

# Fast Logistic Regression for Text Categorization with Variable-Length N-grams

Georgiana Ifrim  
Max-Planck Institute for  
Informatics  
Stuhlsatzenhausweg 85  
66123 Saarbrücken, Germany  
ifrim@mpi-inf.mpg.de

Gökhan Bakır  
Google Switzerland GmbH  
Freigutstrasse 12  
Zürich, Switzerland  
ghb@google.com

Gerhard Weikum  
Max-Planck Institute for  
Informatics  
Stuhlsatzenhausweg 85  
66123 Saarbrücken, Germany  
weikum@mpi-inf.mpg.de

## ABSTRACT

A common representation used in text categorization is the bag of words model (aka. unigram model). Learning with this particular representation involves typically some pre-processing, e.g. stopwords-removal, stemming. This results in *one* explicit tokenization of the corpus. In this work, we introduce a logistic regression approach where learning involves automatic tokenization. This allows us to weaken the a-priori required knowledge about the corpus and results in a tokenization with variable-length (word or character) n-grams as basic tokens. We accomplish this by solving logistic regression using gradient ascent in the space of all n-grams. We show that this can be done very efficiently using a branch and bound approach which chooses the maximum gradient ascent direction projected onto a single dimension (i.e., candidate feature). Although the space is very large, our method allows us to investigate variable-length n-gram learning. We demonstrate the efficiency of our approach compared to state-of-the-art classifiers used for text categorization such as cyclic coordinate descent logistic regression and support vector machines.

## Categories and Subject Descriptors

H.3 [Information Storage and Retrieval]: Content Analysis and Indexing; I.2.6 [Artificial Intelligence]: Learning

## General Terms

Algorithms, Performance, Experimentation

## Keywords

Fast Logistic Regression, N-gram Learning, Text Categorization, Variable-Length N-grams

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KDD'08, August 24–27, 2008, Las Vegas, Nevada, USA.  
Copyright 2008 ACM 978-1-60558-193-4/08/08 ...\$5.00.

## 1. INTRODUCTION

The standard bag of words representation is widely used in text categorization as an explicit tokenization of the training text, before employing learning algorithms. Typically, some language dependent pre-processing is employed, such as stopwords removal or stemming. Furthermore, a feature selection step [32] is often crucial for computational efficiency and generalization. Such feature-engineering often requires detailed knowledge about the language of the text to be categorized. In practice, this results in a lot of tuning of the classifiers in order to find the right unigram features.

However, there are important text classification tasks for which the initial unigram bag-of-words representation does not capture the rich facets of the problem, even if the learner itself is very powerful [13, 18, 23, 27, 34].

Examples are: email categorization, sentiment polarity mining in product or movie reviews, subjectivity versus objectivity mining of given texts, authorship attribution, user classification in social networks, and others. For instance, opinion mining often needs to consider entire phrases such as "... [This president did] *not meet our expectations* ...", and classification in social communities may want to consider titles of music songs that a person likes, which are usually phrases.

For the above applications, more complex features are needed, like word n-grams or even natural-language parse trees resulting in an even more involved tuning procedure. Kudo et al. [22] observed that the performance with n-grams did not differ much from the quality achievable by using deep NLP (natural-language processing) techniques, which would be orders of magnitude more expensive anyway.

In this paper, we focus on n-gram sequences as features. We consider both word-level n-grams to capture phrases, and character-level n-grams to capture morphological variation (stemming, transcription from non-Latin alphabets, misspelling, etc.).

Introducing n-grams as features of a classification model confronts the learner with a combinatorial-explosion problem and a quality-efficiency trade-off. Simply including all n-grams up to some maximum length, say 3 or 4, leads to extremely high-dimensional feature spaces. Although many learners can cope reasonably well with large but sparse input spaces (e.g., [9, 17, 35]), their learning cost is at least linear in the number of features that are present in the training data. Here, high-accuracy classification implies high training cost; conversely, a conservatively bounded set of

n-grams like 2-grams often leads to merely mediocre classification quality. An alternative approach is to pre-process text corpora to identify interesting n-grams by various forms of co-occurrence statistics or frequent-itemset mining [3]. However, this kind of feature engineering also entails high training-time costs, which would prevent it from being used in environments that require frequent re-training (e.g., in spam mail detection). This may be mitigated, to some extent, by active learning (e.g., [20]), but this in turn puts the burden on the users by requiring a potentially large amount of human attention.

