the Logistic Regression with and without references



	Unigram (1, 1)	Bigram (1, 2)	Trigram (1, 3)	Fourgram (1, 4)	Fivegram (1, 5)	Sixgram (1,6)	Sevengram (1,7)	Eightgram (1,8)	Ninegram (1,9)
LSVM (with references)	74.3	89.7	91.7	93.7	94.3	94.7	95	95	95
LSVM (without references)	73.7	86.7	93.7	93.3	93	93.7	94	93.7	93.3

Figure 9: Results of the Linear Support Vector Machine with and without references



	Unigram (1, 1)	Bigram (1, 2)	Trigram (1, 3)	Fourgram (1, 4)	Fivegram (1, 5)	Sixgram (1,6)	Sevengram (1,7)	Eightgram (1,8)	Ninegram (1,9)
MNB (with references)	73	90.7	95	96.3	96	96.3	96	95	95
MNB (without references)	72	92.7	96	96.7	96.3	97	96.3	96.7	96.3

Figure 10: Results of the Multinomial Naïve Bayes with and without references

The results in figures 8, 9, and 10 show that removing the references in the text indicated by hashtags, @ signs, and links, improved the results of the classification of the Multinomial Naïve Bayes. However, the results show that there is no improvement in the results of the Linear Support Vector Machine and Logistic Regression. Therefore, hashtags, links and references indicated by @ signs will be removed only for the Multinomial Naïve Bayes in the final test.

5.7 Final test

In figure 11, the results of the final classification are shown for the final test set. The baseline achieved an accuracy score of 53.7 percent on the test data.

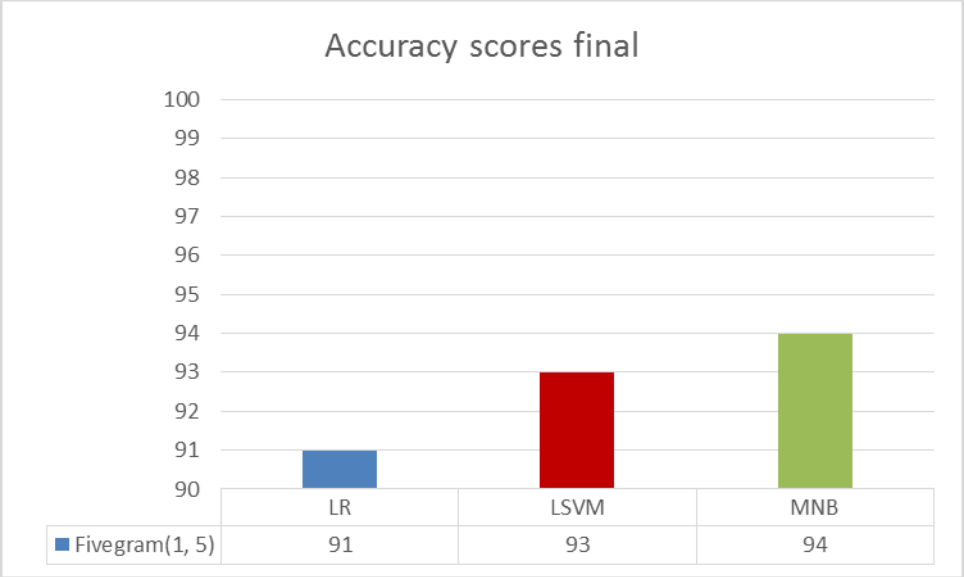


Figure 11: Results of the classifiers on the final test set

The results show that on fivegrams, the best accuracy scores are achieved. The Logistic Regression achieves a steady state from this length, which means that the accuracy score stays the same in larger lengths. For the other classifiers a decay occurs when the n-gram length becomes larger than five. In figure 10 is shown that the Multinomial Naïve Bayes classifier achieves the highest accuracy score, namely an accuracy of 94 percent. In general, all the classifiers were able to achieve a score above 90 percent, which shows that they outperform the baseline. Therefore, all models are good predictors of the Limburgian dialect.

6. Conclusion and Discussion

In this paper a method is presented to detect the Limburgian dialect from Twitter messages based on a dataset collected by using the Twitter API. To recognize whether tweets were Limburgian or Dutch, several classification algorithms have been used.

