Roman Purgstaller, BSc

Dynamic N-Gram Based Feature Selection for Text Classification

MASTER'S THESIS

to achieve the university degree of
Master of Science
Master's degree programme: Software Development and Business Management

submitted to

Graz University of Technology

Institute of Interactive Systems and Data Science
Head: Univ.-Prof. Dr. Stefanie Lindstaedt

Supervisor: Dipl.-Ing. Roman Kern

Graz, August 2018

This document is set in Palatino, compiled with pdfLaTeX2e and Biber.

The LaTeX template from Karl Voit is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, __________________________ _______________________________

Date Signature

Eidesstattliche Erklärung[1]

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am __________________________ _______________________________

Datum Unterschrift

[1]Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

Abstract

Feature selection has become an important focus in machine learning. Especially in the area of text classification, using n-gram language models will lead to high dimensional datasets. In this thesis we propose a new method of dimensionality reduction. Starting with a small subset of features, an iterative forward selection method is performed to extend our feature space. The main idea is, to interpret the results from a trained classifier in order to determine feature importance. Our experimental results over various classification algorithms show that with this approach it is possible to improve prediction performance over other state of the art dimension reduction methods, while providing a more cost-effective feature space.

Contents

List of Figures

Todo list

1 Introduction

The focus of this thesis is to analyse the effects of feature selection for text classification. This area of research is already well studied. For example, Forman, 2003 applied various metrics, like χ^2 or Information Gain, which we will also cover in our work. In general, the aim of feature selection is not only to improve classification effectiveness and computational efficiency, but also to create machine learning models which are clearer and therefore easier to interpret. Especially when categorizing text data, feature selection has great potential. Consider for example the google n-gram corpus (Franz and Brants, 2006), which contains over one trillion words of running texts and over 13 million unique words, even when discarding words with a term frequency lower than 200.

One important characteristic of text classification is the high dimensionality of the feature space. Additionally, an important and extensively analysed concept in natural language processing tasks is the use of n-gram language models. Word n-grams are sets of co-occuring words within a given window, where the number of of words within a n-gram is determined by n.

In contrast to word tokenization, using n-gram language modeling will increase the size of the feature space even further. Especially when choosing a high value for n, the resulting feature space will blow up very fast. For this reason, feature selection is essential to limit the feature space as well as avoid overfitting.

A straight-forward approach for representing text is the *bag-of-words* approach, where each feature corresponds to a single word. In this paper we study the effect of using n-grams. In particular, we describe an algorithm which performs a discriminant analysis, choosing n-gram features dynamically. The identification of discriminant features is achieved using an iterative forward selection approach. The importance of features is determined using various types of ranking criteria. This is an often chosen approach (see for example I. Guyon and Elisseeff, 2003 or Yang and Pedersen, 1997), showing promising results.

We assume that, if a specific unigram is considered important, a bigram containing this unigram might also be meaningful for a specific class. This technique for discovering association rules between features is called apriori principle (Agrawal, Srikant, et al., 1994). For example, the sentence ”Risk free: earn easy money with bitcoin investments” includes several phrases indicating spam. Considering the word ”money” important for the class spam, The phrases ”easy money” and ”earn easy money” might also be equally or even more important. This information is then used for selecting features dynamically. Starting with unigrams, we consider a n-gram potentially important if a lower order n-gram exists in the corpus which is contained in the higher order n-gram and and this lower order n-gram is already selected by the ranking criterion.

Using this approach, we expect to identify specific meaningful phrases, while reducing the size of the feature space significantly, leading to a smaller, more expressive model.

Another interesting aspect we will cover in this thesis is using skip-gram tokenization. Considering the example above, this technique makes it possible to cover phrases like ”earn money”, ”money bitcoin” or even (after removing punctuation) ”risk free money”.

In the course of this thesis we will focus on two main questions:

1. Is it possible to compete with state of the art feature selection methods, while providing a reasonable small feature space?
2. What are good ranking criterions for this approach?

After covering the theoretical focus in chapter 2, we introduce the implemented algorithm in chapter 3. We will then outline the chosen datasets (chapter 4.1) as well as evaluation metrics (chapter 4). In chapter 3.3 we will focus on implementation details. Subsequently, we will present the experimental results in chapter 5. Finally, in the discussion section (chapter 6) we will review the results of our work and provide a future perspective as well as possible extensions.

2 Related Literature and Theoretical Focus

2.1 Feature Extraction

Text classifiers often don't use a characteristic representation of a language. Raw data such as documents consisting of a sequence of characters cannot be fed to a learning algorithm directly. Machine learning algorithms need a different form of input, often incoherent to humans, in order to make it possible to compute satisfying results.

Usually the input for a machine learning algorithm is represented as a vector of weighted features. We define a feature as a string within a document. Furthermore, the process of turning a corpus into numerical feature vectors is called vectorization. Each feature in a document is assigned a weight. More formally, a document d is represented by a vector $\vec{x} = (x_1, ..., x_M)$.

In this chapter we focus on how text is processed in order to create feature sets suitable for machine learning algorithms.

2.1.1 Tokenization

Tokenization is the task of chopping a sequence of characters up into pieces, called tokens, optionally throwing away certain characters, such as punctuation. Tokenization is needed to convert text into features.

An important issue in tokenization is finding the correct tokens to use. One tokenization approach is to split on whitespaces and throw punctuation characters away. Another possible strategy is splitting on all non-alphanumeric characters. Manning and Schütze, 1999 provides a survey on tokenization, pointing out that such strategies could lead to problems. For instance, some punctuation marks such as in *etc* should remain as part of the word. Another problem which could arise in this context are single apostrophes or hyphens. Imagine phrases like *I'm right* or *my friend's*. Splitting on non-alphanumeric characters would create create tokens like *I m right* and *my friend s*. There is no easy way for a tokenizer to differentiate in such cases. Jiang and Zhai, 2007 performed an interesting study on tokenization, defining heuristics for the English language on how to deal with the above mentioned cases. The heuristics included replacing non-functional characters with spaces (such as: ! ", #, %, $), but leaving hyphens in the text. The results suggested that such strategies would improve the performance significantly. The downside of those heuristics is that they are very language specific. Especially when considering the above mentioned cases on hyphens or single aprostophes.

2.1.2 Stop Word Removal

One important goal of feature engineering is finding the most significant words or phrases in documents. The intuition that the most frequent words appearing in a document are the most significant is true for many use cases, such as language detection. However, in this thesis, we focus on classification based on the subject matter of documents, limiting ourselfs to the english language. The most frequent words in the english language will most certainly be words with low significance such as "the" or "and". One common strategy to eliminate certain features containing low value is the use of stop lists. A stop list contains words which are filtered out before the tokenization process. However, stop word removal might lead to problems when significant phrases (such as "The Who"), or homonyms ("can" the verb vs. "can" the noun) are removed. As with addressing certain problems in tokenization, this approach is very language specific and certain techniques such as tf-idf term weighting might provide better results.

2.1.3 Bag-of-Words

Bag-of-words is an often used representation in text classification in which each feature corresponds to a single token. This method counts the occurrences of tokens in a document. The result is a vector in the space of features, which corresponds to tokens found in the text.

2.1.4 N-Gram Tokenization

Bag-of-word models are a simple and often very effective approach. However, important details about the original document, such as Phrases, word order, context and sentences, is lost. Alternatively, adding multi-word expressions can be helpful identifying certain multi-word expressions, such as "United Kingdom" or "white house".

N-grams are basically sequences of n consecutive words from a given text. For example, considering the following sentence: "My favourite treat is cheesecake", would create the following n-grams:

- **unigrams (n=1):** My, favourite, treat, is, cheesecake
- **bigrams (n=2):** My favourite, favourite treat, treat is, is cheesecake
- **trigrams (n=3):** My favourite treat, favourite treat is, treat is cheesecake

N-gram models are an important concept in natural language processing, carrying more information than its components and are often crucual in text classification. They are used in a wide area of application such as sentence completion, data compression, text classification or speech recognition (for example Cavnar, Trenkle, et al., 1994; Fürnkranz, 1998).

2.1.5 Skip-Gram Tokenization

Skip-grams (D. Guthrie et al., 2006) are a technique to overcome data sparsity. Additionally to the created n-grams, k-skip-n-grams are formed by "skipping" tokens and allowing adjacent sequences of words. For example, applying skip-grams to a trigram model using the sentence "I hit the tennis

ball" skipping the token "tennis" will also create the feature "hit the ball", which might be an equally important feature.
Formally, k-skip-n-grams for a sentence $w_1 ... w_m$ are defined as follows:

$$\{w_{i_1}, w_{i_2}, ..., w_{i_n} | \sum_{j=1}^{n} i_j - i_{j-1} < k\} \tag{2.1}$$

Where k is defined as the skip distance to construct the n-gram.

It is worth mentioning that skip-grams will generate many useless n-gram features and will blow up the feature space, especially when choosing a high value for n. D. Guthrie et al., 2006 demonstrates in their research that skip-grams can accurately model context while keeping misinformation to a minimum. The results furthermore suggest that skip-grams are especially useful when the test corpus is similar to the training corpus.

2.2 Issues in Feature Engineering

The encoding and selection of features for a learning algorithm is essential and can have an enormous impact on the resulting language model. In this section we discuss possible problems which could arise in the creating feature sets.

2.2.1 Signal and Noise

In general, signal refers to the underlying pattern we want to learn from data. In the context of text classification the main goal is to detect meaningful words and phrases from text. Noise refers to irrelevant information or randomness in a dataset. For text classification, there is no clear definition for the concept of noise. One obvious definition could be any form of text different from the intended, including spelling errors, errors from speech recognition or handwriting recognition. J Ross Quinlan, 1986 lists faulty measurement, ill-defined threshholds (e.g., when is a person "tall"?) and subjective interpretation (e.g., what criteria are used when describing a person as "athletic"?) as sources for noisy data. In general, a feature becomes a noise feature when it increases the classification error (Manning, Raghavan, Schütze, et al., 2008). This usually happens when rare terms with no information about a class appear especially in one class in the training data. A similar problem to noise is missing values. The impact on noise highly depends on the chosen feature representation. A simple bag of words representation might handle missing values better than a phrase-based representation. However, noise can have significant impact on classification performance (David Dolan Lewis, 1992; J Ross Quinlan, 1986).

2.2.2 Overfitting

A supervised learning model is usually trained in order to generalize to situations which didn't occur in the training data. A model is overfitted if it models the training data too well but fails to generalize to new examples.

On the other hand, it is possible that a training corpus is to simple to produce a representative model. In this case the model is underfitted. Underfitted models thend to have a high bias and low variance, while overfitted models usually have a high variance.

To balance the size of the training data is a well known problem in machine learning which is referred to as bias-variance tradeoff or bias-variance dilemma (Geman, Bienenstock, and Doursat, 1992). We will discuss this topic in more detail in section 2.5.2.

There are numerous ways to prevent overfitting such as cross-validation feature selection (section 2.4). Furthermore, many machine learning algorithms include techniques to constrain the complexity of the model. For example, decision trees (section 2.7) address overfitting by pruning the tree after the learning phase.

2.2.3 Data Sparsity

Allison, D. Guthrie, and L. Guthrie, 2006 define data sparsity as the phenomenon of not observing enough data in a corpus in order to model language accurately. Language is a system of rare events, various and complex, such that true observations cannot be made. Natural language processing typically gathers information from a corpus to create a probability distribution. In many cases, meaningful but unobserved features would be assigned zero probability because of insufficient data.

An often proposed approach to address data sparsity is using smoothing techniques. Smoothing takes some probability "mass" away from sequences seen before, in order to be assigned to sequences which have not been seen. Another very interesting approach to tackle the data sparsity problem is the use of skip-gram modelling (see section 2.1.5).

2.3 Feature Weighting

One very simple method of weighting features is to simply assign a boolean value to each term in the document indicating whether the term occurs in the document. This weighting strategy is called the Binary Independence Model (Robertson and K. S. Jones, 1976). "Independence" indicates that there are no associations recognized between terms, which is not correct for text classification. This assumption is also found in the Naive Bayes model which we will discuss in detail in section 2.6. As pointed out by Manning, Raghavan, Schütze, et al., 2008, one problem of binary assessments is that they do not capture any nuances. The relevance of a term in a document is an absolute decision. Other approaches are based on the following idea: terms which occur more often should receive a higher score. Therefore we could use the number of occurrences of a word as a weighting schema also known as the term frequency denoted by $tf_{t,d}$ for a term t in a document d.

Another popular term weighting method is the $tf - idf$ term weighting (Sparck Jones, 1972), where $tf - idf$ means term-frequency times inverse document-frequency. The measure is defined as follows:

$$tf - idf(t, d) = tf(t, d) * idf(t) \tag{2.2}$$

The idf is computed as

$$idf(t) = log\frac{1 + n_d}{1 + df(d,t)} + 1 \tag{2.3}$$

where n_d is the total number of documents and $df(d,t)$ is the number of documents containing term t.

2.4 Feature Selection

The fundamental idea of feature selection is to construct feature subsets that are useful for building a machine learning model while excluding irrelevant features (I. Guyon and Elisseeff, 2003). Especially in text classification, feature selection is an often studied problem (for instance: Yang and Pedersen, 1997; Joachims, 1998; Koller and Sahami, 1997) due to the high dimensionality of the feature space. The phenomen that, with a high number of features or dimensions we need more and more data to generalize accurately is called *the curse of dimensionality* and is well known in machine learning.

We define feature selection, often also refered as dimension reduction or feature subset selection, as the process of selecting a subset of d features from the original set of D features, where $d < D$.

In general, feature selection is performed for numerous reasons:

- **Reduced training times**
- **Increase prediction accuracy**
- **Better generalization**
- **Interpretability**: Feature selection reduces the complexity of a model and makes it easier to interpret

2.4.1 Filter Methods

Filter methods apply some statistical measure to assign a score to each feature. The resulting scores are then sorted and only the $k - best$ features are subsequently used for classification.

This is a widely used approach for feature subset selection in text classification. For example Yang and Pedersen, 1997 and Forman, 2003 give an extensive overview on various ranking criteria used in text classification. The methods are very effective in computation time, since only the computation and sorting of n scores is required. Additionally, methods are robust against overfitting by considerably reducing variance (Friedman, Hastie, and Tibshirani, 2001). The main disadvantage is that relationships and dependencies between features are ignored. Filter methods are mainly used as a preprocessing step, independent of the choice of the classification algorithm.

In this thesis we consider two popular filter methods, namely χ^2 (Section 2.4.1) as well as information gain (Section 2.4.1).

χ^2 Measure

The χ^2 test measures the independence of two events A and B. In our case A and B are the occurrence of a term and a document. The equation is defined as follows:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{2.4}$$

O_i is defined as the observed frequency of a term in a document, while E_i is the expected number of occurrences, assuming the feature occurence is independent of the class value.

Information Gain

One important concept in information theory is information gain (IG). It measures the amount of information, one random variable contains about another random variable.

Entropy measures the information content of a random variable. More precisely, entropy measures is a measure of unpredictability or impurity. For a discrete random variable X with alphabet χ and a probability mass function $p(x)$, we define the entropy as:

$$H(X) = -\sum_{x \in \chi} p(x) log_2 p(x) \tag{2.5}$$

Furthermore, the conditional entropy for a joint probability mass function $p(x, y)$ and two random variables X and Y is defined as:

$$H(X|Y) = -\sum_{x \in \chi} \sum_{y \in \gamma} p(x, y) log_2 p(x, y) \tag{2.6}$$

Finally, we define the information gain as:

$$IG(Y|X) = H(Y) - H(Y|X) \tag{2.7}$$

In machine learning information gain is often used for feature selection. IG reduces the uncertainty of a random variable Y by subtracting the knowledge we gain, once X is known.

2.4.2 Wrapper Methods

Wrapper methods (Kohavi and John, 1997) search for an optimal subset of features by searching through the space of feature subsets. Each subset is tried with the learning algorithm. The combination yielding the best performance is selected.

In particular, forward and backward selection are used for wrapper methods. In forward selection, each unselected feature is added to the feature subset. The feature with the highest increase in performance is choosen. The algorithm terminates when there is no more increase in performance. Backward selection methods start with the whole feature set and try to remove features. As with forward selection, this step is repeated until no performance improvement is observed.

Since wrapper methods are computationally very expensive, studies often use low dimensional feature sets (e.g. Kohavi and John, 1997; I. Guyon, Weston, et al., 2002). However, these studies often find that wrapper methods perform best (Forman, 2003).

Recursive Feature Elimination

The recursive feature elimination (RFE) method (I. Guyon, Weston, et al., 2002) is the process of repeatedly removing unimportant features. At first, a classifier is trained to determine the importance of each feature. The least important feature is then pruned from the feature set. This procedure is repeated, until the desired number of features is reached. In order to increase performance, it is also possible to remove more then one feature at each iteration. Although RFE was introduced using Support Vector machines, it has also been used with different classification algorithms (See for example Granitto et al., 2006).

2.5 Text Classification

Text classification is the task of assigning predefined classes to new documents. Furthermore, we define a classification task where a given object is assigned to one of two classes as binary or binomial classification or simply two-class classification.

Classification is a field which has many applications. For instance, it can be used to assign images into classes, the detection of spam e-mails or speech recognition.

In this study we focus on the task of classifying documents which is referred to machine learning-based text classification. The corpus is usually divided into training data and test data including the labels for each document, where labeling is the process of assigning one class to each document. Using supervised learning, the decision criterion of the classifier is learned automatically from the training data.

In the next section we describe the text classification problem more formally. In section 2.5.2 we deal with the bias-variance tradeoff, also known as bias-variance dilemma. Subsequently, we focus on classification algorithms which are used for our experiments. We begin with the Naive Bayes in section 2.6, a simple, yet very efficient and often used classification algorithm. Section 2.7 covers Decision Tree Learning. Support Vector machines are discussed in section 2.8.

