Universiteit Leiden
Opleiding Informatica

Dutchism Detector

Name: Bernard van den Boom

Date: 14 July 2016

1st supervisor: dr. Cor J. Veenman

2nd supervisor: dr. Arno J. Knobbe

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)

Leiden University

Niels Bohrweg 1

2333 CA Leiden

The Netherlands

Dutchism Detector

Bernard van den Boom

Abstract

Dutch intelligence agencies can only go after persons that are involved in criminal online behavior when they are in their jurisdiction. We propose a way to overcome the arduous task of manually investigating whether a user on an internet forum, possibly involved in criminal activities, is Dutch or not. We aim to uncover possible characteristics and, more importantly, detect that some English texts are written by a Dutch native user. Ultimately, the goal here is to provide intelligence agencies a tool to distinguish online malicious or threatening comments as from either Dutch natives or non-Dutch natives to be able take appropriate action. To develop a data-driven detector we prepare a specific corpus for this task. The gathering of the data is done rigorously and consists of forum data from a large group of Dutch and non-Dutch users. We then use the bag-of-words representation to extract features in addition to using common preprocessing techniques, word n-grams and tf-idf. Following this process, we compare two classifiers: linear support vector machines and logistic regression. Because our data has considerably unbalanced classes, the choice of performance metrics is an additional challenge. The performance metrics that were found appropriate are precision-recall curves and its average precision score, the $f1$ score and precision. Our results show logistic regression with frequency-based feature selection performs best at predicting Dutch natives. Further study should be directed to the general applicability of the results, that is to find out if the developed models are applicable to other forums with comparable high performance.

Contents

Chapter 1

Introduction

1.1 Background

Intelligence agencies undoubtedly struggle with the massive amount of content that is posted online, some of which is of a criminal nature. Searching for this criminal content on the whole web is a daunting task, especially considering it is often hard to come by the nationality of offenders. What is clear is that criminals use the internet as their medium to sell illegal arms or drugs as well as more extreme cases such as offering hitmen services. We are especially interested in content posted on what is often called the dark web. Content on the dark web is often criminal of nature. Most commonly found content include black markets, child pornography, fraud or mail order services.

Apart from the enormity of the internet, there is another problem for agencies who need to deal with criminal content online. Dutch intelligence agencies cannot follow up on users involved in criminal online behavior that are non-Dutch. At best, they can communicate their information with their fellow intelligence agencies. In other words, intelligence agencies deal with the magnitude of the internet featuring criminal content from users with a wide variety of nationalities of which usually only one is relevant to them. A system for automatic identification of the native language of web users would be ideal. In our case, we specifically aim to identify Dutch users writing in English.

The problem touches upon literature that deals with authorship attribution [32], native language identification [9], review classification [34] or Twitter sentiment classification [12]. Research makes clear texts written by non-native speakers often include hidden clues that betray their native lan-

guage. Writers are prone to misspellings, grammatical errors or unusual turns of phrases that are characteristic of their mother tongue. In this thesis, the goal is to leverage the power of machine learning and automated text analysis to uncover these clues further and, more importantly, detect English texts written by Dutch authors. Our problem is interesting because there is some research on native language identification but not much. The field is relatively new.

In sum, the problem seems relevant for both intelligence agencies as well as research, because it is (1) a practical problem and (2) it could advance research in native language identification, specifically using applicable and practical corpora.

1.2 Problem Statement

All things considered, we can summarize our problem as to provide intelligence agencies a tool to distinguish written online English text by Dutch users from English text written by non-Dutch users by means of text classification models. This statement is used as a guidance throughout this thesis. Note that we are focusing solely on text, using no metadata whatsoever, such as IP addresses, which for one could be costly to follow up on anyway.

1.3 Related work

Apart from beforementioned authorship profiling, review classification or social media sentiment classification, classical tasks of contemporary text classification include identifying emails as spam or not, or categorizing news stories by subject, often using data from available corpora such as Reuters.

Of the contemporary classification problems, native language identification is one that comes closest to our problem, see for example [9]. Much of native language identification research including Gyawali et al. use the TOEFL corpus, while we aim to use a more practical corpus, that does not consist of test essays from non-native students writing in a foreign language. Wong and Dras [35] also use a corpus of learner English. In addition, to the best of our knowledge the Dutch language specifically is not experimented with. As Grant [8] further points out: the most serious challenges of researches of native language identification include deciding on the corpora they will use.

With regards to the document classification task process: it proceeds in a fairly standard process [1]. First, we collect our textual data. Then, we preprocess it by taking common steps such as tokenization, removal of stop-words and stemming words. What often follows is a indexation step, for example by using the vector space model, creating vectors of words. Because we are dealing with text data, we often need to select features to address the high dimensionality of the feature space. This is commonly done by using techniques such as information gain, term frequency or performing the chi-square test [16], [38], [7].