Several experiments have been conducted using three different classifiers. That is, a Logistic Regression, Linear Support Vector Machine, and a Multinomial Naïve Bayes classifier. It was expected that the identification of the Limburgian dialect would be a difficult task, because of several reasons mentioned earlier in the background section. The most important reasons are, that dialect identification is difficult because it is applied to a closely related group of languages, which have similar spelling conventions (Zaidan and Callison-Burch, 2012). Furthermore, dialect identification is a rather new area of research, and most research conducted is focused on the identification of Arabic dialects. For example, in the researches from Zaidan and Callison-Burch (2012), Biadsy et al. (2009), and Sadar, Kazermi and Farzindar (2014). In addition, twitter messages contain misspellings, errors and have different stylings.

Although it seemed that the detection of Limburgian would be a difficult task. The results from the classifiers show high performance on the classification task, and were all able to achieve a score above 90 percent. All the models performed better than the baseline model. The Multinomial Naïve Bayes classifier performed best, while overall the Logistic Regression performed worst. However, in general the classifiers all achieved a high accuracy of above 90 percent, with a peak of 94 percent for the Multinomial Naïve Bayes. Furthermore, the best results were obtained on fivegrams for all the classifiers.

To achieve this percentage, several parameters have been used to optimize the performance. For example, the penalty norm has been adjusted and different regularization

parameters have been used to find the best settings. The L2 penalization norm with a tuned C parameter of 0.1 improved the results of the Linear Support Vector Machine most. The L1 penalization norm with a tuned C parameter of 1.0 improved the results of the Logistic Regression most. Furthermore, in the Multinomial Naïve Bayes we used different Laplace smoothing values to find the optimal model. A Laplace smoothing parameter of 0.5 improved the results best.

The analysis of the misclassified tweets did not lead to improvements for the Logistic Regression and Linear Support Vector Machine. By adjusting the preprocessor, so that it also removed references to names, places and links, led to worse accuracy scores. Therefore, the system only received texts which were made lower case and without punctuation, as was initially intended. However, it did lead to improvement for the Multinomial Naïve Bayes, and therefore the adjustments in the preprocessor were used in the final classification for the Multinomial Naïve Bayes.

Another notification in the error analysis was that there were also errors in the tweets. For future research, it would be interesting to remove these errors. Already in the background section it became clear that tweets contain misspellings, errors and different styles. Therefore, in future research it is possible to study the effect of removing the errors from the text on the accuracy score.

The goal of this study was to build a system that could successfully detect the Limburgian language from Twitter messages. The conclusion is that the goal is achieved, because an accuracy score of 94 percent was achieved. However, it was also the case that this accuracy score was achieved on a relatively small dataset, and this dataset was balanced. Therefore, in future research it would be interesting to perform the same task on larger unbalanced datasets. This larger corpus can be collected by extracting tweets over a longer

period. Furthermore, it is possible to use methods such as crowdsourcing, by asking the public to provide Limburgian texts. Another approach could be, to collect messages from Facebook.

Finally, in the background section it was mentioned that Limburgian is the common name for a diversity of dialects which are found in the Limburgian area. These diversities can be divided into five language areas (i.e. isoglosses). In future research, it is possible to look for the differences between these language areas, and try to identify the areas based on this data.

References

Britannica. (2010).West Germanic languages. Retrieved from http://www.britannica.com/EBchecked/topic/640154/West-Germanic-languages/74785/History

Bergsma, S., McNamee, P., Bagdouri, M., Fink, C. & Wilson, T. (2012). Language Identification for Creating Language-Specific Twitter Collections. *Proceedings of the 2012 Workshop on Language in Social Media* (pp. 65-74). Stroudsburg: PA.

Biadsy, F., Hirschberg, J. & Habash, N. (2009). Spoken Arabic Dialect Identification Using Phonotactic Modeling. *Proceedings of the EACL 2009 Workshop on Computational Approaches to Semitic Languages* (pp. 53-61). New York: University of Colombia.

Cavnar, W. B. & Trenkle, J. M. (1994). N-Gram-Based Text Categorization. *Ann Arbor MI*, 2, 161-175.