2.5.1 The Text Classification Task

We define the text classification problem as the task of assigning a document d_i from the entire document space D to a class c from a fixed set of classes $C = \{c_1, c_2, ..., c_n\}$. A document may be assigned assigned to multiple classes, wich is often referred to as *any-of* problem or ranking classification (Manning, Raghavan, Schütze, et al., 2008; Ikonomakis, Kotsiantis, and Tampakas, 2005). In this thesis we focus on one-of problems where a document is a member of exactly one class.

2.5.2 The Bias-Variance Tradeoff

One important concept in machine learning is the bias-variance tradeoff. The bias defines the error from wrong assumptions the learning algorithm makes resulting from the lack of domain knowledge. Therefore, a high bias result in underfitting. For instance, linear models like Naive Bayes have a high bias for non-linear problems. On the other hand, a non-linear method such as the k-nearest neighbor tend to have a low bias (Manning, Raghavan, Schütze, et al., 2008). Variance, on the other hand, measures the inconsistency of a model. Machine learning models with high variance are typically accurate on average, but inconsistent.

The bias-variance tradeoff was introduced by Geman, Bienenstock, and Doursat, 1992. He argues that there is often a trade-off between the bias and variance to the estimation error. Variance is typically reduced by using smoothing which will on the other hand increse the bias. It is then a matter of weighting the assets of bias and variance to a specific application.

2.6 Naive Bayes Text Classification

The naive Bayes classifier is a very popular supervised learning method which is also often used in text classification tasks. An increasing number of studies (for instance David D Lewis and Ringuette, 1994; Koller and Sahami, 1997) have been published on this classification algorithm. The Bayesian approach for classifying a new instance is to assign the class with the highest probability. The most likely class is computed using the maximum a posteriori probability estimate:

$$c_{map} = \arg\max_{c \in C} \hat{P}(c|d) = \arg\max_{c \in C} \hat{P}(c) \prod_{1 \le k \le n_d} \hat{P}(t_k|c) \tag{2.8}$$

$\hat{P}$ indicates that we don't know the true values for parameters $P(c)$ and $P(t_k|c)$, but only the estimates from the training set.
Equation 2.8 multiplies many probabilities which could easily result in a floating point underflow. To avoid this problem, it is more common to add logarithms of probabilities instead of multiplying probabilities. Therefore, in most implementations of NB, the computation is done like this:

$$c_{map} = \arg\max_{c \in C} [log \hat{P}(c) + \sum_{1 \le k \le n_d} log\ \hat{P}(t_k|c)] \tag{2.9}$$

The conditional parameter $log\ \hat{P}(t_k|c)$ indicates how good the term t_k is for c. The prior log $\hat{P}(c)$ is the relative frequency of c (Manning, Raghavan, Schütze, et al., 2008).

2.6.1 Properties of Naive Bayes

One important aspect of the Naive Bayes (NB) classifier is the independence assumption, which states that the word probabilities for one text position are independent of the words that occur in other positions (Mitchell, 1997). Taking a closer look on text classification tasks and language in general, this is clearly not correct. Despite this limitation the NB classifier has proven to be very successful in many domains, including text classification problems (Domingos and Michael J Pazzani, 1996; David D Lewis, 1998; McCallum, Nigam, et al., 1998).

Domingos and M. Pazzani, 1997 argues the reason that NB often performs well in text classification tasks is that the probability estimation is often of low quality and estimates diverge significantly from the highest score but the “winning” class, which is also often the correct class in NB has a much larger probability than the other classes. Hence, despite the fact that the estimates are bad naive bayes often performs very well.

Another strength of NB is efficiency. The training phase as well as predictions can be accomplished with one pass over the data. This is obviously optimal because we have to look at the data at least once (Manning, Raghavan, Schütze, et al., 2008).

2.6.2 The Multinomial Naive Bayes

One learning method we use in the evaluation phase is the multinomial NB. The multinomial model simply counts the occurrences of words in the document representing the data as count vectors. The distribution is parametrized by vectors $\theta_y = \{\theta_{y1}, ..., \theta_{yn}\}$ for each class. The parameter θ_y is estimated by a smoothed version of the maximum likelihood estimator:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n} \tag{2.10}$$

where N_{yi} is the number of occurrences for feature i that appears in a document d of class y in the training set and N_y is the total count of all features for class y. α describes the smoothing parameter. One transformation to the count vectors is tf-idf term weighting, as described in section 2.3, which is also known to work well in practice (Buitinck et al., 2013).

As with language modeling, using the maximum likelihood estimation creates the problem of zero estimates leading to sparse data. One approach to overcome this problem is smoothing. Smoothing is a technique to adjust the maximum likelihood estimator. Probability is taken away from some occurrences and redistributed to other words. Smoothing is a well studied area of research (see for instance Song and Croft, 1999; J. T. Goodman, 2001). Chen and J. Goodman, 1996 provided a detailed study on smoothing techniques for language modeling. However, in this study we limit the smoothing techniques to lidstone smoothing (Lidstone, 1920; Johnson, 1932; Jeffreys, 1948) and Laplace smoothing (Manning, Raghavan, Schütze, et al., 2008). Setting $\alpha = 1$ is known as Laplace or additive smoothing and setting $\alpha < 1$ is refered to as Lidstone smoothing.

Laplace smoothing eliminates the problem of zero probabilities, however, Chen and J. Goodman, 1996 pointed out that phrases which didn't occur in the training corpus such as *burnish the* and *burnish thou* will be assigned the same probability, although the latter is far more likely. In fact, Gale and Church, 1994 performed a study on the add-one smoothing with the expressive title "What's Wrong with Adding One?". They argue that adding-one performs even worse than MLE and that estimators such as the Good-Turing estimate (Good, 1953) should be preferred. Nevertheless additive smoothing is an often used technique for the NB classifier as well as language modelling. Often Laplace smoothing is used, despite the fact that other smoothing techniques have shown better results (Manning, Raghavan, Schütze, et al., 2008; Chen and J. Goodman, 1996). As already mentioned above there are severial variants of NB classifier, but it has been shown for text categorization problems, the multinomial model is most often the best choice (Eyheramendy, David D Lewis, and Madigan, 2003; McCallum, Nigam, et al., 1998).

2.6.3 The Bernoulli Model

An alternative model to the Multinomial Naive Bayes is the Bernoulli model which generates binary valued feature vectors, either 1 indicating the presence of a term in the document or 0 indicating the absance.

The decision rule for Bernoulli naive Bayes is defined as:

$$P(x_i|y) = P(i|y)x_i + (1 - P(i|y))(1 - x_i) \tag{2.11}$$

where P is the conditional probability of class y containing term x_i.

By ignoring the number of occurences, the Bernoulli model typically makes mistakes when classifying long documents whereas it has to be proven to be more stable on short documents (Buitinck et al., 2013; Manning, Raghavan, Schütze, et al., 2008). However, high variance in document length might cause problems because it is simply more likely for a word to appear in a longer document (McCallum, Nigam, et al., 1998).

2.7 Decision Tree Learning

Decision tree learning is a method for approximating discrete-valued target functions. The learned function is represented by a decision tree. A decision tree is a tree-like graph or model where features are represented by attribute-value pairs. Each (non-leaf) node specifies a test of some attribute of the instance, each branch corresponds to a possible value of the attribute (the outcome of the test) and each leaf note corresponds to a class label. Learned trees can also be represented as sets of if-then rules. The classification process starts at the root node of the tree, testing all attributes specified by this node and moving down to the corresponding tree branch. This process is repeated until a class label is found (Mitchell, 1997)

One advantage of decision trees is, that it consists of a white box model. This means, in contrast to a black box model like a neural network, each decision can be explaind using boolean logic. In terms of performance, the prediction time is logarithmic in the number of data points (i.e. the depth of the tree).

However, Decision trees tend to overfit. Mechanisms such as pruning or limiting the tree size in advance are often used to avoid this issue.

The focus on the remainder of this section lies on discussing different tree creation algorithms as well as

2.7.1 Tree Algorithms

The Iterative Dichotomiser 3

The basic idea behind the ID3 (short for Iterative Dichotomiser 3) (J. Ross Quinlan, 1986) is building the tree iteratively by choosing the best attribute. Trees are grown to their maximum size. This process is is a greedy algorithm, not including any backtracking to reconsider earlier choices. The selection criterion for the ID3 algorithm is called information gain, measuring how well a given attribute separates the training examples according to their target classification. Before we define information gain, we first need to define another quality measure, defining the purity of of an arbitrary collection of examples, which is called entropy. The entropy of a collection of examples S is defined as follows:

$$Entropy(S) = \sum_{y} -p_y log_2 p_y \tag{2.12}$$

where p_y is the proportion of S belonging to class y. The information gain, measuring the effectiveness of an attribute is defined as:

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \tag{2.13}$$

where $Values(A)$ is the set of all possible values for attribute A and S_v is the subset of S for for which attribute A has value v (i.e. $S_v = \{s \in S | A(s) = v\}$) (Mitchell, 1997). The ID3 algorithm has some practical issues, such as overfitting resulting from the fact that it grows each branch of the tree perfectly fitting the training data. One common technique to overcome this problem is post-pruning of the decision tree (Mitchell, 1997). J. Ross Quinlan, 1987 describes other methods to deal with overfitting. For example, reduced-error pruning replaces each node with the best possible leaf (i.e. the most popular class). The change is kept if the new tree would give an equal or fewer number of errors. Another method is called cost-complexity pruning. This strategy generates a series of trees $T_0 ... T_m$ where T_0 is the original tree and T_M is just the root. The subtree minimizing the error rate is chosen for removal. The best tree is chosen by generalized accuracy by a training set or cross-validation. However, there are a large variety of extensions to the basic ID3 algorithm. In the next two sections (sections 2.7.1 and 2.7.1) we discuss two very popular algorithms derived from the basic ID3 algorithm.

The initial definition of ID3 only uses discrete valued attributes. This restriction can be removed so that continuous-valued decision attributes can be used in learning the tree. This is done by dynamically creating a threshold c and a new boolean attribute A_c for a continuous-valued attribute A. A_c is true if $A < c$ and otherwise false. The threshold is picked in a way that produces the greatest information gain. This is accomplished by sorting the examples according to the attributes and then identifying adjacent examples that differ in their target classification. A set of candidate thresholds lying midway between the corresponding values of A can then be generated. Fayyad, 1992 shows that the value of c that maximizes the information gain must lie in such a boundary. Extensions to this approach, splitting attributes into multiple intervals have been discussed by Fayyad and Irani, 1993. Another strategy is defining features by thresholding linear combinations of several continuous-valued attributes (Utgoff, 1991; Murphy and Michael J. Pazzani, 1994).

The C4.5 and C5.0 Algorithm

The C4.5 algorithm (J Ross Quinlan, 2014) is the successor to ID3. The algorithm can be summarized by the following steps:

1. Build the decision tree from the training set, allow overfitting to occur.
2. Convert the learned tree into an equivalent set of if-then rules by creating one rule for each path from the root node to a leaf node.

3. Prune each rule by removing preconditions that result in improving its estimated accuracy.
4. Sort the rules by their estimated accuracy in which they should be applied.

Rules are always be pruned in a way that the removal doesn't worsen the estimated accuracy. It is common to estimate the rule accuracy by using a data set disjoint from the training data. Another strategy, used by C4.5, is the evaluation of the training data itself by using a pessimistic estimate. For this purpose the rule accuracy over the training examples is calculated followed by the standard deviation in this estimated accuracy assuming a binomial distribution. The lower-bound estimate is then taken as the measure of rule performance.

The C5.0 is Quinlan's latest version of the algorithm. While using less memory then the C4.5, it improves accuracy. (Buitinck et al., 2013; R. Pandya and J. Pandya, 2015)

The CART Algorithm

CART (short for Classification and Regression Trees) is very similar to the C4.5 algorithm. The algorithm was first introduced by Olshen, Stone, et al., 1984. It differs in that it supports numerical target variables (regression) and does not compute rule sets. Instead, CART constructs binary trees based on an exhaustive search of all possibilities. Choosing the decision criteria for splitting the tree has a high impact on accuracy.

2.7.2 Feature Importance

Decision trees are built top-down from a root, partitioning the data into subsets. This is done by choosing a decision critierion for splitting nodes. Common strategies include the gini index (Gini, 1912) and information gain (equation 2.12).

Like entropy, the gini impurity is a measure of impurity. It is defined as:

$$G = 1 - \sum_{c=1}^{C} p_c^2 \tag{2.14}$$

where p_c is the probability of a class.

The feature importance for each node is defined as the total decrease in node impurity. This measure is also often refered to as gini importance (Breiman et al., 2005). It can be computed using the following equation:

$$N_t / N * (G - N_{tR} / N_t * G_R - N_{tL} / N_t * G_L) \tag{2.15}$$

where N is the total number of samples, N_t is the the number of samples at the current node and N_{tR} and N_{tL} are the left and right children.

Decision trees can often create very complex structures which will result in overfitting. Techniques such as pruning or limiting the depth of a tree are used to avoid this issue. However, when using decision trees for feature selection, expressive trees might be more desireable to obtain as much information as possible.

2.7.3 Ensemble Methods and Random Forest Classifiers

Ensemble methods combine several machine learning models into one predictive model to produce better results. The aim of this technique is to decrease the variance or bias of a single model.

Two very common techniques to achieve this goal are called bagging and boosting. Bagging is used to reduce the variance by creating several subsets of the training data randomly. Each subset data is then used to train a classifier. The predictions from the different models in the ensemble are averaged to create a more stable learner. While bagging builds each model independently, boosting builds a new learner sequentially, taking the success of the previous model into account.

Random forests are an extension over bagging. A Random forests creates a set of decision trees by selecting subsets of the training data as well as subsets of features randomly. The best split for each tree is therefore picked among a random subset of features. While this results in an increasing bias, the variance also decreases, which usually results in a better model overall (Buitinck et al., 2013).

The feature importance for random forests is computed for each tree as described in section 2.7.2 and then averaged over all trees in the ensemble.

2.8 Support Vector Machines

Support vector machines (SVMs) (Boser, I. M. Guyon, and Vapnik, 1992; Vapnik, 1995) were introduced as a supervised learning model used for classification, regression and outliers detection. SVMs try to find a decision plane (or boundary) drawn in the middle of the void between data points and therefore maximizing the margin between data points in different classes. In general, SVMs can solve linear and nonlinear problems. In this study we experiment only with a linear kernel function. Other kernel functions are more complex to weight, since they are transformed in another dimensional space. The possibility to use nonlinear kernel functions is discussed in section 6.

Given training data of n points as pairs $(\vec{x}_i, y_i)$. In case of linear separable training data, the hyperplane is written as

$$\vec{w} \cdot \vec{x} - b = 0 \tag{2.16}$$

where $\vec{w}$ is the normal vector to the hyperplane and b is a bias. Assuming training data which is not linear separable, linear SVM solves the following optimization problem:

$$\min_{w} \quad \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi(w; x_i, y_i) \tag{2.17}$$

where $\xi(w; x_i, y_i)$ is the loss function and C is a penalty parameter. The two common loss functions are referred to as L1-SVM, which is defined as $max(1 - y_i, w^T x_i, 0)$, and the L2-SVM, written as $max(1 - y_i w^T x_i, 0)^2$. It is possible to include a bias term b by augmenting the vector w and each instance x_i with an additional dimension: $w^T \leftarrow [w^T, b], x_i^T \leftarrow [x_i^T, B]$, where B is a constant value (Fan et al., 2008).

For binary classification, a data point $\vec{x}$ is predicted as positive if $\vec{w}^T \vec{x} > 0$ and negative otherwise. For multi-class classification, we applied the one-vs-the-rest strategy (Hsu, Chang, Lin, et al., 2003) in the evaluation phase.

SVMs in general are very effective in high dimensional space and work well with sparse document vectors which is characteristic for text classification. For instance, Joachims, 1998 applied SVMs to text classification and discovered that SVM outperforms all other classification methods.

2.8.1 Feature Importance

A linear SVM creates a hyperplane that uses support vectors to maximise the distance between classes. The coefficients of the classifier represent the vector coordinates which are orthogonal to the hyperplane. The direction of the coefficients indicates the predicted class. The absolute size of coefficients in relation to each other can be used to determine the importance of features. Those values represent the feature weights.

The feature weights for selecting features are the weights given by equation 2.17. In the evaluation phase a L1-regularized SVM is used for feature ranking. This Setting will result in very sparse weight matrices and is often used to identify important features (Fan et al., 2008). For that reason, we expect that this setup will perform best when selecting a very small number of features.

2.9 Evaluation of Text Classification

One objective when using a text classification algorithm is the minimization of classification error on the test data. However, measuring accuracy is often considered a bad approach, especially for skewed datasets. A skewed dataset is a dataset where one class is over-represented. A classification task might seem to perform well by simply never predicting small classes (i.e. the percentage of documents in the class is very low), and still receive a high accuracy. There are multiple ways to deal with skewed datasets. For example, stratified sampling creates training and test data where classes are well balanced.