After preparing our data we need to make a decision on what classification model to use. Many different models have been used to develop classifiers, some with more success than others. These models include naive Bayes [19], or other Bayesian models, support vector machines [13], logistic regression (also known as maximum entropy) [19], the k-nearest-neighbor classifier [33] and random forest classifiers [36].

1.4 Thesis Overview

The rest of this thesis will have the following structure. In Chapter 2, corpus collection is discussed, methods of handling the data will be explained, and why, how and what machine learning algorithms are used. In other words, Chapter 2 provide a guide through the machine learning process and the methods used. Experiments and our results will be presented in Chapter 3. In Chapter 3 we also discuss our results in detail, critically evaluate our approach, our model and its implications in detail. Finally, in Chapter 4 we conclude with our most significant results and take a look at future research.

Chapter 2

Methods

In this chapter we discuss our corpus collection thoroughly. Because data is not readily available, we have to obtain our data ourselves. We elaborate on how we rigorously gathered data from Reddit, distinguishing between Dutch users and non-Dutch users. Next we discuss feature extraction and address the high dimensionality we face. Finally, we consider our algorithm selections.

2.1 Reddit

In this thesis, Reddit is used as a platform to gather data to eventually develop a model that – on the basis of comments – labels a user as being a native Dutch speaker or a user with a different mother tongue. Reddit is an online community where registered users can, among other things, submit content, vote on submission and comment on them. There are currently around 40 million user accounts. Visitors can also view most content without having a user account [24]. Almost all parts of Reddit are in English; the majority of the users on Reddit is from the U.S. Exact numbers on where Reddit users come from are not publicly available. In our thesis, we use no meta data, which is hard to come by anyway with Reddit.

Reddit is a suitable data source, because it is an online forum that allows for easy access of their content. Ideally, we would use content from the dark web, which is particularly relevant for our problem. However, because it is hard (and sometimes even illegal) to gather data from these web pages, Reddit provides a decent alternative.

2.2 Data Acquisition

Reddit provides us with an API [25] of which the code is open source. It supports many methods including gathering data from particular subreddits and users by making a variety of calls. For this research project, comments are acquired through these methods and by a Python package that allows for simple access to Reddit's API [22]. In essence, we create a labeled dataset where as much comments as possible from one user are treated cumulatively. In authorship identification literature this is called profile-based classification, unlike its opposite instance-based classification [21], where we would consider each comment from a user as a seperate instance.

2.2.1 Dutch users

In order to correctly classify users as either Dutch or non-Dutch, we first obtain data of which it can be known with some reasonable certainty that they are native Dutch users. We gather the data to use it for supervised learning in which Dutch/non-Dutch present the labels. The data is gathered by the following process:

1. Because the vast majority of subreddits is non-Dutch, a list of the largest Dutch subreddits that could be found were gathered manually. We end up with around 900 users from some of the largest subreddits that have been observed to be in Dutch. We make the assumption here that users that post comments in Dutch are in fact Dutch.

2. Of each of these Dutch users, we extract the following content: comment, user, language and subreddit. It includes the *comment* some *user* has commented in a particular *language* as detected by Googles language-detection ported to Python [29] in a certain *subreddit*. A subreddit is a sub forum on Reddit, which collectively form Reddit. We end up with roughly 540000 individual comments from Dutch users.

3. As a last step we remove the rows that include comments in Dutch or any other language that the language detector was able to come up with. The result is around 400000 comments in English from assumed Dutch users.

Although every comment is passed through the language detector, it does not seem to be a sufficient enough approach. The language detector wrongly identifies some comments – and thus some users – as speaking a certain language. It does so because (1) some comments are too short to classify

but are classified (possibly wrongly) anyway and (2) some comments include uncommon characters such as emoticons or Reddit specific slang. How much errors the language detector actually makes is hard to say because the number of comments is too high to check by hand. It might label some short comments correctly, others incorrectly. Indeed, the language detector recommends using "text [that] has some length, almost 10-20 words over" [29]. In any case, there is no way of checking systematically, except for possibly taking a sample.

For the Dutch user data, this is not much of a problem. After all, we know they are Dutch so they are already labeled as having the correct language label *Dutch*. We have also removed all the rows not labeled as having the language *English*. Unsurprisingly, looking through the data by hand, not many comments seem to be labeled as *English* while they are in fact not English, considering most comments are in English on Reddit anyway. All things considered, the language detector proves to be quite effective on the Dutch data set, but it cannot be guaranteed some comments are in a language other than English.

2.2.2 Non-Dutch users

Because we only distinguish between Dutch natives and non-Dutch natives we assume that all the users that do not post comments in Dutch and are not part of our list of Dutch users, are in fact non-Dutch users. The exact process for gathering non-Dutch user data and the attempt at detecting their native country is as follows.