Driessen, G. (2008). In Dutch? Usage of Dutch Regional Languages and Dialects. *Culture and Curriculum*, 18(3), 271-285. doi: 10.1080/07908310508668747

Field, A. (2013). *Discovering Statistics using IBM SPSS Statisics,* London, England: Sage.

Gottron, T. & Lipka, N. (2010). A Comparison of Language Identification Approaches on Short, Query-Style Texts. *LNCS, 5993,* 611-614. 10.1007/978-3-642-12275-0_59

Heeringa, W. (2007). Een andere indeling van de Limburgse dialecten. *Veldeke Jaarboek 2007, 94–10.*

Herten, van, B. (2007). *Automatic Opinion Extraction and Classification from Text* (Doctoral dissertation). Retrieved from https://doclib.uhasselt.be/dspace/bitstream/1942/3731/1/van-hertum.pdf

Houvardas, J. & Stamatatos, J. (2006). N-Gram Feature Selection for Authorship Identification. *Artificial Intelligence: Methodology, Systems, and Applications Lecture Notes in Computer Science, 4183,* 77-86. doi: 10.1007/11861461_10

Huang, A. (2008). Similarity Measures for Text Document Clustering. *Proceedings of the New Zealand Computer Science Research Student Conference 2008* (pp. 49-56). New Zealand: University of Waikato.

Huberman, B. A., Romero, D. M. & Fang, W. (2008). Social networks that matter: Twitter under the microscope. *First Monday, 14,* 1-5.

Hughes, D. J., Rowe, M., Batey, M. & Lee, A. (2012). A tale of two sites: Twitter vs. Facebook and the personality predictors of social media usage. *Computers in Human Behavior,* 28(2), 561-569.

Joachims, T. (2005). Text categorization with Support Vector Machines: Learning with many relevant features. *10th European Conference on Machine Learning* (pp. 137-142). doi: 10.1007/BFb0026683

McCallum, A. & Nigam, K. (1998). A Comparison of Event Models for Naïve Bayes Text Classification. *AAAI-98 workshop on learning for text categorization, 752*, 41-48.

NU. (2014). Twitter groeit naar 284 miljoen actieve gebruikers. Retrieved from http://www.nu.nl/internet/3913892/twitter-groeit-284-miljoen-actieve-gebruikers.html

Ljubešić, N., Mikelić, N. & Boras, D. (2007). Language Identification: How to Distinguish Similar Languages? *Information Technology Interfaces*, 2007, 541-546. doi: 10.1109/ITI.2007.4283829

Peters, J. (2007). Bitonal lexical pitch accents in the Limburgian dialect of Borgloon. *Tones and tunes.* Retrieved from http://scholar.googleusercontent.com/scholar?q=cache:zON3NTqJPS0J:scholar.google.com/+bitonal+lexical+pitch+accents&hl=nl&as_sdt=0,5

Rogati, M. & Yang, Y. (2002). High-performing feature selection for text classification. *Proceedings of the eleventh international conference on Information and knowledge management* (pp. 659-661). doi: 10.1145/584792.584911

Sadat, F., Kazemi, F., Farzindar, A. (2014). Automatic identification of Arabic dialects in social media. *Proceedings of the first international workshop on social media retrieval and analysis* (pp. 35-40). New York: NY.

Scott, S., & Matwin, S. (1999). Feature engineering for text classification. In M. Kaufmann (Ed.), *Proceedings of the Sixteenth International Conference on Machine Learning* (379-388). San Franciso: CA.

Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H., Demirbas, M. (2010). Short text classification in twitter to improve information filtering. *Proceedings of the 33rd international ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 841-842). doi:10.1145/1835449.1835643

Swanenberg, J. (2013). *All dialects are equal, but some dialects are more equal than others* (Tilburg Papers in Culture Studies Paper No. 43).

Tratz, S. C. (2014). *Accurate Arabic Script Language/Dialect Classification* (Report No. ARL-TR-6761). Retrieved from http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA597898

Ubachs, P. J. H. (2000). *Handboek voor de geschiedenis van Limburg.* Retrieved from http://books.google.nl/books?hl=nl&lr=&id=KCg_3mxdsZoC&oi=fnd&pg=PA5&dq=geschiedenis+limburg&ots=Cui6o0dC-f&sig=a3AmT8fclvS2gtToQejbUveZeSk#v=onepage&q=geschiedenis%20limburg&f=false

Zaidan, O. F., Callison-Burch, C. (2011). The Arabic Online Commentary Dataset: An annotated dataset of informal Arabic with high dialectal content. *Proceedings of ACL* (pp.37-41). Baltimore: USA.