The bottom line of these considerations is that all prior methods are strongly limited in their ability to reconcile expressive n-gram-aware feature spaces with fast training procedures. This paper presents a new learning algorithm that is able to work with the entire space of (unbounded) n-grams as features, but automatically selects a compact set of most valuable n-grams for its final model.

## 1.1 Contribution

Our solution, coined *SLR* (for *Structured Logistic Regression*), incorporates the best n-gram features, for variable-length  $n$ , into the feature space while staying highly efficient in its training procedure. To this end, we develop a coordinate-wise gradient ascent technique for maximizing the logistic regression likelihood of the training data. Our method exploits the inherent structure of the n-gram feature space in order to automatically provide a compact set of highly discriminative n-gram features. Instead of computing the gradient value at each coordinate (dimension) corresponding to a possible n-gram feature, we search for the n-gram feature which gives the highest value of the gradient in a given iteration. The vector found this way is non-orthogonal to the full gradient vector, thus guaranteeing that it is a good direction to follow in order to maximize the objective function.

To determine the feature with the best gradient value as fast as possible, we derive a theoretical bound which quantifies the “goodness” of the gradient for each n-gram candidate given its length- $(n-1)$  prefix. This way we can timely decide whether it is worthwhile advancing the search in a particular part of the search space. The effect is that we can prune large parts of the search space, resulting in a practically viable method even for large  $n$ . The result of our learning algorithm is a sparse linear model learned in the space of all possible n-grams in the training data.

We present experiments that compare our *SLR* method against the state-of-the-art classifiers *BBR* (a logistic regression method) [9] and *SVM<sup>perf</sup>* [17]. These opponents are widely viewed as the best known methods for text classification, with fast training procedures. We study a variety of configurations for three different real-life datasets: the opponents can employ n-grams, with different choices of maximum  $n$ , and are tuned for each setting. The F1 measure for our method is comparable to that of the best opponent. In terms of training run-time, *SLR* is more than one order of magnitude faster than its opponents.

To the best of our knowledge, *SLR* is the first method that can incorporate variable-length n-grams into the learning of advanced text classifiers, without any noticeable penalty on the size of the feature space and computational cost of the training.

## 2. RELATED WORK

Recent advances in efficient, regularized learning algorithms, such as SVM [17] and sparse logistic regression [9] have reduced the need for explicitly modifying the input feature space, by better coping with large feature spaces and still providing very good predictive models. These methods still scale linearly with the feature space size and therefore are usually employed with the unigram bag of words representation, rather than the much richer feature space of all (word or character) n-grams in the training text. As a side effect of this efficiency aspect, most text categorization approaches fix the basic token of the text representation at the word level, rather than at the character level. This has the effect of potentially losing some of the robustness of the learned predictive models, since the character-level tokenization may better capture several facets of language use. For example, learning with variable or unrestricted-length character n-grams could better capture spelling mistakes, spam characteristics (punctuation, etc.) or sub-words (implicit stemming) and phrasal features. Furthermore, the sometimes difficult problem of defining word-like segments in Asian language text could be avoided. Other benefits of using variable-length character n-grams could come from the more robust statistics captured by substrings of the text.

Some existing learning approaches can work with character sequences rather than bag of words, for example Markov chain models [8, 28], which are generative approaches, or SVM with string kernel [29], a discriminative approach. Markov chain models can be in fixed order/memory or variable order/memory [25, 34]. The Markov chain models in fixed order  $n$  are usually called n-gram language models [10, 28]. Recently, [27] tried character-level n-gram modeling for text classification, but in order to achieve decent performance one needs to choose an appropriate order  $n$  and employ good smoothing techniques [27, 34]. Markov chain models in variable order adjust the memory length according to the context, hence they are much more flexible than fixed order Markov chain models. The amnesic probabilistic automata (aka PST - prediction suffix trees) [5], text compression [2] methods such as PPM (prediction by partial matching) and PPM\* [4] belong to the family of variable order Markov models. However, previous work has repeatedly shown that generative approaches are generally outperformed by discriminative approaches (e.g. SVM) for word-based text categorization [6, 16, 31, 34]. For string-based (e.g. character-level n-gram) categorization, the number of distinct substrings in a large corpus becomes prohibitively large, thus preventing the straightforward application of most discriminative approaches. SVM with string kernel is a discriminative approach that can perform string-based text categorization. However, SVM with string kernel has not become as popular as the word-based kernel SVM for text classification tasks, due to efficiency and classification performance reasons [24, 34]. Recent work [34] has advocated the usage of an efficient feature selection step for selecting a subset of character-level n-gram features based on a suffix tree algorithm, followed by learning an SVM classifier. This again disconnects the feature selection step from the actual learning algorithm, which is undesirable (the combined process of feature selection followed by a learning algorithm has no clear statistical foundation [9]) and could be avoided by employing efficient classifiers that can do the feature selection on-the-fly as part of the learning process.