1. In order to come up with an initial list of (possible) non-Dutch users we take the top 200 of subreddits where Dutch users from the Dutch user data have posted in (in English). This allows for comments not dealing with widely varying themes, that would likely lead to overfitting the data later on. The idea here is that the dataset becomes more homogenous if we gather comments from subreddits where both Dutch natives as well as non-Dutch users comment in. From each of these subreddits in the created subreddit list, we gather as much users as possible, and filter out all the Dutch users we have found in our Dutch user data. We now have a temporary list of 90000 non-Dutch users. That is 100 non-Dutch users for every 1 Dutch user.

2. Step 2 is identical to Step 2 of the Dutch users data gathering process, except we gather less comments per user, because there are simply more users to gather data from. The gathering

of this data takes a couple of days and results in a large csv file of around 2.5 GB with the same header fields as the Dutch user data set.

Now, as the language detector is proven to be not as effective on short comments and or 'weird' characters and there is no way of checking how good it works, an attempt is made to classify subreddits, instead of single comments, in a certain language. If we assume a user is labeled a language l if the subreddit is in a foreign language and some user posts in that subreddit, we can eventually match users to a language. We hope that subreddits contain enough comments to correctly find out its language and thus the user's language. Often, subreddits are in one language or another, not mixing languages in the particular subreddit. Subreddit language is determined as follows:

1. Create a list of unique subreddits from the non-Dutch user data. These are all the subreddits the presumably non-Dutch users comment in.

2. Of each subreddit, 30 comments are grabbed and concatenated. 30 proves to yield a set of comments large enough to be tested by the language detector.

3. On that concatenation we apply the language detector.

4. The language that is detected is paired with the unique subreddit. The result is a dataset with subreddits and their corresponding language.

After we have determined the subreddit language of all the subreddits that our assumed non-Dutch users posted in, we remove all users that post in Dutch subreddits from our non-Dutch data set and add them to our Dutch data set.

As a last step, but important step, we concatenate all comments belonging to a single unique user together. This is done because concatenating the comments fits to the task. After all, we are interested in determining the language of a user, not of a specific comment. We are much less interested in a classification per message than messages per user.

As a consequence of concatenating comments, it will most likely also be easier for the machine learning algorithm to eventually classify documents. That is, they contain more data to extract information from. This is also why grouped comments of users that are shorter than 2000 characters are removed as well. This decreases the mistakes the model makes on short comments that only include symbols or are illegible.

After closer examination by hand we notice that there remain quite some comment sets that include sentences in Dutch. The language detector apparently has not been able to detect these comments individually. For that reason we run the language detector again, now on the comment set, removing another 100 or so instances of Dutch users. We attempt to delete as much Dutch as possible from the individual comments during gathering first and from the comment sets later.

To summarize, we have gathered data in two main steps. First, we have largely by hand created a list of Dutch users (a minority on Reddit) that post in Dutch subreddits. We then composed a list of preliminary non-Dutch users from the subreddits Dutch users post in. We double checked whether they were non-Dutch by matching the users with the Dutch data, and by language detecting the subreddits they post in. Lastly, we removed rows of comment sets that were put through the language detector once again and were flagged non-English. The data is now ready to be transformed into features, and is in the format:

```
[user] [aggregated comments] [nl/other]
```

The user is the name of the user, the aggregated comments are all the comments gathered from a particular user and concatenated together and the last column is one of the two languages *Dutch* and *English* (which in fact is everything non-Dutch).

2.3 Feature Extraction

In order to be able to use many of the machine learning algorithms, text data needs to be converted into numerical feature vectors. The bag-of-words model is an approach widely used in the field of natural language processing, for example see [27], [28], [17]). With such an approach we essentially disregard word order: every comment is taken as a multiset of its words, keeping only track of the frequency of each word. This frequency is then used as a feature for training the classifier.

Our tool kit provides many parameters that can control how to preprocess data, besides transforming the data with tf-idf. Parameters include stripping accents, determining whether to remove stop words, lowercasing all characters and setting a n-gram range and whether to use character n-grams or word n-grams [1]. Much of the parameters have been kept as simple as possible and as close to contemporary research as possible. We elaborate on our steps and choices below.

[1] For full documentation, including all the parameters, see `http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html`

2.3.1 Tf-idf

First, we tokenize each comment collection into words and then use a technique called term frequency times inverse document frequency (tf-idf). With this technique we quantify the importance of words in documents (comment collections) by calculating weights for them. The most frequent words are not always the words bearing the most information. Indeed, they are often words that are so called stop words ('the', 'and', 'or' etc.). Often, rarer words can give more information whether or not some user is Dutch or not. For example, Dutch users might make certain spelling mistakes that non-Dutch users almost never make.