For model evaluation other measures than accuracy are better suited, namely precision (equation 2.18), recall (equation 2.19) and the F_1 measure (equation 2.20). Precision tells us the fraction of retrieved documents which are relevant, while recall is the fraction of relevant documents that are retrieved. More formally:

$$precision = \frac{tp}{tp + fn} \tag{2.18}$$

$$recall = \frac{tp}{tp + fp} \tag{2.19}$$

where tp (*true positives*) is the number of documents correctly assigned as positive by the classification model, fn ($false\ negative$) is the number of incorrectly assigned negative results and fp ($false\ positive$) is the number of incorrectly assigned positive results. The F_1 measure (introduced by Van Rijsbergen, 1979) is the weighted average of precision (p) and recall (r). It is computed as:

$$F_1 = \frac{2pr}{r + p} \tag{2.20}$$

2.9.1 Binary and Multi-class Evaluation

Some metrics, such as F_1 score are defined for binary classification. However, it is possible to extend those metrics for multi-class classification. In our study we use a macro-average F_1 score, which calculates the mean of all binary metrics, weighting all classes equally. The macro-average is well suited for our purpose since we know that the text corpus used for multi-class classification is partitioned roughly evenly (see section 4.1.1) over all classes. Other measures, such as micro-average or weighted F_1 score are preferred if classes are uneven in size.

3 Algorithm

In this chapter, we present the dynamic n-gram based feature selection algorithm for supervised learning problems. First, we will discuss how the basic forward selection method works and how it is implemented. Afterwards we will discuss different parameter settings for performance tuning.

3.1 N-Gram Based Feature Selection

The main goal of this study is to select features dynamically, considering the importance of features given an estimator that assigns feature weights. We refer to the subset of features, which are considered important by the external estimator, as support.

Based on the ranking criterion, an iterative forward selection method is applied for feature subset selection: First, we start with a subset of the feature space, consisting of all unigrams. The algorithm then performs the following steps for each n in our n-gram range, starting with $n = 2$:

1. The external estimator is trained on the feature subset in order to obtain the importance of each feature.
2. The estimators support is used to extend our feature space by higher order n-grams (e.g. bigrams for $n = 2$). A n-gram is added to the feature space if any support feature is a subset of the n-gram.

We define a n-gram as a subset of a higher order n-gram if the n-gram is contained in the higher order n-gram, i.e. the n-gram is a *subset* of the higher order n-gram. Vice versa, a higher order n-gram might be a superset of a n-gram. (e.g. " *coffee*" is a subset of "*drinks coffee*"). Based on this definition we define the selection criterion as:

$$\exists x \in support : x \subset feature \tag{3.1}$$

Algorithm 1 provides a representation in the form of pseudocode for the described feature selection method.

In the following sections we will evaluate the performance of the proposed algorithm, given various ranking criteria, using different input parameters. We will focus on the following corner points:

1. How can we determine the importance of features
2. What is a good ranking criterion for our feature selection algorithm.
3. How is hyper parameter tuning influencing our results
4. Can our algorithm compete with state of the art feature selection methods

3.2 Variations

In our study we experimented with different variations of the presented algorithm. In the following two sections we introduce some modifications of the presented algorithm and how we expect the results to change.

3.2.1 Alter the Selection Criterion

We can alter the selection criterion from our algorithm, defined in equation 3.1, such that it is necessary that both sub-features must be contained in the current support in order to be selected. More formally we define the alternative selection criterion as:

$$\exists x, y \in support : (x, y) = feature \tag{3.2}$$

Algorithm 1 n-gram based feature selection

```
Require: $X$: Training examples - the document-term matrix, $y$: Class labels
    (i.e.the target vector relative to $X$)
 1: function FIT(X,y)
 2:     $n \leftarrow$ The upper bound of the n-gram range
 3:     $estimator \leftarrow$ The scoring estimator that provides the feature importance
 4:     $num\_components \leftarrow$ The percentage of features to select during each iteration.
 5:     $n\_grams \leftarrow$ Contains a list of extracted n-grams for each n
 6:     $prune\_zero\_score \leftarrow$ if true, features with score zero are pruned at each iteration
 7:
 8:     $selection \leftarrow n\_grams[1]$
 9:     $X\_transformed \leftarrow$ transformed version of X, containing all unigrams
10:     $estimator.fit(X\_transformed, y)$
11:     if $prune\_zero\_score$ then
12:         Remove features from $current\_selection$ with score eqal to zero
13:
14:     for $i = 2$ to $n$ do
15:         $support \leftarrow estimator.get\_support()$
16:         $current\_selection \leftarrow array()$
17:         for $feature\ in\ n\_grams[n]$ do
18:             if $\exists x \in support : x \subset feature$ then
19:                 $current\_selection \leftarrow current\_selection \cup feature$
20:         $selection \leftarrow selection \cup current\_selection$
21:         $X\_transformed \leftarrow$ transformed version of X,
22:                                 containing all features from $current\_support$
23:         $estimator.fit(X\_transformed, y)$
```

This alteration will result in a much smaller feature space. Moreover, it is even possible that no feature will be selected at all for the next iteration. One possibility is to combine this method with skip-grams (see 2.1.5). We will discuss the results of this approach in chapter 5

3.3 Implementation Details

3.3.1 Machine Learning Tools

The proposed algorithm is implemented using the programming language python. Furthermore, data mining and data analysis frameworks are used for evaluation. Next to various essential python libraries, mainly numpy and Scipy (Stéfan van der Walt and Varoquaux, 2011), we would like to mention some frameworks in particular. Classification algorithms are based on the scikit-learn machine learning library (Pedregosa et al., 2011). We also used NLTK (Loper and Bird, 2002) for various preprocessing tasks such as sentence tokenizing. Graphics are created using matplotlib (Hunter, 2007).

3.3.2 Feature Extraction

Scikit-learn already offers an iterface for extracting and transforming the 20newsgroups. The Reuters-21578 dataset on the other hand, was downloaded directly from the UCI machine learning repository and transformed manually.

3.3.3 Data Transformation

Next to a document-term matrix and a target vector, n-Gram based feature selection requires the information, which features to select for each iteration (i.e. the subsets for all feature) from the current support. To store this information a $m * m$ matrix is used, where m is the number of features. Since we expect this matrix to be sparse, the data structure we use is a sparse matrix. Although for constructing, a linked list sparse matrix is used (E. Jones, Oliphant, Peterson, et al., 2001–). The process of constructing this matrix is executed before the actual feature selection and completely encapsulated. Hence, there is no context between feature selection and text data. This approach has multiple advantages. First of all, the memory cost is very low. The whole model can therefore be easily saved, refitted and copied, which especially comes in handy when using Grid Search and Cross Validation. Considering time complexity, since sparse matrices are only saving the indices of non zero values, accessing a particular feature from the feature space can be done in $O(1)$. Additionally, by encapsulating this step, our feature selection method is not limited to text classification tasks, but can also be used on any data with similar characteristics.

4 Empirical Evaluation

In this chapter we provide the results from testing different input parameter settings for the proposed feature selection algorithm as well as for the classification algorithm used for evaluating feature selection performance. The main goal is to find good values for k, the percentage of features selected from our algorithm in each iteration. Furthermore we will look at the different variations introduced in section 3.2.

4.1 Data Sets

4.1.1 The 20 Newsgroups Dataset

The 20 newsgroups dataset (collectecd by Lang, 1995) consists of approximately 20000 newsgroups post on 20 topics. It can be directly downloaded using the python library scikit-learn (Stéfan van der Walt and Varoquaux, 2011). The classes are partitioned roughly even. We removed headers and footers from the samples in the preprocessing phase.

One interesting characteristics of this dataset is, that some classes are closely related to each other (e.g. *rec.autos* and *rec.motorcycles*) while others are highly unrelated (e.g. *sci.crypt* and *soc.religion.christian*). Considering only a subset of those classes might have a not to be underestimated impact on evaluation results. For this dataset, multi-class classification is used. One subset of the whole corpus used in the evaluation section is limited to 4 classes:

- *alt.atheism*
- *comp.graphics*

- *talk.religion.misc*
- *sci.space*

The dataset consists altogether of 2034 documents and roughly 26500 different tokens.

In another set of experiments we used the following set of classes:

- *alt.atheism*
- *comp.graphics*
- *comp.os.ms-windows.misc*
- *comp.sys.ibm.pc.hardware*
- *comp.sys.mac.hardware*
- *comp.windows.x*
- *misc.forsale*
- *rec.autos*
- *rec.motorcycles*
- *rec.sport.baseball*
- *rec.sport.hockey*

We also use the full 20 newsgroups dataset to compare the results from our experiments. In addition, we perform experiments on a preprocessed version of the 20 newsgroups dataset (Cardoso-Cachopo, 2007). The preprocessed version also uses stopwords, words with less than 3 characters are removed and the whole dataset is stemmed. There are also additional adjustments like substituting tabs and newlines by spaces. Furthermore, only letters are kept, punctuation, numbers etc. are turned into spaces. For this dataset, we applied $tf - idf$ term weighting (see section 2.3).

4.1.2 The Reuters Dataset

We downloaded the Reuters-21578 dataset from the UCI machine learning repository (Lichman, 2013). The extracted data consists of 6463 documents and about 25800 different tokens. In contrast to the 20 Newsgroups dataset, we use binary classification, distinguishing between the topic *acquisition* and all other categories.

4.2 Document Data Preparation

The raw training data we use for evaluation is already vectorized and weighted before it can be used for a classification task. For feature weighting we test the term frequency (TF) in combination with the inverse document frequency (IDF). Additionally, we experiment with skip-gram vectorization, different term-removal thresholds as well as a stop word list for the English language (Buitinck et al., 2013). We tested different ranking criteria with various hyper parameter settings and then tested the performance with various classification tasks.

4.3 Evaluation Metrics

After applying our feature selection method, the resulting subset is evaluated using various classification algorithms. To evaluate the performance of feature selection methods we will use the measures precision, recall, F_1, and accuracy. For multiclass classification we use macro averaged precision, recall and F_1 measure, since the 20newsgroups dataset is evenly partitioned (See section 4.1.1).

4.4 Binary and Multiclass Classification

For classification tasks like the naive bayes classifier or linear SVM, the model coefficients are used to predict feature importance. For binary classification, each feature is given a weight, representing the importance for each feature. However, in multiclass classification, the coefficients are are given in the form of a $n * k$ matrix, where k is the number of features and n is the number of classes. In this case, feature coefficients are simply summed up.

4.5 Hyper Parameter Tuning

For hyper parameter tuning we use an exhaustive Grid-search on specific input parameters, testing every possible combination of parameter values. In addition, we use randomized search for some classification algorithms to determine good input parameter values (Buitinck et al., 2013).

V-fold cross validation is performed on the dataset, dividing the dataset in v equaly sized subsets. One subset is tested using a classifier trained on the remaining $v - 1$ subsets. The overall performance is the average of all computed folds. This is a common practice to avoid overfitting (Buitinck et al., 2013; Hsu, Chang, Lin, et al., 2003). For grid search and random search we use a 3-fold cross validation for model evaluation.

5 Experimental Results

5.1 Hyper Parameter Tuning

5.1.1 Term Weighting with Naive Bayes Classification

We analyse the effect of smoothing the classifier using a Grid-search with cross validation. Using our feature selection algorithm, we start a n-gram range of one to three. Smoothing is only used on the classification algorithm. Using smoothing on the ranking criterion has no effect, since the order of the features doesn't change.

In our first set of experiments we use the Bernoulli NB classifier for feature weighting as well as for classification. We used the 20newsgroups dataset as well as a trigram vectorization. This setup performs very poorly, but can be improved slightly when altering the selection criterion as discussed in section 3.2.1, as can be seen in figure 5.1.

The best results for Bernoulli feature weighting is achieved using the altered selection criterion. Using this setup, performance reaches its peak when selecting about 30% of all features for each iteration. However, for $k > 0.3$, the model is apparently overfitted, but can be stabilized by choosing a small value for α.

The same input setting for term weighting as well as for classification with a multinomial NB performs better overall, as can be seen in figure 5.2.

Interestingly, when choosing a high value for k, in contrary to the Bernoulli Naive Bayes, the multinomial Naive Bayes works better with Laplace smoothing. We can observe a similar pattern when using 2-skip-3-gram vectoriza-

	Term frequency vectorization				2-skip-3-gram vectorization			
k	acc	recall	precision	F_1	acc	recall	precision	F_1
10%	0.771	0.750	0.774	0.750	0.776	0.754	0.780	0.754
20%	0.801	0.777	0.811	0.776	0.798	0.769	0.814	0.766
30%	0.824	0.808	0.825	0.808	0.813	0.787	0.821	0.786
40%	0.829	0.812	0.825	0.813	0.820	0.798	0.824	0.796
50%	0.825	0.802	0.824	0.805	0.819	0.792	0.815	0.792
60%	0.826	0.802	0.829	0.805	0.832	0.810	0.830	0.813
70%	0.830	0.812	0.830	0.813	0.832	0.811	0.831	0.813
80%	0.831	0.811	0.828	0.814	0.832	0.814	0.828	0.816
90%	0.828	0.812	0.823	0.814	0.807	0.794	0.799	0.793
100%	0.829	0.815	0.822	0.816	0.776	0.771	0.775	0.766

Table 5.1: Performance of Multinomial Naive Bayes term weighting with varying vectorizing methods using the 20newsgroups corpus. Multinomial Naive Bayes with Laplace smoothing was used for classification.

tion. However, as we can see in table 5.1 using skip-gram vectorization with the multinomial NB does not improve performance.

Extending the 20newsgroups dataset to 10 categories will reduce the models performance slightly. Table 5.2 shows the performance of both datasets.

Using binary classification with the Reuters dataset will result in a very similar pattern. While Laplace smoothing provides the best performance overall in this setting, choosing a small value for alpha will lower performance up to 40% (figures).

In conclusion, while a Bernoulli model performs very poor for term weighting as well as for classification, multinomial Naive Bayes works well with Laplace smoothing, although we were not able to severely improve results using various input parameters.

5.1.2 Term Weighting with Support Vector Classification

Using support vector classification for feature ranking, we examined the influence of different regularization functions as well as the penalty param-

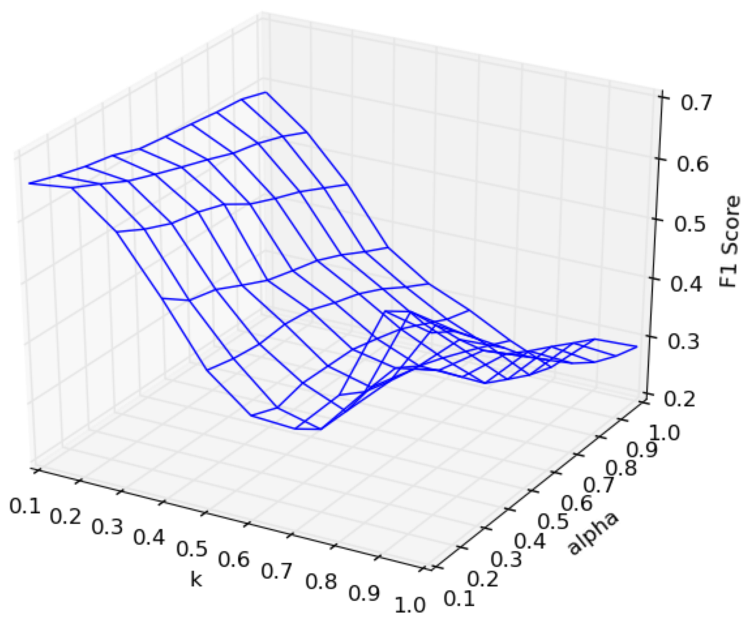

Figure 5.1: Bernoulli Naive Bayes feature weighting using Grid Search with Cross Validation and macro averaged F-scoring. The 20newsgroups dataset is 2-skip-3-grams vectorized. k is the percentage of selected features in each iteration and alpha the additive smoothing parameter.

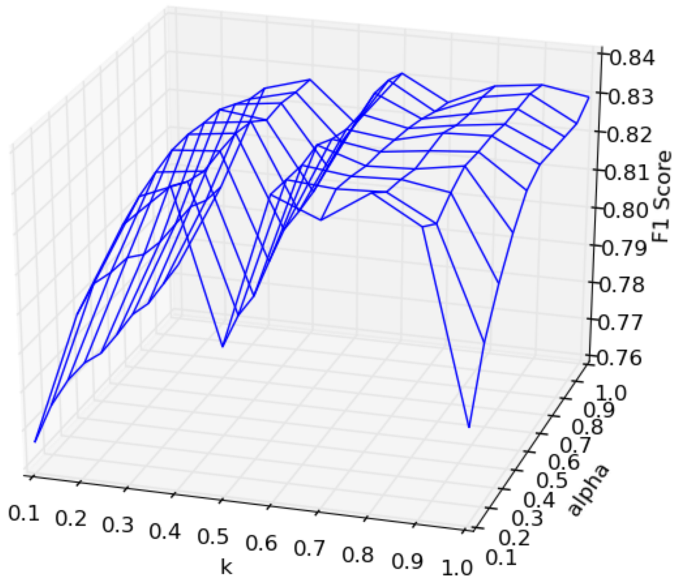

Figure 5.2: Multinomial Naive Bayes feature weighting using Grid Search with Cross Validation and macro averaged F-scoring. The used dataset was 20newsgroups, trigram vectorized. k is the percentage of selected features in each iteration and alpha the additive smoothing parameter.