For maximum likelihood logistic regression, the most common optimization approach in statistical software is the multidimensional Newton-Raphson method and its variants [26]. Newton algorithms have the advantage of converging in very few iterations. For high-dimensional problems such as text categorization, however, Newton algorithms have the serious disadvantage of requiring  $O(d^2)$  memory, where  $d$  is the number of model parameters. A variety of alternate optimization approaches have therefore been explored for maximum likelihood logistic regression, and for regularized (Maximum A Posteriori) logistic regression. Some of these algorithms, such as limited memory BFGS [26], conjugate gradient [26], and hybrids of conjugate gradient with other methods [19], compute the gradient of the objective function at each step. This requires only  $O(d)$  memory (in addition to the data itself). Efron et al. [7] describe a new class of “least angle” algorithms for lasso linear regression and related models. Other methods solve a series of partial optimization problems. Some of these methods use the subproblems to maintain an evolving approximation to the gradient of the full objective [26], which still requires  $O(d)$  memory. Others use each subproblem only to make progress on the overall objective, using only constant memory beyond that for the parameter vector. The one dimensional subproblems may be based on processing one parameter at a time, as in iterative scaling [15], and cyclic coordinate descent [30, 35]. Some of these algorithms have already shown promise on text categorization or other language processing tasks. One of the methods we compare with, Bayesian Logistic Regression (BBR) [9], is an efficient implementation of regularized cyclic coordinate descent logistic regression.

### 3. METHOD PROPOSED

#### 3.1 Logistic Regression Model

Let  $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$  denote the training set. Let  $d$  be the number of distinct  $n$ -grams in the feature space. The training samples are represented as binary vectors  $x_i = (x_{i1}, \dots, x_{ij}, \dots, x_{id})^T$ ,  $x_{ij} \in \{0, 1\}$ ,  $i = 1, N$ .  $y_i \in \{0, 1\}$  are class labels encoding membership (1) or non-membership (0) of the training samples in the category. We focus here on binary classification; multi-class classification is treated as several binary classification problems. Let  $X$  be the set of all samples  $x$ . Let  $\beta = (\beta_1, \dots, \beta_j, \dots, \beta_d)$  be a parameter vector. Under the logistic regression model, the probability of a sample belonging to class “one” is ([11]):  $p(x_i; \beta) = \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}}$ . The goal is to learn a mapping  $f : X \rightarrow \{0, 1\}$  from the given training set  $D$  such that given a sample  $x \in X$ , we can predict a class label  $y \in \{0, 1\}$ . Learning such a mapping for logistic regression is equivalent to finding the parameter vector  $\beta$ , that maximizes the log-likelihood of the training set. The log-likelihood of the training set for logistic regression is ([11]):

$$l(\beta) = \sum_{i=1}^N [y_i \cdot \beta^T \cdot x_i - \log(1 + e^{\beta^T \cdot x_i})] \quad (1)$$

We take an optimization approach for solving this problem. Next, we show a procedure for maximizing  $l(\beta)$ , based on a coordinate-wise gradient ascent in the space of all  $n$ -grams in the training set.

#### 3.2 Coordinate-wise gradient ascent in the space of all $n$ -gram sequences

Recall from Section 2 that all prior methods for logistic regression need at least  $O(d)$  memory where  $d$  is the size of the overall feature space. With  $u$  distinct unigrams, there are  $O(u^n)$  potential  $n$ -grams in the training set, thus for large  $u$  and  $n$ , this would incur a very high cost.

In this section we present a new approach that in practice requires  $o(d)$  memory, for solving logistic regression in the large space of all  $n$ -gram sequences in the training text. This becomes possible because we do not need to explicitly store all the distinct features, which would already use  $O(d)$  memory. Our algorithm is based on a branch-and-bound approach which chooses the maximum gradient ascent direction projected onto a single dimension (i.e., candidate feature).