The calculation of tfidf is as follows:

$$TF_{ij} = \frac{f_{ij}}{max_k f_{kj}} \tag{2.1}$$

with f_{ij} the frequency of word i in document (comment) j. So, the term frequency of term i in comment is f_{ij}, normalized by dividing it by the total number of words in the comment. The most frequent term in comment j gets a term frequency of 1, others fractions. The inverse document frequency $IDF_i = \log_2(\frac{N}{n_i})$ with N the total number of comments and n_i the number of documents containing word i. Tf-idf then becomes $TF_{ij} \times IDF_i$. Higher values for tf-idf are obtained by a high frequency for the comment, but low frequency over all comments [23].

2.3.2 N-grams

The use of n-grams is common in text classification tasks, see for example [2], [9]. With word n-grams we extract n contiguous words from a document. Similarly, we could use syllable or character n-grams. n-grams are aimed at eventually providing us with a prediction of the next word or character in a sequence of words (for example the next word in a sentence). Although fairly simple, n-grams have proven to be very effective in many applications. It was not feasible to test a wide variety of n-gram ranges during our resarch, although that might have improved our final results. Because it quickly became clear character n-grams performed less than word n-grams we have not followed on using character n-grams. Furthermore, we decided to use unigrams, word n-grams of size 1, because using bigrams or even trigrams leads to memory problems caused by the high number of then created features.

2.3.3 Other parameters

We have decided to lowercase all characters. We lowercase to prevent our vocabulary from increasing significantly in size without showing an increase in performance according to early tests. We do not strip any accents or remove any stop words. Stop words are usually removed in text classification, but in our case it might be useful, namely, Dutch users might choose different stop words than non-Dutch users.

Lastly, we binarize the term frequency in tf-idf. Purely by some initial experiments, this parameter setting has proven to be giving better performance. Term frequencies are also by default normalized by using the $l2$-norm.

After feature extraction we end up with as many as 2300000 features.

2.4 High dimensionality

2.3 million features can lead to some problems. First, storing so many features requires a vast amount of memory storage. Secondly, high dimensionality might lead to worse performance or even overfitting, when dimensionality increases while training samples remain fixed [11], [37]. Overfitting might be a direct result of this so called curse of dimensionality. Fortunately both our models are linear and regularized which – assuming properly tuned penalty parameters – makes them more resistant to overfitting [10]. We further cover two ways of dealing with the high dimensionality of our data.

One common way of addressing high dimensionality is feature selection. Feature selection has in many cases proven to be useful to simplify the eventual model without giving up performance and even improve generalization accuracy and avoid 'overfitting'. In recent literature, mutual information, information gain, term frequency and chi-squared are among the feature selection techniques that have proven most effective [30]. We experiment with chi-squared and term frequency. For term frequency, we limit the maximum number of features by ordering by term frequency in our corpus. With chi-squared feature selection the statistical chi-squared test is used to select features [7].

Regularization is another way of dealing with the curse of dimensionality. For instance, feature selection can be done by $L1$ regularization. We, however, use $L2$ regularization, because $L1$ regular-

ization gives worse results. Regularization ensures the weights the model gives are not fit too well: weight values are penalized by regularization. Simply put, the difference between the two types of regularizations is that $L1$ regularization can shrink weights to zero, effectively eliminating them, while $L2$ regularization shrinks the weights too but eliminates none.

What cannot be left unmentioned when using feature selection is that it might result in a loss of information, because even the worst features might score better than random. This implies that these features contribute to performance. Indeed, in text classification there seem to be only a limited number of truly irrelevant features, according to Joachims [13].

2.5 Model Learning

As has been mentioned in Section 1.3, machine learning algorithms that are often used with text data include naive Bayes, or other Bayesian models, support vector machines, logistic regression (also known as maximum entropy) and the *k*-nearest-neighbor classifier. The latter algorithm generally is fairly slow compared to the others. For that reason, it has not been tried out in the experiments. Naive Bayes, although it is a simple and relatively quick algorithm, also does not work very well with the implementation that is offered with Pythons Scikit library. The class imbalance was hard to overcome. The two remaining algorithms that are mentioned, regularized logistic regression (LR) and support vector machines (SVMs), were tested.

2.6 Class imbalance and model support

While naive Bayes was not ideal for dealing with class imbalance without having to extend Scikit and develop more advanced methods, the other algorithms have built in support for dealing with class imbalance. The class imbalance is of paramount importance to address. The imbalance of classes in the data – there is around one Dutch user for every 100 non-Dutch users – poses a new challenge. Fortunately, both the implementations of logistic regression and the linear support vector machine support a class weight that adjusts weights according to the imbalance. It does so by dividing the total number of samples (comments) by the number of classes (*Dutch* or *English*) times the frequency of the class label [20]:

$$\text{weight}(y) = \frac{\text{\# samples}}{\text{\# classes} \times \text{\# occurrences of y}} \tag{2.2}$$

2.6.1 Logistic Regression

Logistic regression, also known as maximum entropy, is a discriminative classifier, which builds a model upon the features that are most distinctive for a class. It was developed in 1958 by Cox [3]. The binary classifier returns well calibrated unbiased probabilities, because it, above all, optimizes log-loss [14], [20]. Logistic regression thus works with probabilities, in contrast to support vector machines. It requires the tuning of parameter C, which we tune by gridsearching using average precision as performance measure. We find a C of 1000 performs best. By default sklearn's implementation uses $l2$-penalization. Both logistic regression as well as regularization come from the LIBLINEAR library [5].