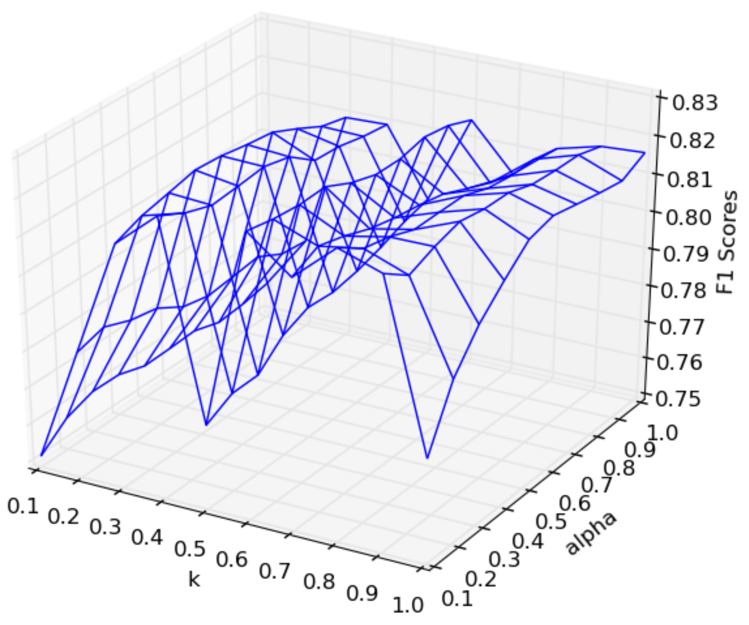

Figure 5.3: 2-Skip 3-Gram Multinomial Naive Bayes Grid Search with Cross Validation and altered selection criterion using a macro averaged F-measure. The 20newsgroups dataset is 2-skip-3-grams vectorized. k is the percentage of selected features in each iteration and alpha the additive smoothing parameter.

	20NG, 4 categories				20NG, 10 categories			
k	acc	recall	precision	F_1	acc	recall	precision	F_1
10%	0.771	0.750	0.774	0.750	0.679	0.683	0.724	0.656
20%	0.801	0.777	0.811	0.776	0.698	0.701	0.748	0.676
30%	0.824	0.808	0.825	0.808	0.708	0.711	0.758	0.690
40%	0.829	0.812	0.825	0.813	0.708	0.712	0.760	0.693
50%	0.825	0.802	0.824	0.805	0.710	0.713	0.763	0.697
60%	0.826	0.802	0.829	0.805	0.711	0.714	0.762	0.701
70%	0.830	0.812	0.830	0.813	0.713	0.716	0.767	0.704
80%	0.831	0.811	0.828	0.814	0.716	0.719	0.769	0.706
90%	0.828	0.812	0.823	0.814	0.721	0.723	0.768	0.707
100%	0.829	0.815	0.822	0.816	0.723	0.725	0.767	0.708

Table 5.2: Performance of Multinomial Naive Bayes term weighting for two subsets of the 20newsgroups dataset. Multinomial Naive Bayes with Laplace smoothing was used for classification.

eter C. Fan et al., 2008 argues that L1-regularization is often used to identify important features. Therefore, for feature selection, we use L1-regularized L2-loss Support Vector classification (SVC). Initial experiments show that for classification a L2-regularized L2-loss SVC works best. In order to identify good values for the penalty parameter C of the proposed ranking method we conduct a grid search with cross validation on $C = 2^{-5}, 2^{-3}, ..., 2^{13}$. A similar setting was also used by Fan et al., 2008 in order to identify good values for C. Considering the results shown in figure 5.4, setting C to approximately 0.5 seems like a good choice, especially when choosing a small value for k.

We also analyse the impact the impact on the penalty parameter C on the classification algorithm. In this experiment, we execute a Grid search using a SVC for feature weighting as well as classification. For instance, figure 5.5 shows the result of the experiment using the Reuters datasets. For both, the 20newsgroups dataset as well as the Reuters data, for feature weighting as well as classification, a very small value for C is preferable.

Overall, The results of both experiments indicate that a hyperplane with a large margin is more robust and therefore preferable over correctly separating as many instances as possible.

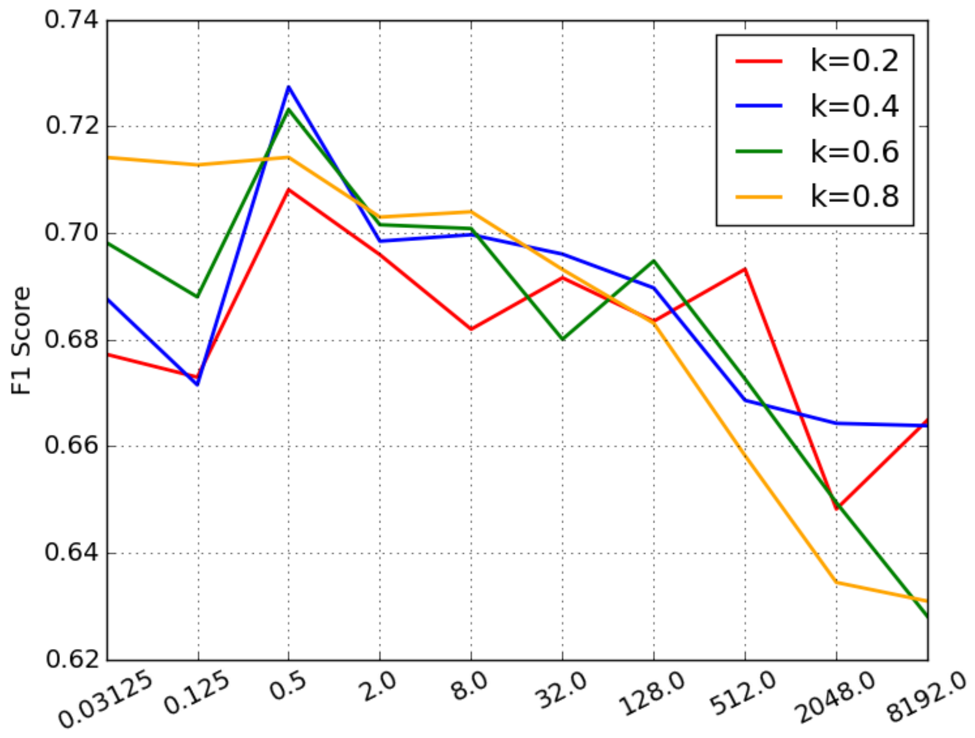

Figure 5.4: Support Vector Classification feature weighting. Grid search on penalty parameter $C = 2^{-5}, 2^{-3}, ..., 2^{13}$

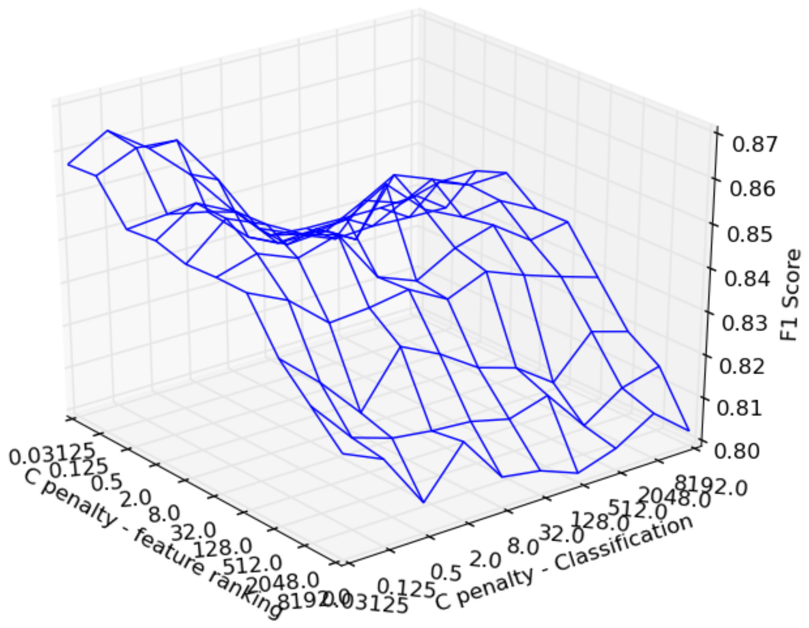

Figure 5.5: Support Vector Classification feature weighting using the Reuters dataset. A Grid search is performed on penalty parameter $C = 2^{-5}, 2^{-3}, ..., 2^{13}$ for feature weighting as well as classification.

	Gini Impurity				Information Gain			
split	acc	recall	precision	F_1	acc	recall	precision	F_1
2	0.772	0.763	0.763	0.761	0.772	0.762	0.763	0.760
5	0.776	0.767	0.766	0.765	0.766	0.757	0.757	0.755
10	0.777	0.767	0.767	0.765	0.771	0.762	0.762	0.759
20	0.772	0.762	0.762	0.760	0.772	0.763	0.764	0.761

Table 5.3: Performance of feature selection using Decision tree term weighting with gini impurity and information gain as splitting criteria. k is set to 30%. For classification a Multinomial naive bayes classifier was used. The 20newsgroups dataset was used for classfication. Split refers to the minimum number of samples required to split an internal node.

5.1.3 Term Weighting with Decision Tree Classifiers

We evaluate various parameter settings for a decision tree classifier using an optimized version of the CART algorithm 2.7.1. As already discussed in section 2.7.2, in the feature selection phase we prefer a rather expressive decision tree, hence avoiding pruning strategies or limiting the maximum depth of the tree.

In a first set of experiments we evaluate the inpact of information gain and the gini impurity as ranking criterions as well as different settings for the minimum number of samples required for splitting a node. The results for the 20newsgroups dataset are shown in tables 5.3 and 5.4. Both settings use decision trees for term weighting, but with different classifiers. While decision tree classification performs moderate, naive bayes classification provides better results overall. Interestingly, using binary classification with the Reuters dataset, decision tree classification performs better, while naive bayes classification provides average results (tables 5.5, 5.6).

5.1.4 Term Weighting with Information Gain and χ^2

In another set of experiments we use Information Gain as well as the χ^2 formula for term weighting with our feature selection algorithm and compared the results by simply selecting the same percentage of features according to the highest score.

	Gini Impurity				Information Gain			
split	acc	recall	precision	F_1	acc	recall	precision	F_1
2	0.617	0.593	0.601	0.592	0.619	0.593	0.599	0.592
5	0.611	0.592	0.602	0.591	0.605	0.580	0.588	0.578
10	0.608	0.587	0.594	0.585	0.604	0.581	0.591	0.580
20	0.616	0.596	0.607	0.595	0.615	0.591	0.598	0.588

Table 5.4: Performance of feature selection using Decision tree term weighting with gini impurity and information gain as splitting criteria. k is set to 30%. For classification a decision tree classifier was used. The 20newsgroups dataset was used for classfication. Split refers to the minimum number of samples required to split an internal node.

	Gini Impurity				Information Gain			
split	acc	recall	precision	F_1	acc	recall	precision	F_1
2	0.737	0.830	0.653	0.649	0.731	0.827	0.651	0.644
5	0.738	0.830	0.654	0.650	0.735	0.831	0.653	0.647
10	0.741	0.834	0.655	0.653	0.734	0.831	0.653	0.647
20	0.737	0.831	0.654	0.650	0.730	0.828	0.651	0.643

Table 5.5: Performance of feature selection using Decision tree term weighting with gini impurity and information gain as splitting criteria. k is set to 30%. For classification a multinomial naive bayes classifier was used. The Reuters dataset was used for classification. Split refers to the minimum number of samples required to split an internal node.

	Gini Impurity				Information Gain			
split	acc	recall	precision	F_1	acc	recall	precision	F_1
2	0.929	0.823	0.842	0.832	0.927	0.812	0.838	0.824
5	0.928	0.819	0.840	0.829	0.930	0.819	0.847	0.832
10	0.925	0.812	0.835	0.822	0.929	0.825	0.842	0.833
20	0.928	0.821	0.840	0.830	0.929	0.825	0.843	0.834

Table 5.6: Performance of feature selection using Decision tree term weighting with gini impurity and information gain as splitting criteria. k is set to 30%. For classification a decision tree classifier was used. The Reuters dataset was used for classification. Split refers to the minimum number of samples required to split an internal node.

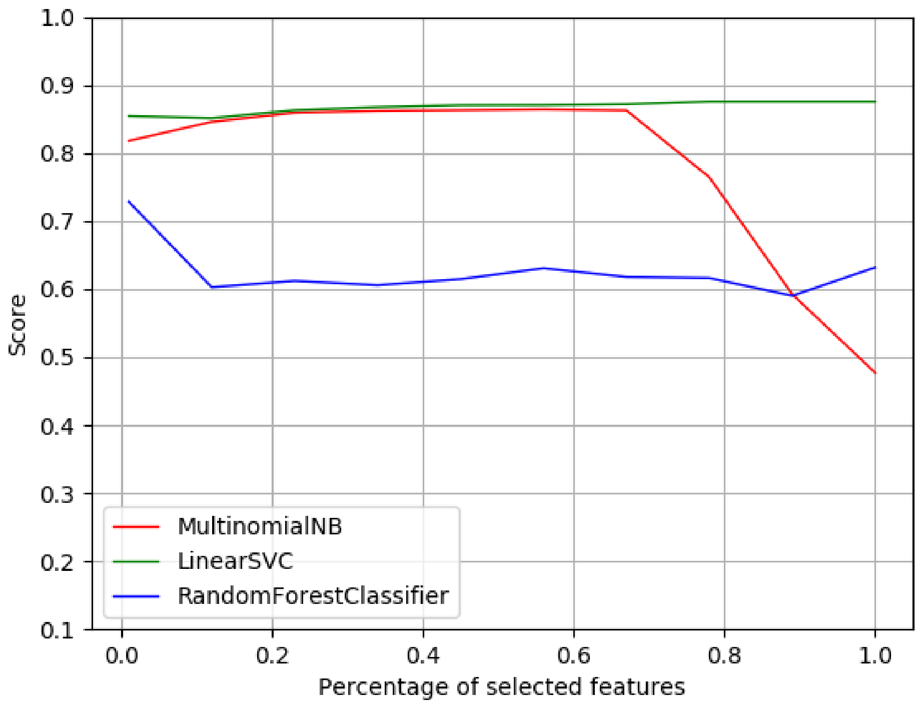

Figure 5.6: Selecting a percentage of features according to the highest score using the chi^2 formula. The used dataset is the Reuters news corpus.

The results for both selection criteria are similar. Overall, the multinomial NB outperforms SVC and Random Forest classification. The classification performance doesn't improve when choosing a higher value for k. (See for instance, figures 5.6 and 5.7).

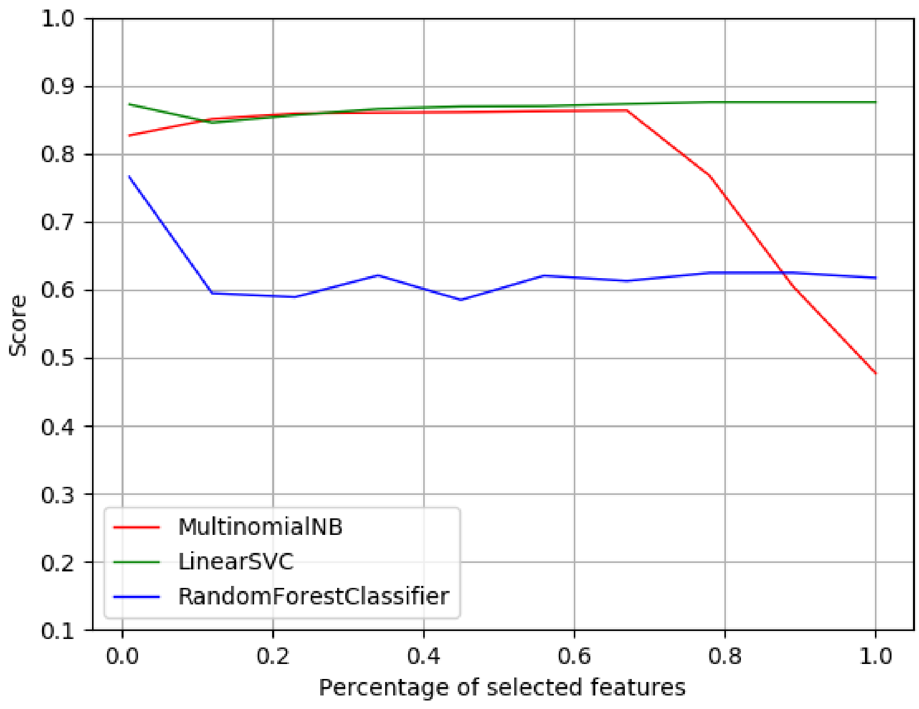

Figure 5.7: F_1 Scores selecting a percentage of features according to the highest score using information gain. The used dataset is the Reuters news corpus.

5.2 Comparison of Individual Results

5.2.1 Experimental Methods

After experimentation with different hyper parameter settings, our aim is the comparision of individual result using the insights we gained from hyper parameter tuning. We compare not only different variations of the proposed feature selection algorithm, but also current state-of-the-art feature selection algorithms. For this purpose we studied a variety of vectorization methods, feature selection algorithms and classifiers.

As datasets, we use the 20 newsgroup as well as the Reuters corpus, as presented in section 4.1.1 and section 4.1.2.

For feature selection we chose the following weighting methods for our feature selection algorithm:

- Multinomial NB.
- Linear L1-regularized L2-loss SVC, setting the penalty parameter C to 0.03125.
- Random Forest Classification using the gini impurity
- χ^2

The results from the previous section indicate that using information gain for term weigthing is very similar to χ^2 term weighting. Hence, in this section we consider only χ^2 for term weighting.

For comparision, we used the following feature selection algorithms:

- χ^2 feature selection
- Recursive Feature Elimination using the same SVC as above.
- Feature selecting based on feature weights using logistic regression.

Finally, we evaluated the performance on the feature selection methods above using a variety of classifiers, as listed below:

- Multinomial and Bernoulli NB
- Random forest classifier
- Linear L2-regularized L2-loss SVC

- Logistic regression

The hyper parameters are partially set, using the results from the previous section. Additionally, we dynamically chose the best input parameters for the proposed classification algorithms, using either grid search or randomized search depending on the size of the parameter space.

For the 20newsgroups dataset and trigrams, we achieved the best results using Logistic Regression for feature selection in combination with a multinomial NB classifier. In general, multinomial NB outperformed all other classification methods for this setting. Using the proposed dynamic selection approch, we achieved the best results with a RFC as weighting method. We were also able to achieve good results using SVC for term weighting. The altered selection criterion, introduced in section 3.2.1 works best using SVC. Although we were not able to improve classification results considerably, we achieved a much lower prediction latency. Table 5.10 shows the 20 best results in terms of the F_1 Score for this setting.