Zaidan, O. F., Callison-Burch, C. (2012). Arabic dialect identification. *Computational*

Linguistics, 40(1), 171-202.

Appendices

Appendice 1: Search query

kump Kump zeuke Zeuke vernuujd Vernuujd gehalde Gehalde bliej Bliej um Um gaef Gaef se Semer Mer auk Auk weure Weure pien Pien Ouch ouch bietje Bietje wuurd Wuurd wieert Wieert un Un ut Ut ein Ein dae Dae kiek Kiek der Dernae Nae nea Nea sjaon Sjaon vasteloavend Vasteloavend zoervleisj Zoervleisj noe Noe rieje Rieje eate Eate aete Aete kin Kin Zweigel zweige Gaatsnieer gaatsnieer Erpel erpel petat Petat erbere Erbere loester Loester good Good ummer Ummer appelmoos Appelmoos erm Erm fratz Fratz Aovundj aovundj aek Aek balkebrie Balkebrie bin Bin snapst Snapst bein Bein Bessem bessum beer Beer bie Bie bisjop Bisjop blome Blome book Book bloomkoel Bloomkoel Telder telder Teljer teljer teller Teller knoak Knoak broabandj Broabandj breie breie boks Boks mik Mik bruedje Bruedje Brutje brutje illik Illik vasteloavundj vasteloavundj daag Daag meinste Meinste dinken Dinken sjottelslet Sjottelslet durchslaag Durchslaag meinen Meinen doch Doch haije Haije doon Doon doar Doar pratsj Pratsj vaam Vaam doorsjpeule Doorsjpeule doem Doem Pruusse pruusse dauwe Dauwe Ech Ech Pekske pekske knien Knien knijn Knijn wiekser Wiekser Zwaegel zwaegel ummer Ummer eazel Eazel gank Gank goan Goan vazel Vazel geboare Geboare gedoan Gedoan geduns Geduns guf Guf geaf Geaf gelaen Gelaen rechop Rechop gevreat Gevreat gesjreeve Gesjreeve geweun Geweun sjtiev Stjiev meadje Meadje diech Diech glaas Glaas dich Dich goojendaag Goojendaag Groeate groeate veut Veut geit Geit sjink Sjink Sjonk sjonk handj Handj heng Heng henj Henj handjsjoon handjsjoon hel Hel hoare Hoare hubs Hubs hubse Hubse Haerle haerle kloar Kloar klaer Or Klaer helemoal Halemoal book Book goot Goot haemdj Haemdj leef Leef heukske Heukske Gebrook gebrook wiezoe Wiezoe det Det vleis Vleis hoes Hoes ederein Ederein iech Or Iech gewuuen Or Gewuuen gou Gou doesje Doesje laupend Laupend Uch uch hauw Hauw