Using equation (1), the gradient of  $l$  with respect to a coordinate value  $\beta_j$  evaluated at a given parameter vector  $\beta$  is:

$$\frac{\partial l}{\partial \beta_j}(\beta) = \sum_{i=1}^N x_{ij} \cdot \left( y_i - \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right) \quad (2)$$

Let  $j$  be a coordinate corresponding to a given  $n$ -gram sequence  $s_j$ , and  $j'$  be a coordinate corresponding to a super sequence of  $s_j$ ,  $s_{j'}$ , i.e.  $s_j$  is a prefix of  $s_{j'}$ . We write  $s_j \in x_i$  to mean  $x_{ij} \neq 0$ .

The following theorem, inspired by work on boosting [22], gives a convenient way of computing an upper bound on the gradient value for any super sequence  $s_{j'} \supseteq s_j$ .

**THEOREM 1.** *For any  $s_{j'} \supseteq s_j$  and  $y \in \{0, 1\}$ , the absolute value of the gradient of  $l(\beta)$  with respect to  $\beta_{j'}$  is bounded by  $\mu(\beta_j)$ , where*

$$\mu(\beta_j) = \max \left\{ \sum_{\{i|y_i=1, s_j \in x_i\}} x_{ij} \cdot \left( 1 - \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right), \sum_{\{i|y_i=0, s_j \in x_i\}} x_{ij} \cdot \left( \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right) \right\}.$$

**PROOF.** *We split the analysis into two subproblems, the first concerning the “positive” class, and the second concerning the “negative” class. First we prove the bound for the positive class:*

$$\frac{\partial l}{\partial \beta_{j'}}(\beta) = \sum_{i=1}^N x_{ij'} \cdot \left( y_i - \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right) \quad (3)$$

$$= \sum_{\{i|s_{j'} \in x_i\}} x_{ij'} \cdot \left( y_i - \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right) \quad (4)$$

$$\leq \sum_{\{i|y_i=1, s_{j'} \in x_i\}} x_{ij'} \cdot \left( 1 - \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right) \quad (5)$$

$$\leq \sum_{\{i|y_i=1, s_j \in x_i\}} x_{ij} \cdot \left( 1 - \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right). \quad (6)$$

*The last inequality holds due to the fact that  $\{i|y_i = 1, s_{j'} \in x_i\} \subseteq \{i|y_i = 1, s_j \in x_i\}$ , for any  $s_{j'} \supseteq s_j$ .*

Similarly, we can show for the negative class that

$$\frac{\partial l}{\partial \beta_{j'}}(\beta) \geq \sum_{\{i|y_i=0, s_j \in x_i\}} x_{ij} \cdot \left( -\frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right). \quad (7)$$

Thus we have:

$$\begin{aligned} \sum_{\{i|y_i=0, s_j \in x_i\}} x_{ij} \cdot \left( -\frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right) &\leq \frac{\partial l}{\partial \beta_{j'}}(\beta) \\ &\leq \sum_{\{i|y_i=1, s_j \in x_i\}} x_{ij} \cdot \left( 1 - \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right) \end{aligned} \quad (8)$$

The absolute value of the gradient of  $l(\beta)$  at coordinate  $j'$  corresponding to  $n$ -gram sequence  $s_{j'}$  is thus bounded by  $\mu(\beta_{j'})$ :

$$\left| \frac{\partial l}{\partial \beta_{j'}}(\beta) \right| \leq \max \left\{ \sum_{\{i|y_i=1, s_j \in x_i\}} x_{ij} \cdot \left( 1 - \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right), \sum_{\{i|y_i=0, s_j \in x_i\}} x_{ij} \cdot \left( \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right) \right\}.$$

□

The theorem essentially states that at a given coordinate, i.e.  $n$ -gram sequence, we can decide whether the gradient of  $l(\beta)$  can be improved by further extending this sequence. This facilitates casting the learning process as a search for the coordinate with best gradient value, rather than computing the full gradient vector, in each optimization iteration. This process searches the entire space of all possible subsequences of the text, and guarantees to find the globally optimal feature (i.e. coordinate) in terms of the gradient value. We sketch the implementation of our algorithm in the next subsection.