The results for the full 20newsgroups dataset are very similar. Dynamic n-gram based feature selection works best when choosing a random forest classifier for term weighting (see table 5.8).

We applied the same experiments for the preprocessed version of the 20newsgroups dataset from Cardoso-Cachopo, 2007. Overall, the results are considerably better using this dataset. In contrary to the other analysed datasets, χ^2 and NB term weigthing performs slightly better than RFC. The results of the experiment are shown in table 5.9.

Using the Reuters dataset, we achieved the best classification results using SVC. As with the 20newsgroups dataset, Logistic Regression also works very well for classification. In terms of feature selection, N-gram based feature selection using a random forest classifier for term weigthing performs best overall.

Feature selection	Classifier	F_1 score	accuracy	features
Log. Reg.	NB [1]	0.782561	0.786401	76152
-	NB	0.770966	0.777531	373168
Dyn. n-gram - RFC[2]	NB	0.767238	0.771619	76325
Log. Reg.	Log. Reg.	0.741831	0.750924	76152
Dyn. n-gram - SVC [3]	NB	0.741618	0.741316	13292
Dyn. n-gram - SVC	NB	0.741519	0.744272	50104
RFE - SVC[4]	NB	0.732922	0.741316	111950
-	Log. Reg.	0.730248	0.740576	373168
Dyn. n-gram - NB	Log. Reg.	0.730228	0.741316	51385
Dyn. n-gram - RFC	Log. Reg.	0.729956	0.738359	76325
Dyn. n-gram - χ^2	NB	0.729680	0.751663	79224
χ^2	NB	0.724561	0.748707	111950
χ^2	Log. Reg.	0.720919	0.736881	111950
Dyn. n-gram - NB	NB	0.720588	0.743533	51385
Dyn. n-gram - χ^2	Log. Reg.	0.711534	0.726534	79224
RFE - SVC	Log. Reg.	0.700547	0.711013	111950
Dyn. n-gram - RFC	SVC	0.696227	0.701404	76325
Log. Reg.	SVC	0.685725	0.691796	76152
-	SVC	0.680514	0.685144	373168
Dyn. n-gram - SVC	Log. Reg.	0.679555	0.685883	50104

Table 5.7: The best classification results in terms of the F_1 score using a subset of the 20 newsgroups dataset. The *features* column shows the number of features selected by the feature selection algorithm.

[1]For this set of experiments, the Multinomial NB is used

[2]N-gram based feature selection with using Random Forest Classification for term weighting

[3]N-gram based feature selection using the altered selection criterion

[4]Rekursive Feature Elimination with SVC Term weighting

[5]For this set of experiments, the Multinomial NB is used

[6]N-gram based feature selection with using Random Forest Classification for term weighting

[7]Rekursive Feature Elimination with SVC Term weighting

[8]N-gram based feature selection using the altered selection criterion

[9]For this set of experiments, the Multinomial NB is used

[10]N-gram based feature selection with using Random Forest Classification for term

Feature selection	Classifier	F_1 score	accuracy	features
Log. Reg.	Log. Reg.	0.648463	0.646840	374983
-	NB	0.643686	0.661445	1971375
Dyn. n-gram - NB [5]	Log. Reg.	0.643241	0.642326	300019
-	Log. Reg.	0.643193	0.642061	1971375
Dyn. n-gram - RFC[6]	Log. Reg.	0.642345	0.641264	418025
Dyn. n-gram - χ^2	Log. Reg.	0.639186	0.642459	437363
χ^2	Log. Reg.	0.636894	0.639671	591412
Dyn. n-gram - RFC	NB	0.635910	0.653346	418025
Log. Reg.	NB	0.634737	0.654142	374983
Dyn. n-gram - NB	NB	0.631816	0.649894	300019
Log. Reg.	SVC	0.631117	0.632634	374983
Dyn. n-gram - RFC	SVC	0.629034	0.628917	418025
-	SVC	0.622626	0.624270	1971375
Dyn. n-gram - χ^2	NB	0.618491	0.644981	437363
RFE - SVC[7]	Log. Reg.	0.618286	0.617631	591413
Dyn. n-gram - NB	SVC	0.617102	0.617499	300019
Dyn. n-gram - SVC	NB	0.615484	0.625863	272908
Dyn. n-gram - SVC	Log. Reg.	0.613662	0.611259	272908
χ^2	NB	0.609816	0.634626	591412
Dyn. n-gram - SVC[8]	Log. Reg.	0.609032	0.607807	65133

Table 5.8: The best classification results in terms of the F_1 score using the full 20 newsgroups dataset. The *features* column shows the number of features selected by the feature selection algorithm.

Feature selection	Classifier	F_1 score	accuracy	features
Dyn. n-gram - χ^2	SVC	0.854810	0.856270	423705
Dyn. n-gram - NB[9]	SVC	0.854696	0.855871	458304
χ^2	SVC	0.854154	0.855739	612854
Log. Reg.	SVC	0.853119	0.854277	442126
-	SVC	0.853093	0.854676	2042849
Dyn. n-gram - RFC[10]	SVC	0.847313	0.848698	448432
-	NB	0.840164	0.842588	2042849
Dyn. n-gram - χ^2	NB	0.839029	0.841259	423705
Log. Reg.	NB	0.837775	0.840462	442126
χ^2	NB	0.836918	0.839400	612854
Dyn. n-gram - NB	NB	0.835274	0.837938	458304
Dyn. n-gram - RFC	NB	0.828884	0.832359	448432
RFE SVC[11]	SVC	0.773804	0.775638	612855
RFE SVC	NB	0.769421	0.774442	612855
-	Log. Reg.	0.761076	0.777630	2042849
Log. Reg.	Log. Reg.	0.757224	0.774309	442126
χ^2	Log. Reg.	0.757102	0.774044	612854
Dyn. n-gram - NB	Log. Reg.	0.755979	0.772981	458304
Dyn. n-gram - χ^2	Log. Reg.	0.753748	0.770988	423705
Dyn. n-gram - SVC	SVC	0.738068	0.740170	238880

Table 5.9: The best classification results in terms of the F_1 score using a preprocessed version of the full 20 newsgroups dataset. The dataset is preprocessed using stemming and removing all characters but letters. The dataset is tf-idf vectorized with trigrams. The *features* column shows the number of features selected by the feature selection algorithm.

Feature selection	Classifier	F_1 score	accuracy	features
Dyn. n-gram - RFC	SVC	0.952895	0.954131	159040
-	SVC	0.952193	0.953503	740029
Log. Reg.	SVC	0.951784	0.953189	132732
RFE - SVC	SVC	0.948790	0.950047	222009
Dyn. n-gram - χ^2	SVC	0.947334	0.950361	222009
χ^2	SVC	0.947036	0.950047	222008
Dyn. n-gram - SVC	SVC	0.945531	0.946591	89399
Dyn. n-gram - RFC	Log. Reg.	0.945334	0.948476	159040
Dyn. n-gram - NB	SVC	0.945186	0.948790	80835
Log. Reg.	Log. Reg.	0.945113	0.948476	132732
-	Log. Reg.	0.944665	0.948162	740029
RFE - SVC	Log. Reg.	0.943039	0.946277	222009
Log. Reg.	NB	0.942874	0.943764	132732
Dyn. n-gram - NB	Log. Reg.	0.940695	0.945335	80835
Dyn. n-gram - χ^2	Log. Reg.	0.940397	0.945020	222009
χ^2	Log. Reg.	0.940397	0.945020	222008
Dyn. n-gram - SVC[12]	Linear SVC	0.938770	0.940308	16931
Dyn. n-gram - SVC	Log. Reg.	0.938669	0.942821	89399
Dyn. n-gram - SVC[13]	Log. Reg.	0.936822	0.940936	16931
Dyn. n-gram - RFC	NB	0.934470	0.939994	159040

Table 5.10: The best classification results in terms of the F_1 score using the reuters dataset. The *features* column shows the number of features selected by the feature selection algorithm.

In another set of experiments we compare how the number of selected features impacts the classification results. We compare three feature selection methods:

- Dynamic n-gram based feature selection with random forest classification for term weighting
- χ^2 Feature selection

weighting

[11]Rekursive Feature Elimination with SVC Term weighting

[12]N-gram based feature selection using the altered selection criterion

[13]N-gram based feature selection using the altered selection criterion

- Recursive feature elimination using support vector classification for term weighting.

In 5.8, 5.10 and 5.11 we show the results using the 20newsgroups and the Reuters dataset respectively. Especially when looking at the results for the Reuters dataset, using dynamic n-gram based feature selection will perform best when selecting a very small amount of features. χ^2 feature selection, on the other hand, performs better using a higher percentage of selected features. Especially, when using the Reuters dataset, dynamic n-gram based feature selection reaches its peak using a very small percentage of features.

5.3 Runtime Evaluation

In this chapter we analyse the training and prediction time on various feature selection methods using different input settings. The main goal is to illustrate the impact of feature selection on the computational performance, especially the prediction time of classification algorithms.

For the experiments in this chapter we used the 20newsgroups dataset with trigram vectorization.

Regarding execution runtime for dynamic n-gram based feature selection, the time complexity is primarily dependent on the chosen feature weighting method. We analyse the feature reduction time for the most promising weigthing methods, according to the previous sections and compare it with other feature selection methods used in this thesis. More precisely, we compare the dynamic n-gram based feature selection algorithm with recursive feature elimination, logistic regression feature selection and χ^2 feature selection. For the former two, we use a linear L1-regularized L2-loss SVC as well as RFC for term weigthing. The penalty parameter C for linear regression and SVC is set to 0.03125, the tolerance for the stopping criteria is set to 0.0001. The results of the experiment is shown in table 5.11. As expected, training random forests is rather slow, since the computational complexity of random forests using the CART algorithm for M random trees is $O(MKN^2logN)$, where N denotes the number of samples and K the number of features randomly drawn (Louppe, 2014). The computational

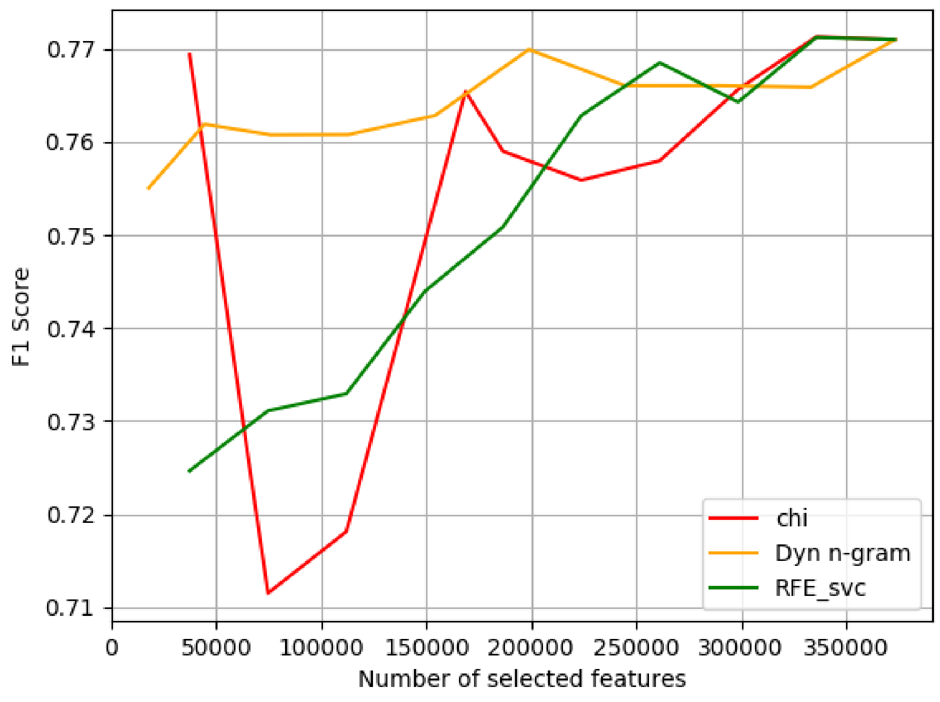

Figure 5.8: Impact on classification performance, depending on the number of selected features, using various feature selection methods. The selected 20newsgroups dataset is vectorized using trigrams. For dynamic n-gram based feature selection, a random forest classifier was used for term weighting. For classification, a multinomial Naive Bayes classifer with $\alpha = 0.1$ was used. *RFE_svc* stands for Recursive feature elminiation using a linear L1-regularized L2-loss SVC for term weighting.

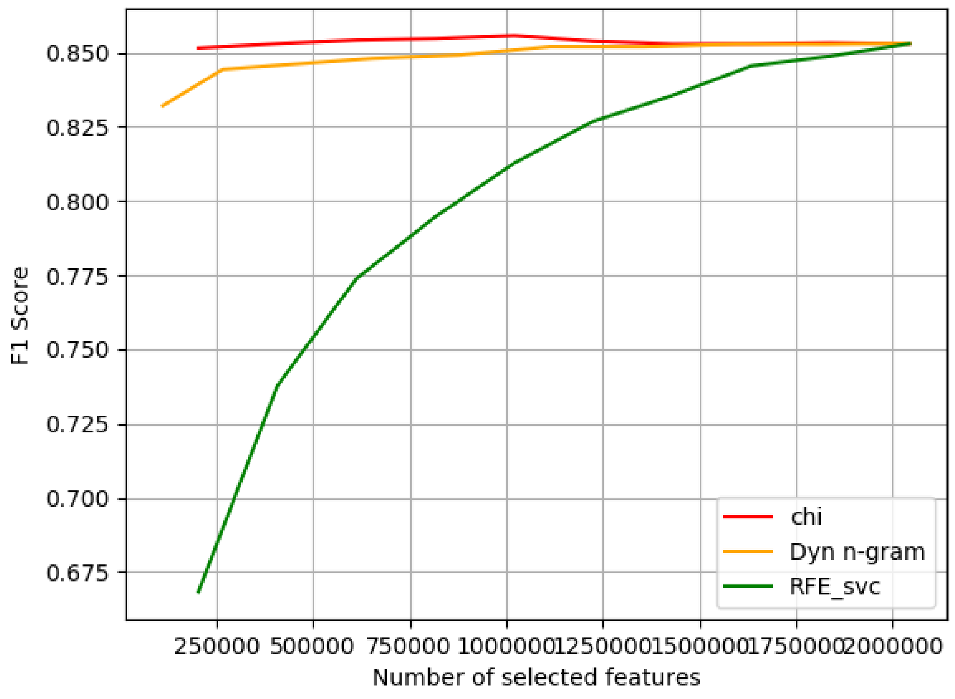

Figure 5.9: Impact on classification performance, depending on the number of selected features, using various feature selection methods. The 20 newsgroups dataset is preprocessed using stemming and removing all characters but letters. The dataset is tf-idf vectorized with trigrams. For dynamic n-gram based feature selection, a random forest classifier was used for term weighting. For classification, a linear L2-regularized L2-loss SVC was used. *RFE_svc* stands for Recursive feature elminiation using a linear L1-regularized L2-loss SVC for term weighting.

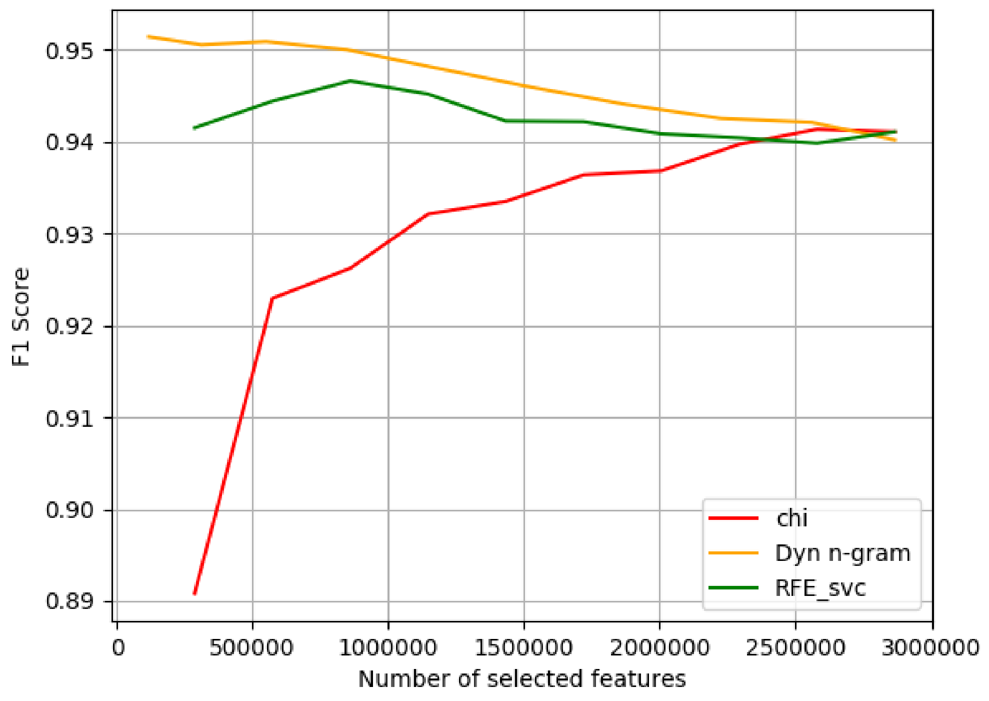

Figure 5.10: Impact on classification performance, depending on the number of selected features, using various feature selection methods. We used the Reuters dataset and performed 2-skip-3-grams vectorization. For dynamic n-gram based feature selection, a random forest classifier was used for term weighting. For classification, a linear L1-regularized L2-loss SVC was used. *RFE_svc* stands for Recursive feature elminiation using a linear L1-regularized L2-loss SVC for term weighting.