verstoan Verstoan versting Versting verstang Verstang Doorein doorein genoame Genoame zeen Zeen joar Joar joare Joare neet Neet doe Doe bis Bis wies Wies gej Gej krank Krank diene Diene mien Mien verstuis Verstuis versjteis Verstjeis wichter Wichter geer Geer uch Uch Gear gear kleid Kleid gear Gear kees Kees zonger Zonger sjroet Sjroet klidje Klidje kieke Kieke kindj Kindj Kleijer kleijer knakwosj Knakwosj knoevel Knoevel kumse Kumse hiej Hiej keunigin Keunigin keunig Keunig teske Teske tas Tas kriet Kriet kruuts Kruuts vlaai Vlaai kierse Kierse laaj Laaj lemmele Lemmele leaf Leaf leeve Leeve Sjweagele sjweagele loester Loester Mesjtreech mesjtreech Maestrich maestrich Mestreeg mestreeg mier Mier Sies sies Sieske sieske Millek millek miech Miech sjoen Sjoen chique Chique mooder Mooder moes Moes nutte Nutte puupke Puupke naas Naas vreate Vreate geluive Geluive Zeiver zeiver Sjloefe sjloefe piese Piese spass Spass sjiete Sjiete plezeer Plezeer sokker Sokker kale Kalle gevreat Gevreat Ruiver ruiver rauke Rauke vlemme Vlemme Poor poor sjoap Sjoap Sjloap sjloap Schatteke schatteke sjoon Sjoon zweitnek Zweitnek speegel Speegel Sjproete sjproete sjtoan Sjtoan ongerboks Ongerboks schmiecht Schmiecht batsen Batsen Stroat stroat kruutje Kruutje zeem Zeem krevat Krevat sjlieps Sjlieps Zjwame zjwame toafel Toafel buus Buus those Thoes tied Tied heem Heem troan Troan adieje Adieje shotelsplak Shotelsplak ziej Ziej verken Verken hendig Hendig sjlum Sljum doarsjlaag Doarsjlaag ooi Ooi sjrikkelijk Sjrikkelijk vla Vla vleeg Vleeg voot Voot gieps Gieps vrunjtelijk Vrunjtelijk vrunj Vrunj vrachwage Vrachwage voot Voot woarum Woarum Wiezoe wiezoe Woor woor et Et dien Dien vriedig Vriedig sjink Sjink deis Deis watblief Watblief wear Wear weit Weit gedoon Gedoon moeremoos Moeremoos zakdook Zakdook wilkom Wilkom kniep Kniep tess Tess germ Germ tuutje Tuutje zekske Zekske verstingen Verstingen zalt Zalt meistal Meistal zeen Zeen aaf Aaf aafbrenne

Aafbrenne aafdaak Aafdaak aafdeiling Aafdeiling aafdrage Aafdrage aafblieve Aafblieve aafbloaze Aafbloaze aafbringe Aafbringe aafdaak Aafdaak aangerich Aangerich Aafgesjpraoke aafgesjpraoke aafhoale Aafhoale Aafhanjele aafhanjele aafliere Aafliere aafraekening Aafraekening aafrichte Aafrichte aafsjpeule Aafsjpeule aafsjrieve Aafsjrieve aafsjroeve Aafsjroeve aafteikene Aafteikene aaftraeje Aaftraeje Aanbaeje aanbaeje aanbetale Aanbetale aanbevaele Aanbevaele aanbelle Aanbelle Aanboew aanboew aanblieve Aanblieve aanbringe Aanbringe aanbrenne Aanbrenne aandeil Aandeil aanein Aanein aangaeve Aangaeve aankleije Aankleije aanknuipe Aanknuipe aankieke Aankieke Aankoup aankoup aanmeudige Aanmeudige aanmelje Aanmelje aanmirke Aanmirke aanranje Aanranje aanrech Aanrech aanpaote Aanpaote aansjlaag Aansjlaag aansjtaeke Aansjtaeke aansjpraok Aansjpraok Aansjtaon aansjtaon aanteikene Aanteikene aansjuve Aansjuve aansjtriepe Aansjtriepe aanvlege Or Aanvlege aanvraoge Aanvraoge Achterkantj achterkantj achteroet Achteroet adjeu Adjeu merci Merci adjus Adjus aek Aek aerpel Aerpel aevekes Aevekes aetlaepel Aetlaepel albaer Albaer Algemein algemein alik Alik Woonsdig woonsdig dinsdaag Dinsdaag donderdig Donderdig Maondig maondig angers Angers Angere angere antik Antik antwaord Antwaord Aojer Or aojer aolie Aolie aom Aom aud Aud sjool Sjool autowaeg Autowaeg auwer Auwer auwluuj Auwluuj baek Baek Baer Or baer baeter Or Baeter bajer Bajer Baog baog baove Baove onger Onger bazel Bazel belsj Belsj benj Benj benuujd Benuujd benzien Benzien benzienpomp Benzienpomp besjikbaar Besjikbaar besjildere Or Besjildere besjlaag Besjlaag besjloet Besjloet besjpare Besjpare besjpringe Besjpringe besjpele Besjpele Besjpraeke besjpraeke besjrieve Besjrieve besjpritse Besjpritse besjtelle Besjtelle besjterve Besjterve besjtaon Besjtaon besjtudere Besjtudere besjuut Besjuut besjtuur Besjtuur betroewbaar Betroewbaar beteikenis Beteikenis beuk Beuk bevrundj Bevrundj bewaeging Bewaeging bewaere