2.6.2 Support Vector Machines

In a nutshell, the goal of an SVM is to find a seperating hyperplane that is optimal, maximizing the margin of the training data. As Joachims rightly points out, SVMs are "universal learners" that can learn independent of the dimensionality of the feature space [13]. Text data has many properties, such as the ones listed above, with which SVMs deal very well. Not only theoretically, but also in practice Joachims shows SVMs turn out to show good performance on text categorization tasks. Additionally, just like with logistic regression, our linear SVM only requires us to tune parameter C, the penalty parameter of the error term. We find $C = 0.5$ produces the highest average precision during gridsearch.

Chapter 3

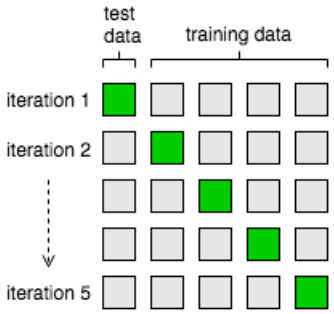
Evaluation

3.1 Evaluation methods and performance metrics

3.1.1 Cross Validation

To test the models, we use 5-fold cross validation with stratification. We stratify because stratification generally performs better than regular cross-validation, both in terms of bias as well as variance, according to Kohavi [15]. We apply this technique by dividing our data set in five subsets using one of the subsets as test set, the rest as training set. We repeat this process five times until every subset has been assigned test set. Because we use stratification, we take into account the balance of the classes in our division of data such that it correctly reflects the entire data set. That is, every subset has roughly the same percentage of each class as the original data set. The general cross-validation is clarified in Figure 3.1.

Figure 3.1: Simple 5-fold cross validation



3.1.2 Performance Metrics

Usually, the performance of a classifier is measured by accuracy. However, in this case the accuracy would quickly approach 99%, would a classifier always label a comment non-Dutch. So, a challenge is to find an appropriate performance metric for evaluating our model.

Fortunately, there are many ways of testing a binary classifier on how well it performs when dealing with imbalanced classes. The receiver operating characteristic curve (ROC curve), and its area under the curve (AUC) is such a way [31]. The graphical curve plots the true positive rate (TPR) against the false positive rate (FPR) at various thresholds. The AUC then tells us something about how well the algorithm does; the higher the better it does as predicting the class label. It is equal to the probability that the classifier will rank a randomly chosen Dutch instance higher than a randomly chosen non-Dutch instance [6]. However, it turns out the ROC AUC does not seem to work well for problems with more negatives (non-Dutch) than positives (Dutch). To be more specific, "a large change in the number of false positives can lead to a small change in the false positive rate used in ROC analysis" [4]. An alternative that does not take into account the false negatives (FN) is the precision recall curve (PR curve) and its area under the curve. This metric compares FP with TP, not TN and presents the tradeoff between precision and recall. For that reason, we have chosen the PR curve and its AUC, not the ROC curve. We especially use the area under the precision recall curve, which is also known as the average precision score and describes the precision recall curve.

Another useful performance metric that can be used to judge the classifier on includes the $f1$ score which is defined as the harmonic mean of precision and recall. In our particular case, however, we would prefer to let our model find Dutch users with a high probability, then find more Dutch users but also classify many non-Dutch users as Dutch. In other words, we are willing to trade some recall for more precision. We consider precision more important than recall; we rather have false negatives than false positives. Still, with the $f1$ score we can keep an eye on recall as well.

The latter performance metrics (precision, recall, $f1$ score) are set-based measures, calculated using sets of comments that are unordered. On the contrary, precision recall curves deal with a balance between precision and recall. It works with thresholds that allow us to trade precision for recall. The performance at a range of thresholds can be visualised by plotting a precision recall curve. Its area under the curve (average precision score) then provides an excellent performance metric to compare models with.

All things considered, our most valuable performance metrics are average precision and precision. The equations below summarizes how our metrics are calculated.

Performance metrics:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.1a}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.1b}$$

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{3.1c}$$

$$\text{Average precision} = \text{area under the precision/recall curve} \tag{3.1d}$$

3.2 Experiments

The following experiments compare the performance of logistic regression with a linear support vector machine. As has been mentioned before, training is performed with Python's Scikit-learn library on the dataset that we have obtained from Reddit. Our final dataset is significantly skewed and consists of 91539 users and comment bases of which 852 are determined Dutch users, 90687 non-Dutch users. Class imbalance is $1 : 100$. We are using the parameter settings as described in Section 2.3. The task here is to classify a comment set from a particular user as either *Dutch* (1) or *non-Dutch* (0), thus presenting a binary classification task.