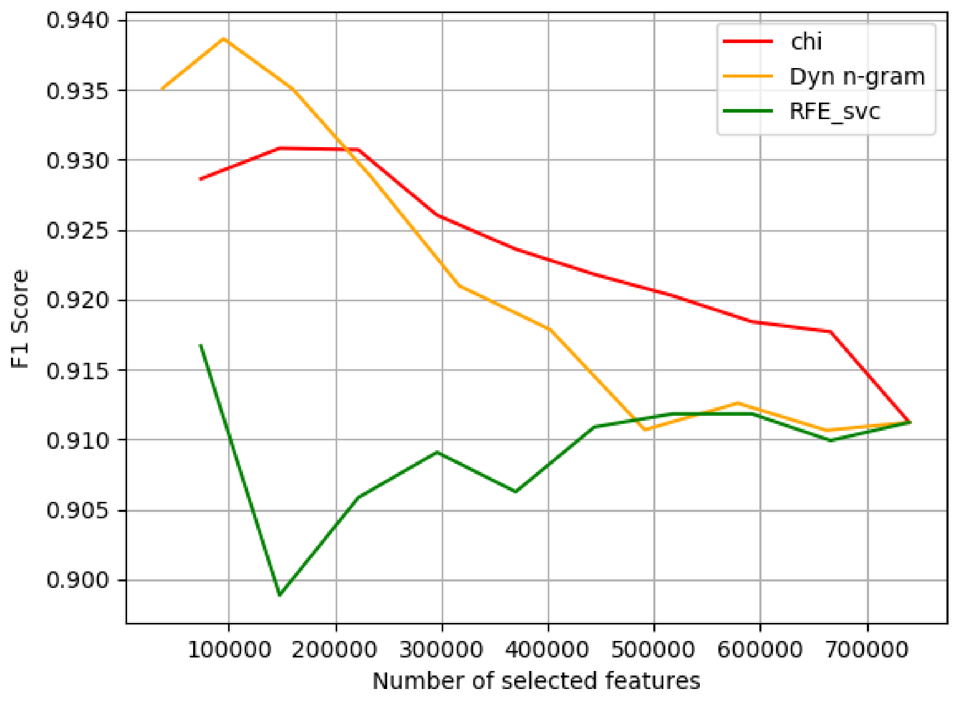

Figure 5.11: Impact on classification performance, depending on the number of selected features, using various feature selection methods.The selected Reuters dataset is vectorized using trigrams. For dynamic n-gram based feature selection, a random forest classifier was used for term weighting. For classification, a multinomial Naive Bayes classifer with laplace smoothing was used. *RFE_svc* stands for Recursive feature elminiation using a linear L1-regularized L2-loss SVC for term weighting.

Feature selection	execution time (seconds)
Dyn. n-gram feature selection with RFC for term weighting	57,47
Dyn. n-gram feature selection with SVC for term weighting	15.27
Recursive feature elimination with RFC for term weighting	520.07
Recursive feature elimination with SVC for term weighting	238.78
Logistic Regression	300.82
χ^2	2.74

Table 5.11: Execution time for feature selection algorithms on the full 20newsgroups dataset using trigram vectorization. For term weighting a linear L1-regularized L2-loss SVC was used with penalty parameter C set to 0.03125. The tolerance tolerance for the stopping criteria for linear SVC and logistic regression is set to 0.0001.

complexity for support vector machines depends on the number of support vectors and therefore also on the penalty parameter C. Bottou and Lin, 2007 argues, that with a small value for C the time complexity is at least n^2. Additionally, the actual runtime is also dependend on the actual implementation and optimization strategies. Here, recursive feature elimination removes 10% of features at each iteration. The algorithm can be improved in terms of execution time by increasing the percentage of features to remove. This might, however, reduce classification performance.

Another important aspect of feature selection is the reduction of training time a classification algorithm uses. For instance, by reducing the feature space from 1971375 to 418025 with dynamic n-gram based feature selection and RFC for term weighting, we are able to reduce training time considerably, as shown in table 5.12.

Finally, we analysed the latency at which predictions can be made. The prediction latency is defined as the number of predicitons the machine learning model can make in a given amount of time. For this purpose we benchmarked the prediction time for various machine learning models. Figure 5.13 shows predictions in bulk while figure 5.12 shows atomic predictions (i.e. one by one).

	Training time (seconds)	
Classification	Reduced feature space	Full feature space
NB	0.418	1.681
SVC	95.776	422.652
Log. Reg.	86.602	562.332
RFC	20.148	125.173

Table 5.12: Influence on the time needed to train various classifiers on the full trigram vectorized 20newsgroups dataset using feature selection. For feature selection, dynamic n-gram based feature selection with RFC for term weighting was used.

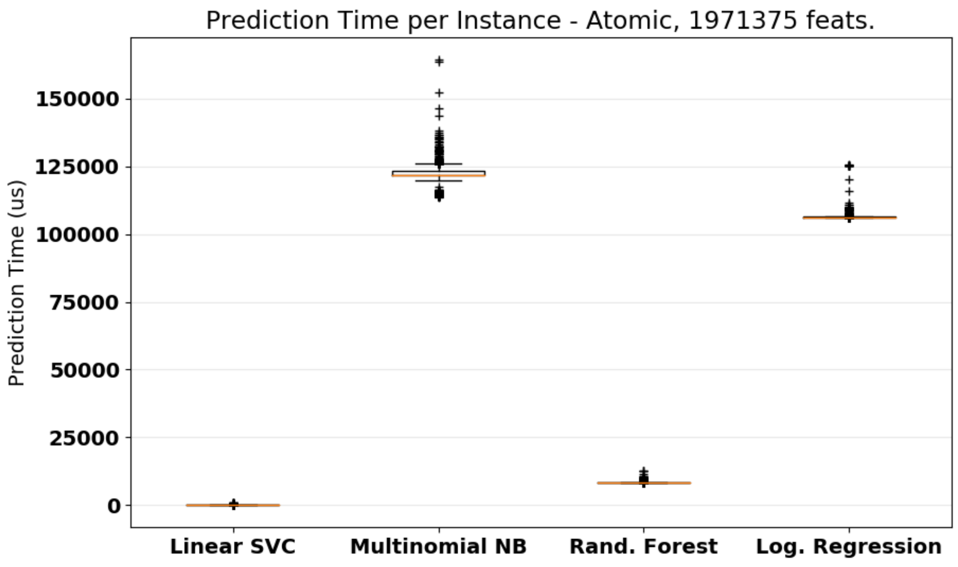

Figure 5.12: The latency for various classification algorithms doing atomic predictions (i.e. one by one). The 20newsgroups dataset with trigram vectorization was used.

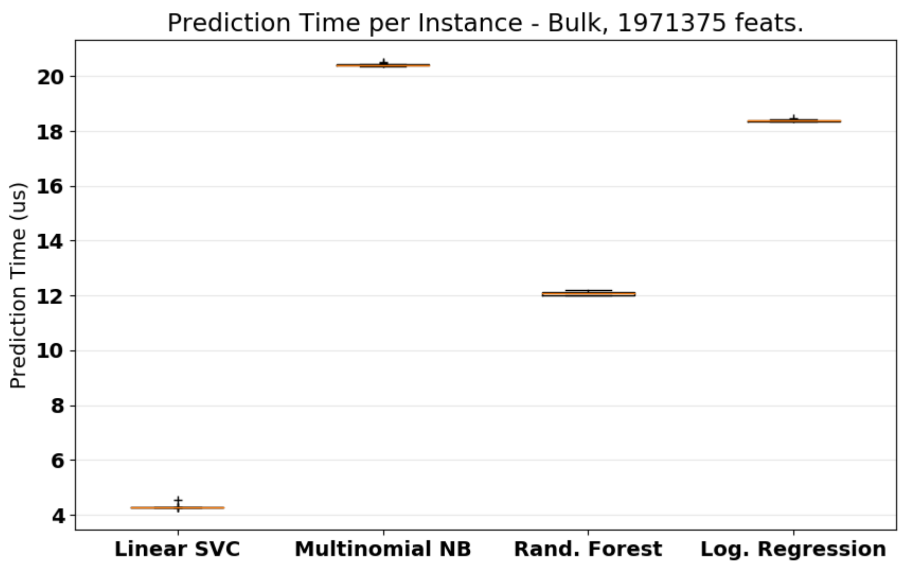

Figure 5.13: The latency for various classification algorithms doing predictions in bulk. The 20newsgroups dataset with trigram vectorization was used.

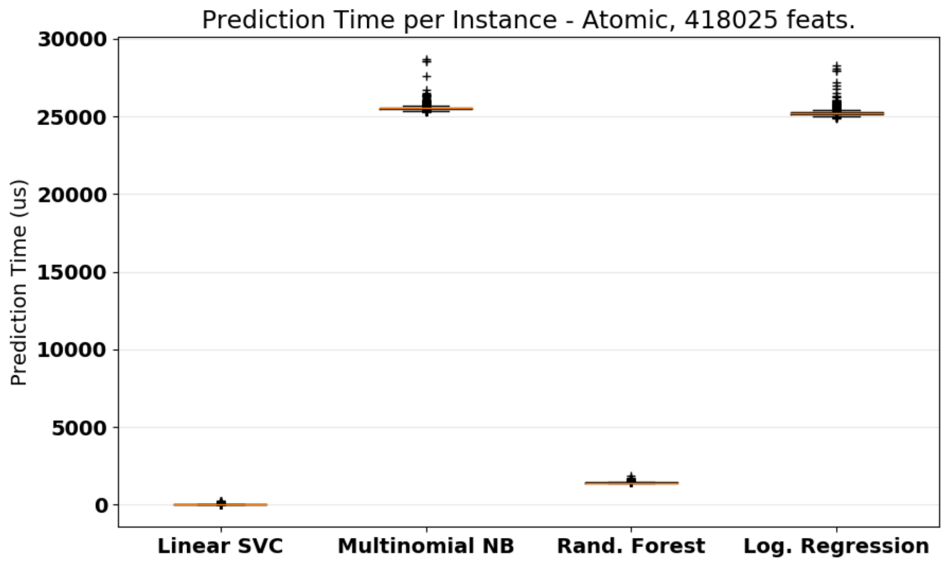

Figure 5.14: The latency for various classification algorithms doing atomic predictions (i.e. one by one). The 20newsgroups dataset with trigram vectorization was used. For feature selection, dynamic n-gram based feature selection with RFC for term weighting was used.

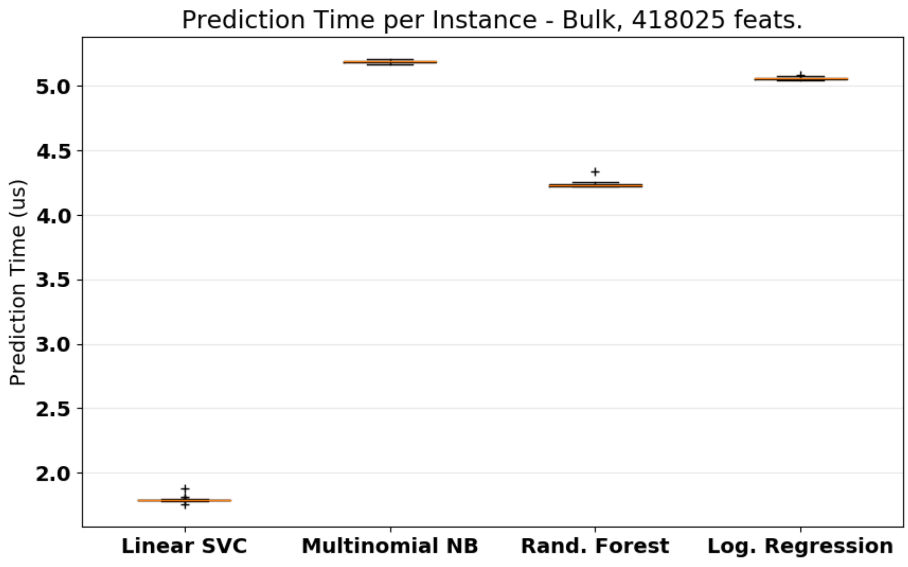

Figure 5.15: The latency for various classification algorithms doing predictions in bulk. The 20newsgroups dataset with trigram vectorization was used. For feature selection, dynamic n-gram based feature selection with RFC for term weighting was used.

6 Discussion

In general, using dynamic n-gram based feature selection to build a machine learning model will most likely not improve the training time necessary to build the model, since the feature selection process is rather expensive, denpending on the chosen term weighting method. We were, however, able to reduce prediction time while providing equal or better classification performance.

We observed in our experiments, that feature selection works especially well when using a very low number of features, thus resulting in a very small feature space. Especially when using skip-grams, we were able to perform better then other feature selection algorithms (figure 5.10).

We were able to improve classification results as well as reduce prediction time. Especially random forest classification for term weigthing seems to be well suited for this task. Using this input setting, we were able to compete with other state of the art feature selection methods in our experiments.

In addition, the results for term weigthing using support vector machines look very promising. Using the altered selection criterion, introduced in section 3.2.1, we were able to achieve good classification performance as well as providing a considerably smaller feature space. Especially when using L1-regularized support vector classification, resulting in sparse feature vectors, this approach works very well.

We achieved a significant improvement using the preprocessed, $tf - idf$ vectorized 20 newsgroups dataset. The fact that numbers are removed, seem to have a high impact on the classification result. By not applying this step, the classification algorithm will learn from noisy training data.

In general, by applying feature selection we were able to improve accuracy for the dataset from Cardoso-Cachopo, 2007 from 0.8284 to 0.8562 for SVC with a linear kernel. While RFC term weigthing works best for all other datasets, χ^2 and NB achieves better classification results for this dataset.

In contrast to other feature selection algorithms, it is not possible to select an exact number of features, since features are chosen dynamically in each iteration. It is, however, possible to approximate the size of the feature space by chosing the percentage of selected features for each iteration. The size of the resulting feature space also depends on the term weighting criterion. For instance, selecting features with high term frequency in one iteration will most certainly select more features, which are considered potentially important, in the next iteration than features with low term frequency, thus resulting in a larger feature space overall.

7 Future Work

As already mentioned in section 2.8, we only use linear kernel functions for our SVM. The is also be possible to experiment with nonlinear kernel functions for feature ranking. For instance, Mangasarian and Kou, 2007 use nonlinear SVMs for feature selection. Liu et al., 2011 extends the linear SVM Recursive Feature Elimination (RFE) algorithm I. Guyon, Weston, et al., 2002 to a nonlinear kernel function. The same process can be applied to our algorithm.

In general, it is possible to use the feature selection algorithm with any external classification algorithm or any other metric, provided that the algorithm assigns weights to features.

As already mentioned in section 3.3.3, the proposed feature selection method is not limited to text classification tasks. Another interesting addition to our work would be the selection process using data with similar characteristics.

8 Conclusion

We have presented a new feature selection approach using an iterative forward selection method based on word n-gram models. We focused on the question if the dynamic n-gram based feature selection method is able to compete with other state of the art feature selection methods, while providing a smaller feature space. In our experiments, we were able to compete with other feature selection methods, such as recursive feature elimination or χ^2 feature selection. The results also indicate that the proposed algorithm works especially well when selecting a very small feature space, often outperforming other feature selection methods.

In addition, our aim was to provide a smaller, less complex feature space in order to reduce training and prediction time as well as increase classification performance. By reducing the feature space we were not able to reduce training time to build a machine learning model. We considerably reduced predicition time while providing equal or better classification performance.

Another main focus of this thesis was the evaluation of different weighting methods. The results from our experiments indicate that random forest classification works very well as term weigthing method. Other term weighting methods look also very promising. For instance, support vector machines achieved good classification performance, especially when using the altered selection criterion, as described in section 3.2.1. We achieved the best results with the preprocessed dataset from Cardoso-Cachopo, 2007 using χ^2 and NB term weighting. In their study they achieved an accuracy of 0.8284. By applying our feature selection process we were able to achieve an accuracy of 0.8562 for a SVM with linear kernel.

Appendix

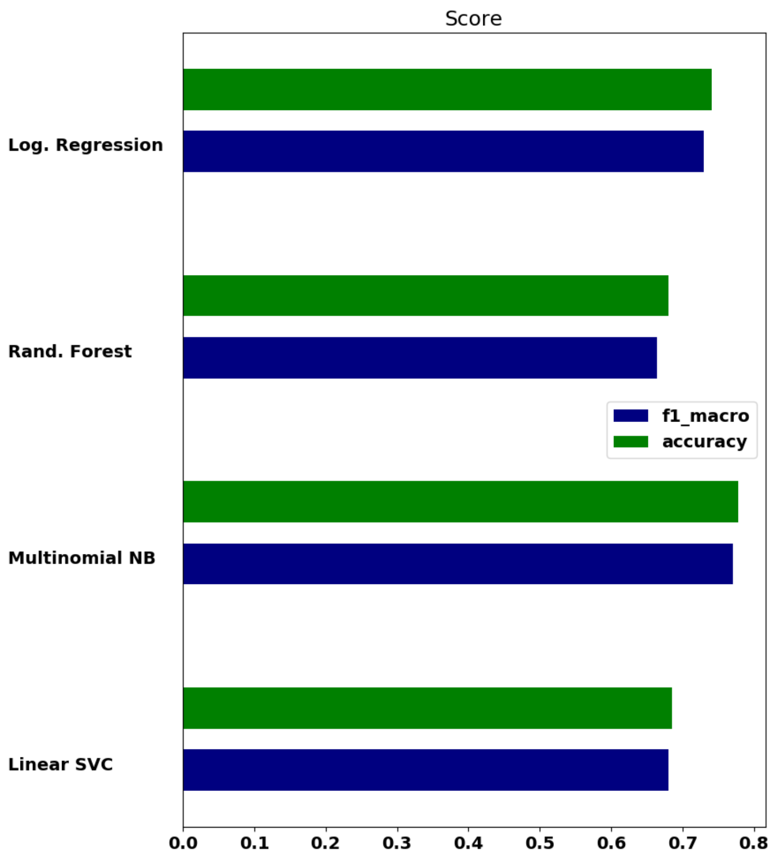

Figure .1: Classification performance for different classification algorithms using the 20 newsgroups dataset with trigrams.