Bewaere bewaege Bewaege bewieze Bewieze bezeike Bezeike bewirke Bewirke bezeikerie Bezeikerie bezunjer Bezunjer bezoepe Or Bezoepe bieboew Bieboew biehaole Biehaole biejein Biejein biejeinrope Biejeinrope bieval Bieval biezaak Biezaak biewirke Biewerke bikke Bikke bitje Bitje blaad Blaad blajer Blajer blajere Blajere blaoze Blaoze bleik Bleik bleud Bleud blieve Blieve blinddook Blinddook boetekantj Boetekantj broelof Broelof broen Broen broeke Broeke broed Or Broed broewer Broewer broewe Broewe broor Broor zitterd Zitterd brouze Brouze Bunj bunj buun Buun buus Buus centimaeter Centimaeter daag Daag daakpan Daakpan dabbe Dabbe dageroad Dageroad Daezelfde daezelfde meziek Meziek gein Gein doa Doa Daobie Or daobie Doabie doabie daomit Daomit Daonao daonao Daorom daorom Daorentoe daorentoe deef Deef deenst Deenst deer Deer Deep deep Defek defek degelik Degelik deig Deig dees Dees deil Deil dekbed Dekbed den Den derm Derm desnoeds Desnoeds det Det deurklink Deurklink diek Diek dien Dien dichter Dichter dich Dich direk Direk doe Doe doed Doed doef Doef zunj Zunj doedzunj Doedzunj doelpuntj Doelpuntj doem Doem does Does doesj Doesj Doezendmaol doezandmaol doorbiete Doorbiete doorbaore Doorbaore baore Baore Gaon gaon dach Dach kieke Kieke kiek Kiek laeze Laeze liere Liere kaoke Kaoke kaok Kaok haole Haole moele Moele Rieje rieje sjakele Sjakele sjete Sjete spjeule Or Spjeule sjrappe Sjrappe sjtarte Sjtarte sjtrieke Sjtrieke sjuve Sjuve sjtarte Sjtarte douf Douf drage Drage draen Draen draod Draod drek Drek dreug Dreug driej Driej driemaol Driemaol driepoet Driepoet paol Paol drieve Drieve droef Droef duje Duje duip Duip echgenoet Echgenoet einmaolig Einmaolig einige Einige erm Erm erme Erme es Es euver Euver blieve Blieve euverblieve Euverblieve doon Doon euverein Euverein goeje Goeje handj Handj haer Haer hauwe Hauwe euverheid Euverheid euveriges Euveriges euverlas Euverlas euvermach Euvermach sjikke Sjikke sjete

Sjete riete Riete rope Rope sjakele Sjakele plekke Plekke sjreve Sjreve sjpuite Sjpuite sjtaek Stjaek sjpanne Sjpanne sjwumme Sjwumme euverste Euverste euverval Euverval euvervalle Euvervalle wirk Wirk wirke Wirke euverzeen Euverzeen euverwirke Euverwirke gaeve Gaeve goeje Goeje haole Haole kieke Kieke kraom Kraom laot Laot laote Laote fabrekantj Fabrekantj faktuur Faktuur fannetik Fannetik febrik Febrik fatsenere Fatsenere fernuus Fernuus fien Fien fies Fies Fieste fieste fitser Fitser flares Flares flemoesj Flemoesj flink Flink fluustere Fluustere froemel Froemel gaar Gaar gaaf Gaaf gaas Gaas Gael gael gaer Gaer gaeve Gaeve gans Gans gaonde Gaonde gauw Gauw geboage Geboage geboaje Geboaje gebeed Gebeed geboew Geboew geer Geer gehiemelte Gehiemelte gehurig Gehurig geis Geis gein Gein kraom Kraom kwatsj Kwatsj gekruud Gekruud gelaegenheid Gelaegenheid gelaje Gelaje geldj Geldj noed Noed geleef Geleef geliek Geliek gelierd Gelierd gemein Gemein raod Raod gemieterd Gemieterd Gepaot gepaot gereidsjap Gereidsjap gereidmake Gereidmake gesjtaag Gesjtaag getuge Getuge geveul Geveul gevraet Gevraet vrunj Vrunj gewaere Gewaere gewich Gewich Gewoen gewoen gewuunlijk Gewuunlijk gezeen Gezeen gezeiver Gezeiver gezelsjap Gezelsjap gezich Gezich gezondj Gezondj glaas Glaas glansriek Glansriek godmiljaar Godmiljaar goodkaup Goodkaup gooj Gooj good Good gool Gool gouwe Gouwe graaf Graaf graas Graas grach Grach greij Greij greuj Greuj Greun greun greunte Greunte gries Gries Gros gros gruwelik Gruwelik hae Hae haer Haer haekel Haekel nuuj Nuuj vleisj Vleisj handjbook Handjbook dreug Dreug baog Baog handj Handj hanjel Hanjel haof Haof haok Haok haoke Haoke haoks Haoks haole Haole Or haor Haor haopelijk Haopelijk haordreuger Haordreuger haorfien Haorfien haos Haos haoveneer Haoveneer Hasselt hasselt Hauf hauf hatelijk Hatelik hauflaeg Hauflaeg haufvol Haufvol heit Heit helop Helop henjig Henjig herhaole Herhaole herkeze Herkeze herkinbaar Herkinbaar laeve