Note that because time and computing resources are constrained in making this thesis, only a confined number of parameter settings were able to be used for experiments. For example, vectorizing bigrams creates memory problems. In addition, because of the vast number of available parameters that the Sklearn library provides, we cannot try out every possible setting. For that reason, we base our parameter settings among other things on the choices made in similar research.

3.3 Results

The results are presented in terms of the beforementioned f_1 score, precision score and average precision score. Moreover, stratified 5-fold cross validation is performed using no feature selection, chi-squared feature selection as well as term frequency feature selection.

3.3.1 Important features

First of all, both classifiers keep track of the importance of features by having a coefficients attribute that holds the coefficients of the features. If we order these features by importance we get an interesting insight into the word usage of non-Dutch and Dutch users. Some of the most important features (the features with the highest coefficients) after running LR – the SVM shows similar results – include terms that clearly increase the chances that a user is Dutch. These include 'the netherlands', 'dutch', 'nl', 'holland' etc. and some Dutch words that have slipt through the language detector as was expected. The second most important features are of a more interesting category: they are spelling errors. The model seems to have found out Dutch users make certain spelling errors that might identify them as being Dutch. These include the spelling mistakes 'eachother' and 'ofcourse' or 'offcourse' which in fact both are to be written as two words 'each other' and 'of course'. 'trough' is another typo or spelling errors the model marks as being important as well as 'focussed' which should be 'focused'. Unfortunately, we get little insight into sentence structure of Dutch writers, because of the bag of words model and unigrams we use.

Table 3.1: Results of experiments with two models, logistic regression and a linear support vector machine with different feature selection methods

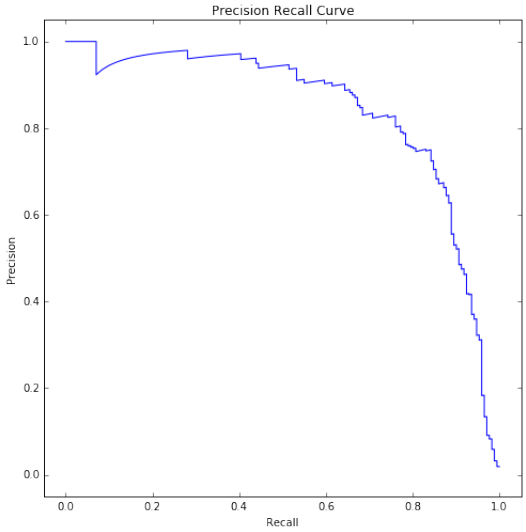
	Logistic Regression				Linear SVM			
	none	chi2	top 10000	top 100000	none	chi2	top 10000	top 100000
F1	0.772	0.772	0.736	**0.772**	0.750	0.758	0.729	0.754
Precision	0.827	0.840	0.741	**0.846**	0.797	0.803	0.690	0.808
Average Precision	0.829	0.829	0.785	**0.833**	0.813	0.819	0.780	0.818

Table 3.1 shows the results on our Reddit dataset. As has been mentioned three measures of performance have been used: $f1$, precision and average precision score.

For logistic regression, we find that $f1$ scores are similar across the choices of feature selection methods (0.75). Of the four test settings, logistic regression with maximum features set to 100000 gives the best performance. To be fair, the difference of using this feature selection method or chi-squared feature selection is only slight. The precision recall curve of our best performing model is shown in Figure 3.2a as well. Also, see Figure 3.2b for the precision/recall curve of our best performing linear SVM. Our linear support vector machine setup shows comparable good results. Again, chi-squared feature selection and frequency based feature selection score nearly the same.

Overall, we note that logistic regression considering only the top 100000 features scores best on every performance metric, although the difference with some others are small.

(a) Logistic regression – maximum features 100000



(b) Linear SVM – maximum features 100000

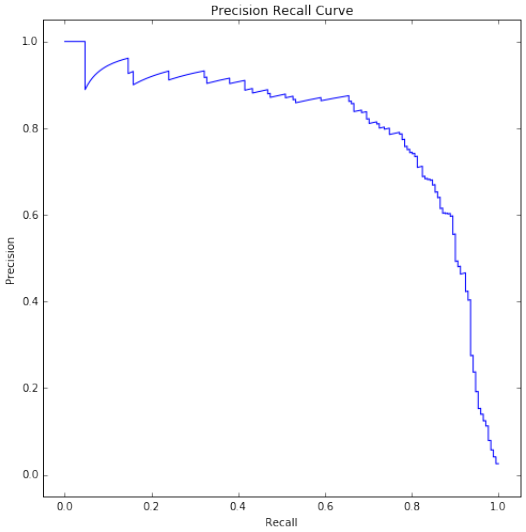


Figure 3.2: Precision/recall curves of the two best performing models

3.4 Discussion

Results in Table 3.1 are acceptable compared to other studies in the text classification field and as became clear from the results, logistic regression has shown the best results. Now, considering our background and problem statement, we are not only interested in what model performs best, but also how we would use the model to make sensible recommendations to intelligence agencies. To put that differently, we might not only be interested in predicting a class label, we are also interested in some sort of value that expresses a confidence on that prediction. Some models are better at giving estimates of class probabilities than others. Some even lack support for any kind of probability prediction. Fortunately, logistic regression is an algorithm that has the advantage of returning probabilities, because it directly optimizes log loss [18].