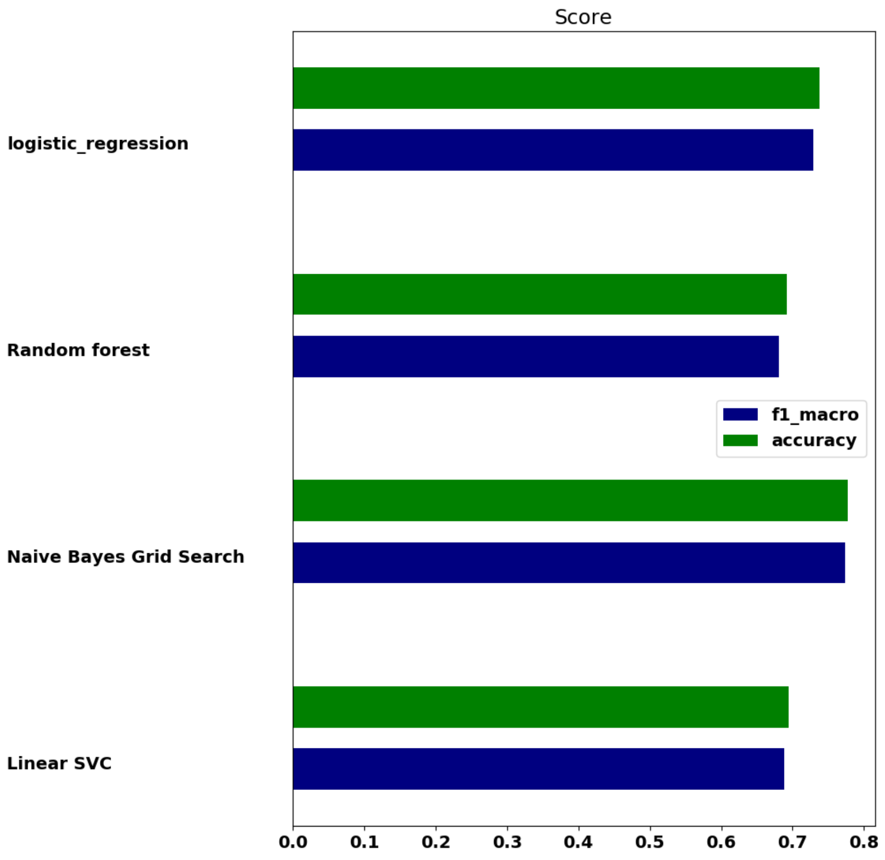

Figure .2: Classification performance for different classification algorithms using the 20 newsgroups dataset with trigrams. Dynamic n-gram based feature selection with RFC for term weigthing was applied.

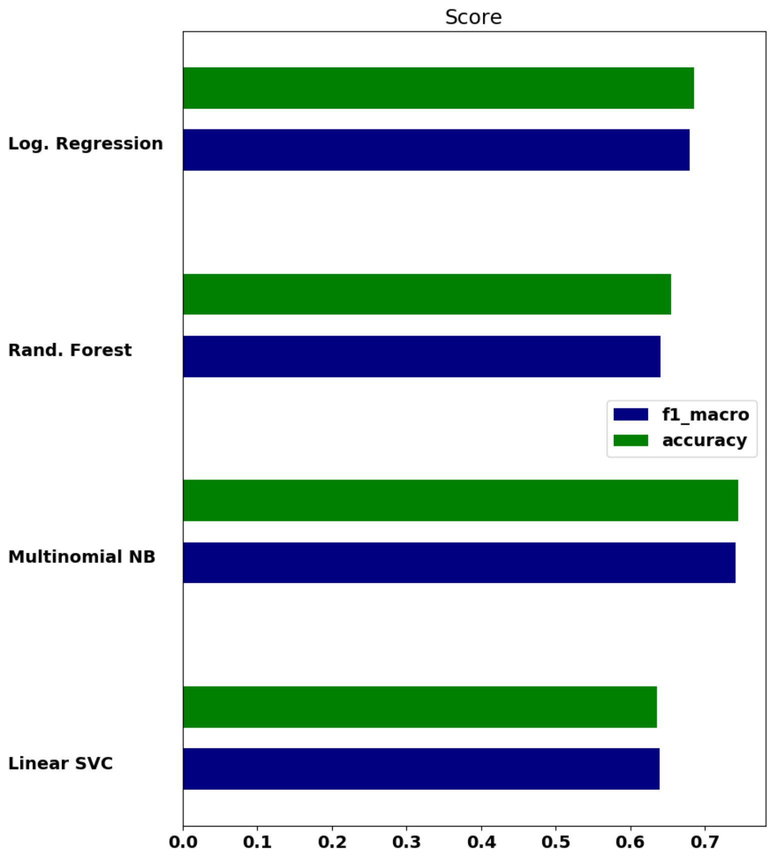

Figure .3: Classification performance for different classification algorithms using the 20 newsgroups dataset with trigrams. Dynamic n-gram based feature selection with SVC for term weigthing was applied.

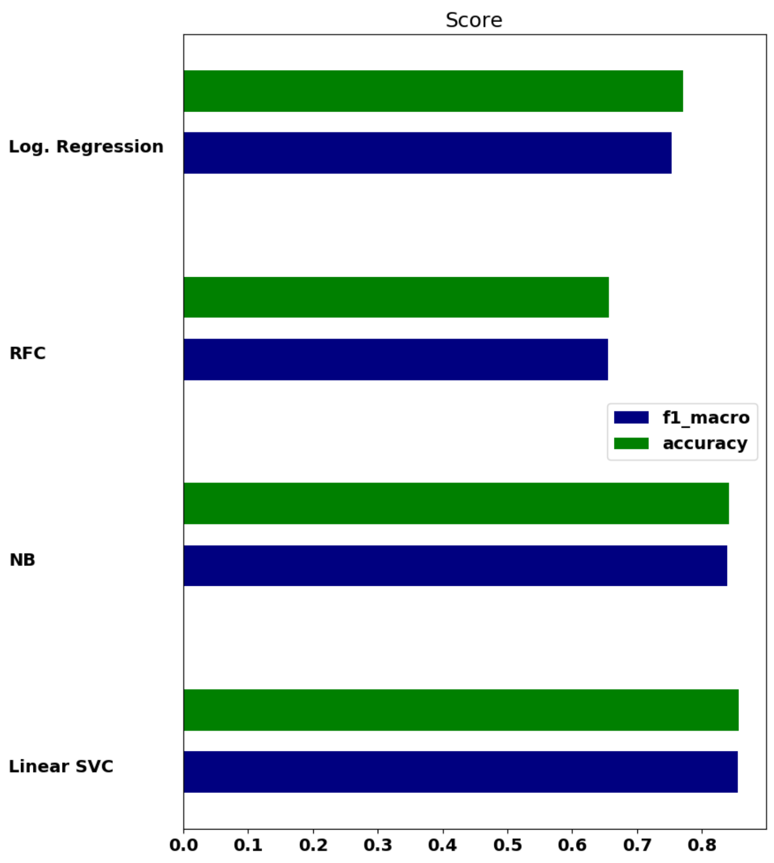

Figure .4: Classification performance for different classification algorithms using the 20 newsgroups dataset with trigrams. Dynamic n-gram based feature selection using the chi^2 measure for term weigthing was applied.

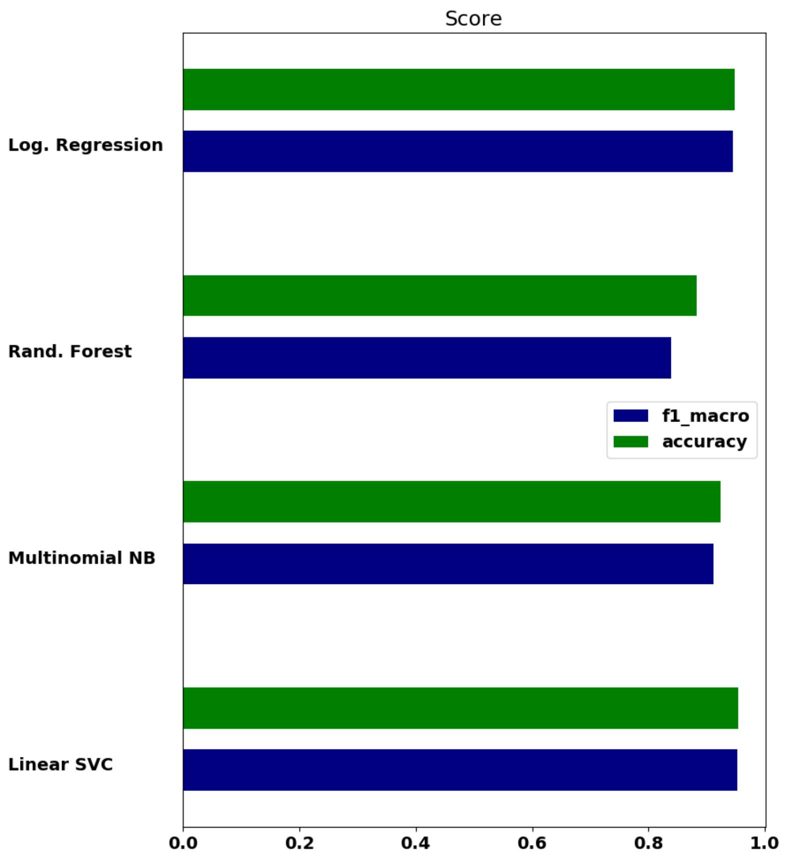

Figure .5: Classification performance for different classification algorithms using the reuters dataset with trigrams.

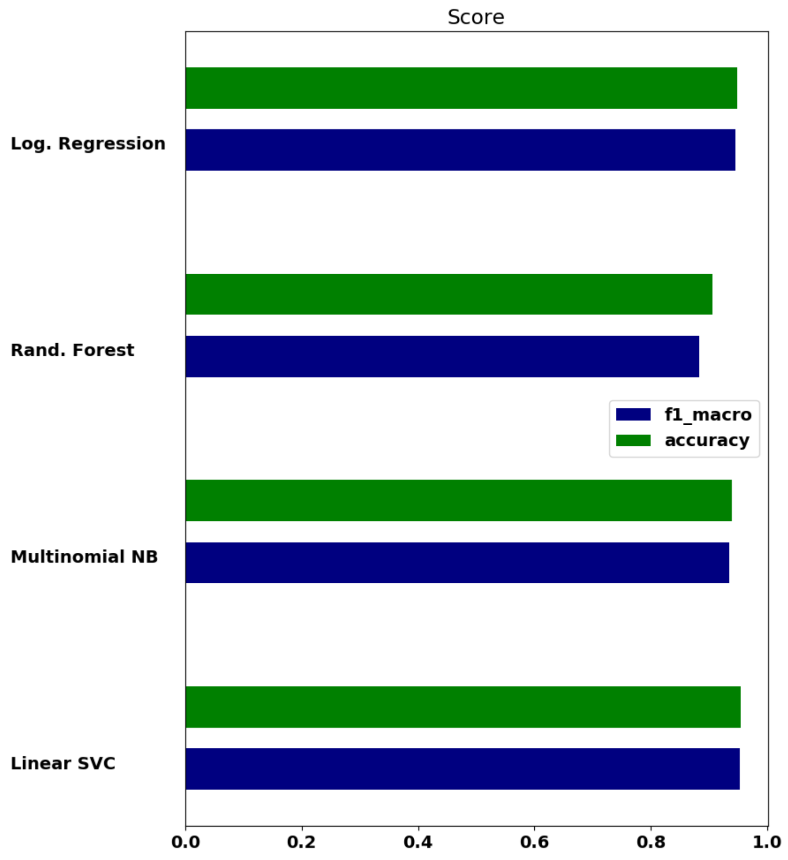

Figure .6: Classification performance for different classification algorithms using the reuteres dataset with trigrams. Dynamic n-gram based feature selection with RFC for term weigthing was applied.

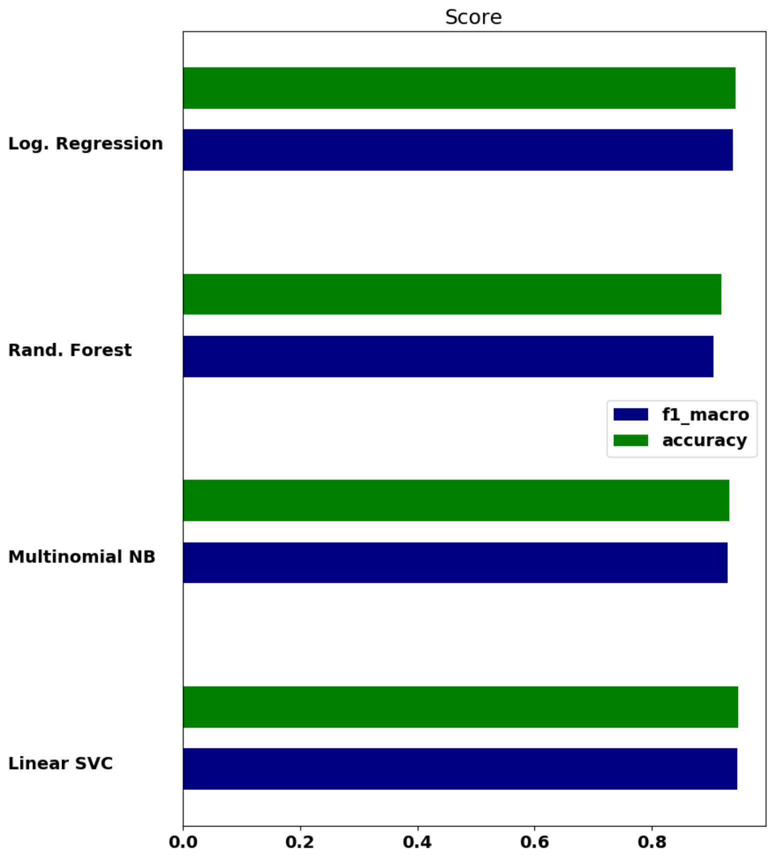

Figure .7: Classification performance for different classification algorithms using the reuters dataset with trigrams. Dynamic n-gram based feature selection with SVC for term weigthing was applied.

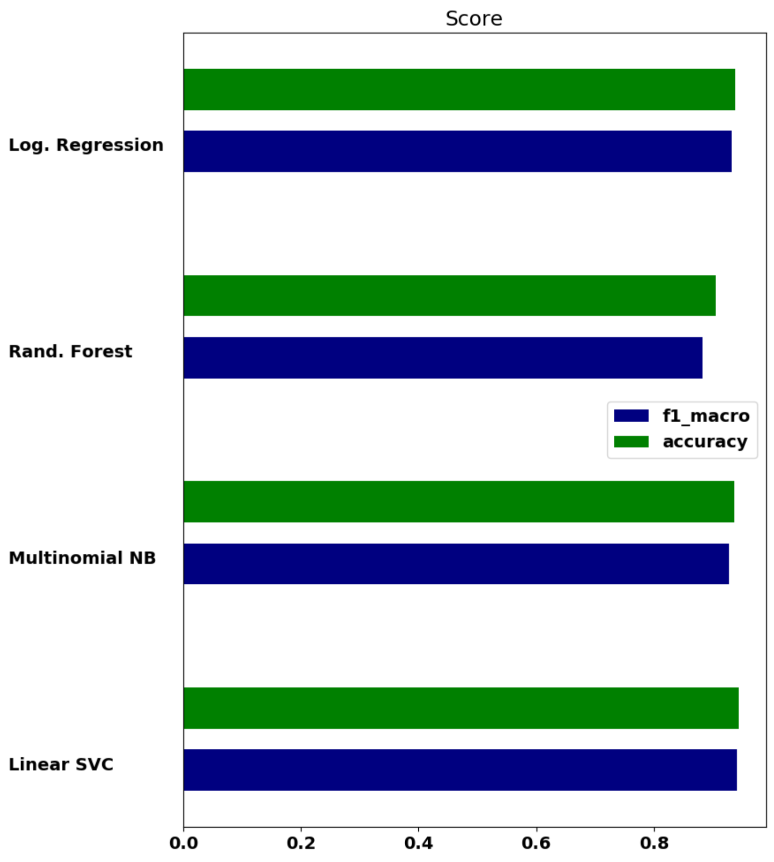

Figure .8: Classification performance for different classification algorithms using the reuters dataset with 2-skip-3-grams.