Laeve laeze Laeze sjmilte Sjmilte herte Herte heur Heur Hie hie hienao Hienao hiering Hiering hievan Hievan hierlijk Hierlijk hinjer Hinjer blaor Blaor Bloare bloare Höbbe höbbe hoes Hoes hood Hood Hook hook hooste Hooste huuske Huuske ierlik Ierlik iers Iers ierste Ierste ies Ies iezer Iezer iezig Iezig inaome Inaome blieve Blieve bringe Bringe indeile Indeile inflasie Inflasie ingank Ingank ingels Ingels ingeval Ingeval inhoale Inhoale inkoupe Inkoupe inrichte Inrichte ins Ins inzeen Inzeen inzeipe Inzeipe janke Janke jaomer Jaomer jaor Jaor jaortal Jaortal jeder Jeder jeh Jeh kaat Kaat kael Kael kaers Kaers kaetel Kaetel kaever Kaever kaf Kaf kaffee Kaffee kal Kal kantj Kantj kaole Kaole kepot Or Kepot ketse Ketse keul Keul keutel Keutel kirmes Kirmes Kits kits inrichte Inrichte ins Ins inzeen Inzeen inzeipe Inzeipe janke Janke zoepkool Zoepkool stjasiefestasie Stjasiefestasie Kits kits klaor Klaor klaorkomme Klaorkomme klaorsjtaon Klaorstjaon kleid Kleid kleijer Kleijer klink Klink klip Klip klommel Klommel knalpiep Knalpiep koel Koel drieve Drieve content Kontent kopdook Kopdook kortbie Kortbie kratse Kratse krempel Krempel krets Krets kriet Kriet krietwit Krietwit kromp Kromp kuns Kuns kuuj Kuuj kwis Kwis laaj Laaj laam Laam laeg Laeg laepel Laepel leave Leave laever Leaver laje Laje landj Landj lanje Lanje laot Laot laote Laote leef Leef ledder Ledder leefs Leefs leid Leid lempke Lempke lestig Lestig letterlik Letterlik lever Lever lieke Lieke liem Liem Limburgse limburgse Limburgs Limburgs linze Linze abrikozen Abrikozen rieste Rieste maas Maas Maedje maedje mach Mach maete Or Maete laot Laot laote Laote leef Leef ledder Ledder leefs Leefs leid Leid lempke Lempke l1 L1 greuts Greuts Manj manj mannelik Mannelik Maeter meater maord Maord meistal Meistal meister Meister mets Mets metser Metser mich Mich middig Middig miervoud Miervoud misdoon Misdoon misdaon Misdaon sjpraek Sjpraek moel Moel moeze Moeze mot Mot

mottig Mottig nach Nach naat Naat naer Naer naef Naef nao Nao naod Naod naodeil Naodeil neet Neet waal Waal noe Noe gein Gein nuuj Or Nuuj oer Oer oet Oet opzeuke Opzeuke oug Oug paat Paat paer Paer paerd Paerd Paosje paosje plaog Plaog plaoge Plaoge veur Veur mit Mit weej Weej ich Ich du Du

Table 2: Search query for Limburgian tweets