With our logistic regression model we can advise intelligence agencies in a couple of ways, while not overlooking some important points. One option is to simply let the model classify the user that intelligence agencies provide us with. We would scrape as much comments as possible from that user and feed it to our model. Another option is to recommend based on the probabilities the logistic regression model comes up with. They both provide a tool to intelligence agencies and at the same time uncover some hidden clues that we sought after in the form of spelling mistakes. Nevertheless, both options also pose some problems in practice.

Clearly, Reddit, although quite representative of an internet forum, might not necessarily represent a

forum where many criminals post comments. Despite using feature selection, it is hard to determine how generalizable the model is to other forums. Besides, not all forums offer an easy way of gathering information. It is unlikely that many forums offer a similar API that Reddit offers. This can, however, be partially overcome by simpley scraping the website with a variety of tools widely available (for example [26]). What is hard to overcome is the fact that some forums allow users to post anonymously. Our model requires a set of comments, because it needs longer comments in order to be able to make a classification. Short comments alone will almost certainly not work with our model. So, in case a comment is posted anonymously and the comment is short, our model would not solve our problem of distinguising Dutch users from non-Dutch users. After all, we cannot gather more comments from the same user without them having some sort of name or id. To add to that, even if the user would not post anonymously, our approach also requires the length of aggregated comments to be of some sufficient size. Users might not post as much content at all. What is clear, is that performance of our model will suffer in case of shorter comments.

Another problem we came across that should have been addressed more carefully is how to use the language detector to remove Dutch or other comments in a foreign language from our data. We first used the detector on the comments individually, then as a whole, but still found Dutch phrases in our data here and there. We ultimately look for a model that distinguishes English from users with a different native tongue, not detect a foreign language.

Regardless of these limitations, we have two options to distinguish users. Option one solves the problem of distinguishing users by letting the model do all the work. Some misclassifications will certainly be made, but performance is reasonable. Still, these misclassifications are understandbly undesirable for intelligence agencies, because pursuing Dutch labeled users that end up not being Dutch is costly, while not pursuing non-Dutch labeled users is not. The issue of false positives was partially addressed by trying to maximize precision.

Option two allows for more flexibility in advising law enforcement. We now need to decide on how to use the calibrated probabilities produced by the logistic regression model. One way to go about this would be to rank all instances of the test set of Dutch users by probability and place a certain user, that is used as input to the model, in that ranking accordingly. If it is ranked in our top x, after we have observed that our model made the correct classification in some high percentage of the time, we recommend investigating the (Dutch) user. Would the user be ranked out of that top x, we would not advise intelligence agencies to undertake action. The choice of x is hard and needs

more research. In our case, we have not been able to come up with a good approach to advising intelligence agencies by ranking a user. For that reason it is advised to stick to the model's decision or discuss with both parties what to make of the probabilities the model produces.

Chapter 4

Conclusions

This thesis has posed an interesting and practical case of text classification: can we distinguish Dutch users from non-Dutch users on a typical online forum? We have shown that with the help of popular machine learning algorithms such as logistic regression and support vector machines, a differentiation – without using any metadata – can be made between Dutch and non-Dutch users that post comments on an internet forum. Despite the difficulty of data collection, we experimented with two models performing well on the dataset we acquired from Reddit, with $L2$ regularized logistic regression performing best, albeit only slightly. Interestingly, the models found some clues we were looking for. It found that important features included obvious dutch words, but also common spelling errors. Generalizability might be an issue, as other internet forums might pose obstacles such as offering no API or more problematic, allow users to post anonymously. In the end, we propose two pragmatic solutions to the original problem: (1) we can distinguish Dutch users by simply adhering to the model's decision on a user or (2) discuss an approach based on the well-calibrated probabilities logistic regression produces.

4.1 Future research

Future research should first of all be directed towards finding out whether the developed models can be applied to other forums, that is research applicability to other forums. Also, results can be improved by expanding computing power to be able to use bigrams and trigrams and allow for a more advanced grid search. Moreover, with more time and computing power, intensive tree-

based algorithms could be tried out as they have been used widely in the field. Lastly, we were unsuccessful in preventing foreign languages from slipping through. Would you want to distinguish English, written by natives or non-natives, one would thus need to consider approaches that filter out foreign languages from their data during or after data collection more effectively.