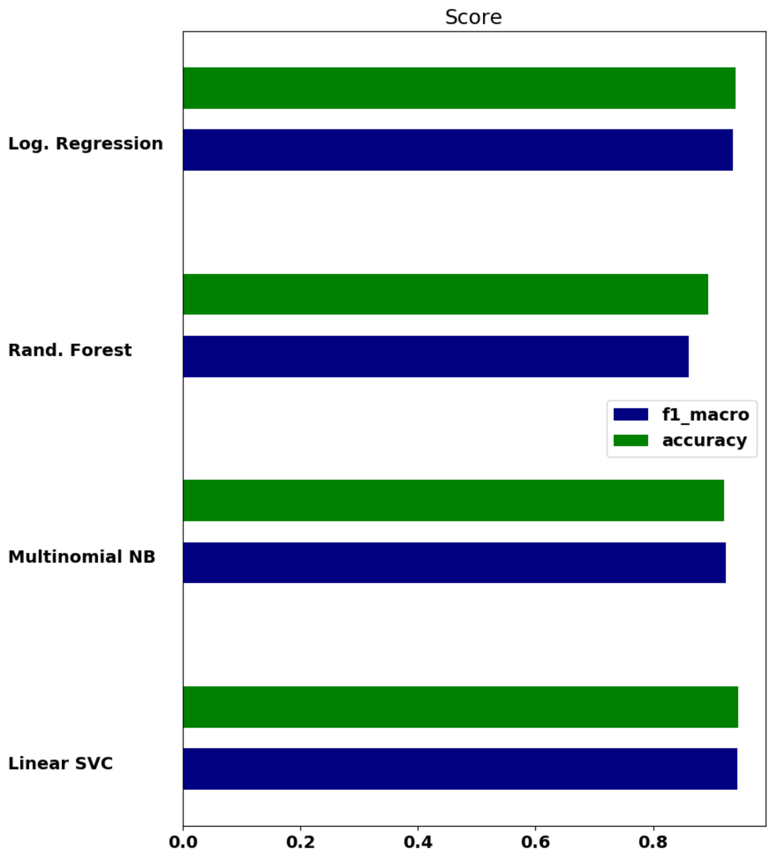

Figure .9: Classification performance for different classification algorithms using the reuters dataset with 2-skip-3-grams. Dynamic n-gram based feature selection with the altered selection criterion and SVC for term weigthing was applied.

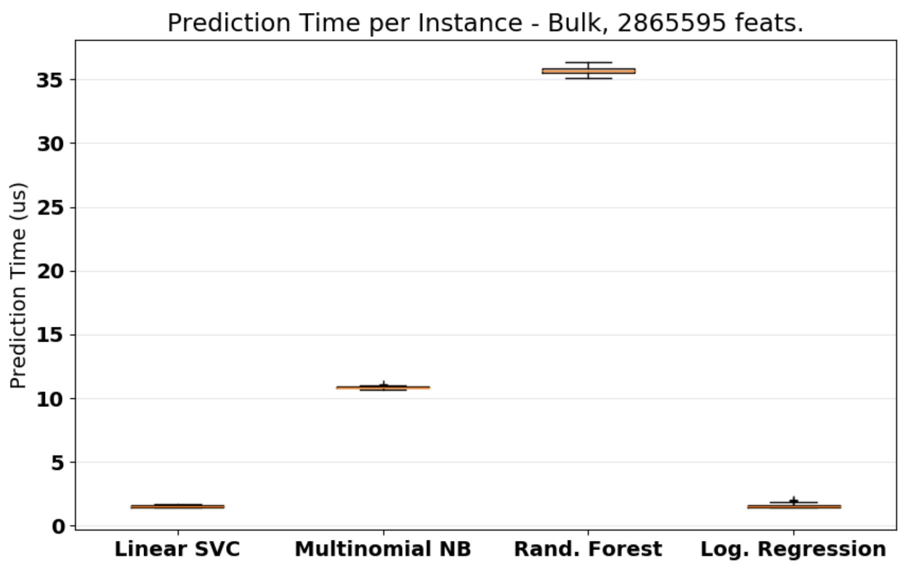

Figure .10: The latency for various classification algorithms doing predictions in bulk. The reuters dataset with 2-skip-3-grams vectorization was used.

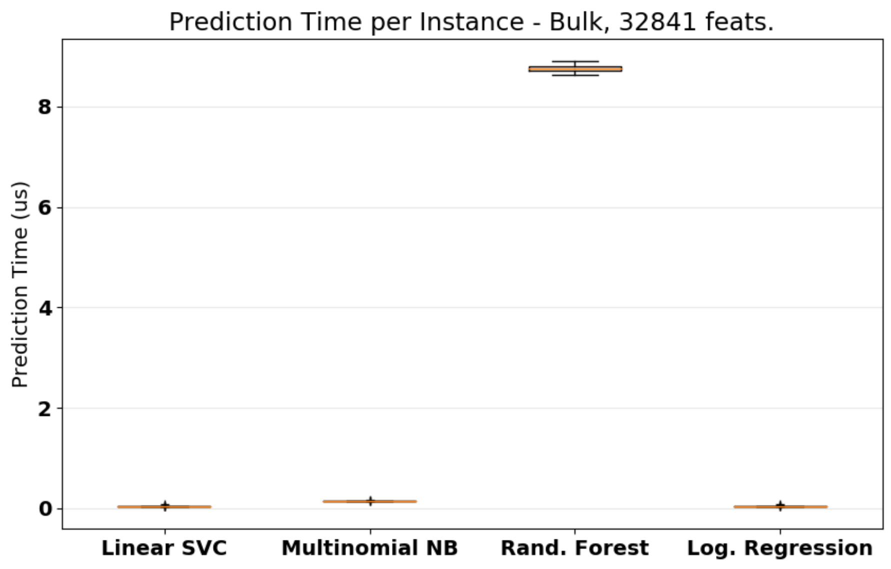

Figure .11: The latency for various classification algorithms doing predictions in bulk. The reuters dataset with 2-skip-3-grams vectorization was used. For feature selection, dynamic n-gram based feature selection with the altered selection criterion and SVC for term weigthing was applied.

Bibliography

Agrawal, Rakesh, Ramakrishnan Srikant, et al. (1994). “Fast algorithms for mining association rules.” In: *Proc. 20th int. conf. very large data bases, VLDB*. Vol. 1215, pp. 487–499 (cit. on p. 2).

Allison, Ben, David Guthrie, and Louise Guthrie (2006). “Another look at the data sparsity problem.” In: *Text, Speech and Dialogue*. Springer, pp. 327–334 (cit. on p. 7).

Boser, Bernhard E, Isabelle M Guyon, and N. Vladimir Vapnik (1992). “A training algorithm for optimal margin classifiers.” In: *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, pp. 144–152 (cit. on p. 22).

Bottou, Léon and Chih-Jen Lin (2007). “Support vector machine solvers.” In: *Large scale kernel machines* 3.1, pp. 301–320 (cit. on p. 58).

Breiman, L et al. (2005). “Classification and regression trees, Wadsworth international group, Belmont, California, USA, 1984; BP Roe et al., Boosted decision trees as an alternative to artificial neural networks for particle identification.” In: *Nucl. Instrum. Meth. A* 543, p. 577 (cit. on p. 21).

Buitinck, Lars et al. (2013). “API design for machine learning software: experiences from the scikit-learn project.” In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122 (cit. on pp. 16, 17, 20, 22, 33, 34).

Cardoso-Cachopo, Ana (2007). *Improving Methods for Single-label Text Categorization*. PdD Thesis, Instituto Superior Tecnico, Universidade Tecnica de Lisboa (cit. on pp. 32, 48, 63, 67).

Cavnar, William B, John M Trenkle, et al. (1994). “N-gram-based text categorization.” In: *Ann Arbor MI* 48113.2, pp. 161–175 (cit. on p. 5).

Chen, Stanley F and Joshua Goodman (1996). "An empirical study of smoothing techniques for language modeling." In: *Proceedings of the 34th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pp. 310–318 (cit. on p. 16).

Domingos, Pedro and Michael Pazzani (1997). "On the optimality of the simple Bayesian classifier under zero-one loss." In: *Machine learning* 29.2-3, pp. 103–130 (cit. on p. 15).

Domingos, Pedro and Michael J Pazzani (1996). "Beyond independence: Conditions for the optimality of the simple Bayes classifier." In: *Proceedings of the 13th International Conference on Machine Learning*, pp. 105–112 (cit. on p. 15).

Eyheramendy, Susana, David D Lewis, and David Madigan (2003). "On the naive bayes model for text categorization." In: (cit. on p. 16).

Fan, Rong-En et al. (2008). "LIBLINEAR: A Library for Large Linear Classification." In: *Journal of Machine Learning Research* 9, pp. 1871–1874 (cit. on pp. 23, 40).

Fayyad, Usama Mohammad (1992). "On the induction of decision trees for multiple concept learning." In: (cit. on p. 19).

Fayyad, Usama Mohammad and B. Keki Irani (1993). "Multi-interval discretization of continuous-valued attributes for classification learning." In: Amherst, MA: Morgan Kaufmann, pp. 112–119 (cit. on p. 19).

Forman, George (2003). "An extensive empirical study of feature selection metrics for text classification." In: *The Journal of machine learning research* 3, pp. 1289–1305 (cit. on pp. 1, 10, 12).

Franz, Alex and Thorsten Brants (2006). *All our n-gram are belong to you*. https://research.googleblog.com/2006/08/all-our-n-gram-are-belong-to-you.html. Accessed: 2018-02-18. Linguistic Data Consortium, 2006 (cit. on p. 1).

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani (2001). *The elements of statistical learning*. Vol. 1. Springer series in statistics New York (cit. on p. 10).

Fürnkranz, Johannes (1998). "A study using n-gram features for text categorization." In: *Austrian Research Institute for Artifical Intelligence* 3.1998, pp. 1–10 (cit. on p. 5).

Gale, William A. and Kenneth W. Church (1994). "What's wrong with adding one." In: *Corpus-Based Research into Language. Rodolpi* (cit. on p. 16).

Geman, Stuart, Elie Bienenstock, and René Doursat (1992). "Neural networks and the bias/variance dilemma." In: *Neural computation* 4.1, pp. 1–58 (cit. on pp. 7, 14).

Gini, Corrado (1912). "Variabilità e mutabilità." In: *Reprinted in Memorie di metodologica statistica (Ed. Pizetti E, Salvemini, T). Rome: Libreria Eredi Virgilio Veschi* (cit. on p. 20).

Good, Irving J (1953). "The population frequencies of species and the estimation of population parameters." In: *Biometrika* 40.3-4, pp. 237–264 (cit. on p. 16).

Goodman, Joshua T (2001). "A bit of progress in language modeling." In: *Computer Speech & Language* 15.4, pp. 403–434 (cit. on p. 16).

Granitto, Pablo et al. (2006). "Recursive Feature Elimination with Random Forest for PTR-MS analysis of agroindustrial products." In: 83 (cit. on p. 12).

Guthrie, David et al. (2006). "A closer look at skip-gram modelling." In: *Proceedings of the 5th international Conference on Language Resources and Evaluation (LREC-2006)*, pp. 1–4 (cit. on pp. 5, 6).

Guyon, Isabelle and André Elisseeff (2003). "An introduction to variable and feature selection." In: *Journal of machine learning research* 3.Mar, pp. 1157–1182 (cit. on pp. 2, 9).

Guyon, Isabelle, Jason Weston, et al. (2002). "Gene selection for cancer classification using support vector machines." In: *Machine learning* 46.1, pp. 389–422 (cit. on pp. 12, 65).

Hsu, Chih-Wei, Chih-Chung Chang, Chih-Jen Lin, et al. (2003). "A practical guide to support vector classification." In: (cit. on pp. 23, 34).

Hunter, J. D. (2007). "Matplotlib: A 2D graphics environment." In: *Computing In Science & Engineering* 9.3, pp. 90–95. DOI: 10.1109/MCSE.2007.55 (cit. on p. 28).

Ikonomakis, M, S Kotsiantis, and V Tampakas (2005). "Text classification using machine learning techniques." In: *WSEAS Transactions on Computers* 4.8, pp. 966–974 (cit. on p. 13).

Jeffreys, Harold (1948). *Theory of Probability*. second edition. Oxford: Clarendon Press (cit. on p. 16).

Jiang, Jing and Chengxiang Zhai (2007). "An empirical study of tokenization strategies for biomedical information retrieval." In: *Information Retrieval* 10.4-5, pp. 341–363 (cit. on p. 4).

Joachims, Thorsten (1998). *Text categorization with support vector machines: Learning with many relevant features*. Springer (cit. on pp. 9, 23).

Johnson, W. Ernest (1932). "Probability: deductive and inductive problems." In: *Mind* 41.164, pp. 421–423 (cit. on p. 16).

Jones, Eric, Travis Oliphant, Pearu Peterson, et al. (2001–). *SciPy: Open source scientific tools for Python*. `http://www.scipy.org`. Accessed: 2018-02-18 (cit. on p. 29).

Kohavi, Ron and George H John (1997). "Wrappers for feature subset selection." In: *Artificial intelligence* 97.1-2, pp. 273–324 (cit. on pp. 11, 12).

Koller, Daphne and Mehran Sahami (1997). "Hierarchically classifying documents using very few words." In: (cit. on pp. 9, 14).

Lang, Ken (1995). "Newsweeder: Learning to filter netnews." In: *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 331–339 (cit. on p. 31).

Lewis, David D (1998). "Naive (Bayes) at forty: The independence assumption in information retrieval." In: *Machine learning: ECML-98*. Springer, pp. 4–15 (cit. on p. 15).

Lewis, David D and Marc Ringuette (1994). "A comparison of two learning algorithms for text categorization." In: *Third annual symposium on document analysis and information retrieval*. Vol. 33, pp. 81–93 (cit. on p. 14).

Lewis, David Dolan (1992). "Representation and learning in information retrieval." PhD thesis. University of Massachusetts (cit. on p. 7).

Lichman, M. (2013). *UCI Machine Learning Repository*. `http://archive.ics.uci.edu/ml`. Accessed: 2018-02-18 (cit. on p. 32).

Lidstone, G. James (1920). "Note on the general case of the Bayes-Laplace formula for inductive or a posteriori probabilities." In: *Transactions of the Faculty of Actuaries* 8.182-192, p. 13 (cit. on p. 16).

Liu, Quanzhong et al. (2011). "Feature selection for support vector machines with RBF kernel." In: 36, pp. 99–115 (cit. on p. 65).

Loper, Edward and Steven Bird (2002). "NLTK: The Natural Language Toolkit." In: *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics* (cit. on p. 28).

Louppe, Gilles (2014). "Understanding random forests: From theory to practice." In: *arXiv preprint arXiv:1407.7502* (cit. on p. 53).

Mangasarian, Olvi L and Gang Kou (2007). "Feature selection for nonlinear kernel support vector machines." In: *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*. IEEE, pp. 231–236 (cit. on p. 65).

Manning, Christopher D and Hinrich Schütze (1999). *Foundations of statistical natural language processing*. MIT press (cit. on p. 4).

Manning, Christopher D, Prabhakar Raghavan, Hinrich Schütze, et al. (2008). *Introduction to information retrieval*. Vol. 1. Cambridge university press Cambridge (cit. on pp. 7, 8, 13–17).

McCallum, Andrew, Kamal Nigam, et al. (1998). "A comparison of event models for naive bayes text classification." In: *AAAI-98 workshop on learning for text categorization*. Vol. 752. Citeseer, pp. 41–48 (cit. on pp. 15–17).

Mitchell, Tom M. (1997). *Machine Learning*. English. Internat., 24. [print.] New York, NY [u.a.]: McGraw-Hill Science/Engineering/Math. ISBN: 0071154671; 9780070428072; 9780071154673; 0070428077 (cit. on pp. 15, 17–19).

Murphy, Patrick M. and Michael J. Pazzani (1994). "Exploring the decision forest: An empirical investigation of Occam's razor in decision tree induction." In: *Journal of Artificial Intelligence Research*, pp. 257–275 (cit. on p. 19).

Olshen, LBJFR, Charles J Stone, et al. (1984). "Classification and regression trees." In: *Wadsworth International Group* 93.99, p. 101 (cit. on p. 20).

Pandya, Rutvija and Jayati Pandya (2015). "C5. 0 algorithm to improved decision tree with feature selection and reduced error pruning." In: *International Journal of Computer Applications* 117.16 (cit. on p. 20).

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12, pp. 2825–2830 (cit. on p. 28).

Quinlan, J. Ross (1986). "Induction of decision trees." In: *Machine learning* 1.1, pp. 81–106 (cit. on p. 18).

Quinlan, J Ross (1986). "The effect of noise on concept learning." In: *Machine learning: An artificial intelligence approach* 2, pp. 149–166 (cit. on pp. 6, 7).

Quinlan, J. Ross (1987). "Simplifying decision trees." In: *International journal of man-machine studies* 27.3, pp. 221–234 (cit. on p. 19).

Quinlan, J Ross (2014). *C4.5: programs for machine learning*. Elsevier (cit. on p. 19).

Robertson, Stephen E and K Sparck Jones (1976). "Relevance weighting of search terms." In: *Journal of the American Society for Information science* 27.3, pp. 129–146 (cit. on p. 8).

Song, Fei and W Bruce Croft (1999). "A general language model for information retrieval." In: *Proceedings of the eighth international conference on Information and knowledge management*. ACM, pp. 316–321 (cit. on p. 16).

Sparck Jones, Karen (1972). "A statistical interpretation of term specificity and its application in retrieval." In: *Journal of documentation* 28.1, pp. 11–21 (cit. on p. 8).

Stéfan van der Walt, S. Chris Colbert and Gaël Varoquaux (2011). "The NumPy array: a structure for efficient numerical computation." In: *Computing in Science & Engineering* 13.2, pp. 22–30 (cit. on pp. 28, 31).

Utgoff, E. Paul (1991). "Linear machine decision trees." In: COINS Technical Report 91-10 (cit. on p. 19).

Van Rijsbergen, Cornelis J (1979). *Information Retrieval*. London: Butterworths (cit. on p. 24).

Vapnik, N. Vladimir (1995). *The Nature of Statistical Learning Theory*. Springer (cit. on p. 22).

Yang, Yiming and Jan O Pedersen (1997). "A comparative study on feature selection in text categorization." In: *ICML*. Vol. 97, pp. 412–420 (cit. on pp. 2, 9, 10).