Bibliography

[1] Bhumika, Sukhjit Sehra, and a Nayyar. A Review Paper on Algorithms Used for Text Classification. *Ijaiem*, 2(3):90–99, 2013.

[2] William B Cavnar, John M Trenkle, and Ann Arbor Mi. N-Gram-Based Text Categorization. *In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, 1994.

[3] David R. Cox. The regression analysis of binary sequences (with discussion). *J Roy Stat Soc B*, 20:215–242, 1958.

[4] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 233–240, New York, NY, USA, 2006. ACM.

[5] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[6] Tom Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, June 2006.

[7] George Forman. An Extensive Empirical Study of Feature Selection Metrics for Text Classification. *Journal of Machine Learning Research*, 3:1289–1305, 2003.

[8] T. Grant. Native Language Identification. page 1.

[9] Binod Gyawali, Gabriela Ramirez, and Thamar Solorio. Native Language Identification: a Simple n-gram Based Approach. *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 224–231, 2013.

[10] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2013.

[11] G. Hughes. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14(1):55–63, Jan 1968.

[12] Danesh Irani, Steve Webb, Calton Pu, Ferst Drive, and Boyd Gsrc. Study of Trend-Stuffing on Twitter through Text Classification. *CEASseventh annual Collaboration, Electronic messaging, AntiAbuse and Spam Conference*, page cited 11, 2010.

[13] Thorsten Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *Machine Learning*, 1398(LS-8 Report 23):137–142, 1998.

[14] Daniel Jurafsky and James H. Martin. Classification: Naive bayes, logistic regression, sentiment, Aug 2015.

[15] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[16] T. Li, C. Zhang, and M. Ogihara. A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics*, 20(15):2429–2437, 2004.

[17] Andres McCallum and Kamal Nigam. A Comparison of Event Models for Naive Bayes Text Classification. *AAAI/ICML-98 Workshop on Learning for Text Categorization*, pages 41–48, 1998.

[18] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 625–632, New York, NY, USA, 2005. ACM.

[19] Kamal Nigam, Andrew Kachites Mccallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2):103–134, 2000.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot,

and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[21] Nektaria Potha and Efstathios Stamatatos. *A Profile-Based Method for Authorship Verification*, pages 313–326. Springer International Publishing, Cham, 2014.

[22] PRAW. Praw: The python reddit api wrapper, 2016.

[23] Anand Rajaraman and Jeffrey David Ullman. Data Mining. *Mining of Massive Datasets*, 18 Suppl:114–142, 2011.

[24] Reddit. What's new on reddit: Happy 10th birthday to us! celebrating the best of 10 years of reddit, Jun 2015.

[25] Reddit. reddit.com: api documentation, 2016.

[26] Leonard Richardson. Beautiful soup.

[27] Marcal Rusinol and Josep Llados. Logo spotting by a bag-of-words approach for document categorization. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pages 111–115, 2009.

[28] Sam Scott and Stan Matwin. Text Classification Using WordNet Hypernyms. *Learning*, pages 45–51, 1998.

[29] Nakatani Shuyo. Language detection library for java, 2010.

[30] Sanasam Ranbir Singh, Hema a. Murthy, and Timothy a Gonsalves. Feature Selection for Text Classification Based on Gini Coefficient of Inequality. *JMLR: Workshop and Conference Proceedings 10 The Fourth Workshop on Feature Selection in Data Mining*, pages 76–85, 2010.

[31] Kent A. Spackman. Signal detection theory: Valuable tools for evaluating inductive learning. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 160–163, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[32] M Sudheep Elayidom, Chinchu Jose, Anitta Puthussery, and Neenu K Sasi. Text Classification for Authorship Attribution Analysis. *Advanced Computing: An International Journal (ACIJ)*, 4(5):1–10, 2013.

[33] Bruno Trstenjak, Sasa Mikac, and Dzenana Donko. KNN with TF-IDF based framework for text categorization. *Procedia Engineering*, 69:1356–1364, 2014.

[34] Peter D Turney. Thumbs up or thumbs down? Semantic Orientation applied to Unsupervised Classification of Reviews. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 417–424, 2002.

[35] S.M.J. Wong and Mark Dras. Contrastive analysis and native language identification. *... of the Australasian Language ...*, page 53, 2009.

[36] Baoxun Xu, Xiufeng Guo, Yunming Ye, and Jiefeng Cheng. An improved random forest classifier for text categorization. *Journal of Computers (Finland)*, 7(12):2913–2920, 2012.

[37] S. Yin, Z. Huang, L. Chen, and Y. Qiu. A approach for text classification feature dimensionality reduction and rule generation on rough set. In *Innovative Computing Information and Control, 2008. ICICIC '08. 3rd International Conference on*, pages 554–554, June 2008.

[38] Zhaohui Zheng, Xiaoyun Wu, and Rohini Srihari. Feature selection for text categorization on imbalanced data. *ACM SIGKDD Explorations Newsletter*, 6(1):80, 2004.