Once we find the feature with the best gradient value, we adjust the value of the weight vector:

$$\beta^{new} = \beta^{old} + \epsilon \cdot \frac{\partial l}{\partial \beta_j}(\beta^{old})$$

and repeat the search for the coordinate with maximum (absolute) gradient value. This essentially produces one candidate feature per iteration.  $\epsilon$  is known in the literature as the *step length*, and is usually estimated via line search algorithms [26]. The iterations typically start at  $\beta^{old} = 0$  ([11]). Note that, since each restricted gradient direction is not conjugate to the previous ones, the chosen feature is not necessarily distinct from the already selected features. This is a common property of gradient methods and carries over to our greedy gradient ascent method. The outcome of this iterative process is a very sparse weight vector  $\beta$ , which is a linear model learned in the space of all  $n$ -gram sequences.

### 3.3 Algorithm

A high level overview of our gradient-based search algorithm is shown in Algorithm 1. This algorithm returns the best (in terms of gradient value)  $n$ -gram feature and it is called repeatedly up to the number of desired optimization iterations. The gradient value in iteration  $i$  is always computed at the parameter vector estimated in iteration  $i - 1$ , thus the selection of a new feature depends on the set of previously chosen features. The main parameter of our method

is the number of optimization iterations, which directly influences the size of the final model. We estimate this parameter by thresholding the aggregated change in score predictions (details in Section 4.3). We also implement a line search algorithm for estimating the step length in the direction with best gradient value. We currently use a binary search algorithm for estimating the step length [26]. Although the search space is very large, the pruning bound proposed in this paper effectively prunes the search. We have empirically observed that the pruning condition presented in Theorem 1 prunes more than 90% of the search space.

---

#### Algorithm 1 Find best $n$ -gram feature

---

```

1: Input: Training set  $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ ,
   where  $x_i$  is a training document,  $y_i \in \{0, 1\}$  is a class label
2: Output: Optimal feature (e.g. with best gradient value)
3: begin
4:   global  $\tau$ , best_feature
5:    $\tau = 0$  //suboptimal value of gradient
6:   //for each single unigram
   foreach  $s' \in \bigcup_{i=1}^N \{s | s \in x_i, |s| = 1\}$ 
7:     grow_sequence( $s'$ )
8:   end
9:   return best_feature
10: end

1: function grow_sequence( $s$ )
2:   if  $\mu(s) \leq \tau$  then return // $\mu(s)$  as in Theorem 1
3:   if  $\text{abs}(\text{gradient}(s)) > \tau$  then
4:     best_feature =  $s$  //suboptimal solution
      $\tau = \text{abs}(\text{gradient}(s))$ 
5:   end
6:   foreach  $s'' \in \{s' | s' \supseteq s, s' \in \bigcup_{i=1}^N x_i, |s'| = |s| + 1\}$ 
7:     grow_sequence( $s''$ )
8:   end
9: end

```

---

## 4. EXPERIMENTS

In this section we compare our learning technique to logistic regression and support vector machines.

For logistic regression we use the open source implementation by Genkin et al. [9] which we denote by BBR. This is a recent implementation of regularized logistic regression that was particularly designed to simultaneously select variables and perform learning. BBR is able to handle a large set of features via regularized cyclic coordinate gradient descent. See [9] for details.

For support vector machines we used the latest open source  $SVM^{perf}$  solver by Joachims et al. [17] which is especially tuned for linear problems.

### 4.1 Methodology

To study the effect of using variable-length  $n$ -gram sequences as basic features, we vary the maximum (word or character)  $n$ -gram length and train all methods in the space of all sequences up to a fixed length. For example, if we fix the  $n$ -gram size to  $n = 3$ , this means we train in the space of all  $n$ -grams up to trigrams, e.g. all the unigrams, bigrams and trigrams.

For BBR and SVM, we first use a state-of-the-art pattern mining tool [1, 21, 33] in order to produce all the  $n$ -grams up to a given size (e.g., up to 5-grams), and then learn the two classifiers in this space. Thus the feature space is the same for all methods, but for BBR and SVM we need to generate the space explicitly using a pattern mining tool while *SLR*

searches the entire space automatically and incrementally adds only those features that contribute to a good classifier.

We evaluate all methods with respect to training run-times and micro and macro-averaged F1 measure [31], with word level and character level n-grams. We show that our method can benefit from using arbitrary length n-grams, which is reflected in the F1 measure, and that due to our pruning bound we are much faster than the other methods.

## 4.2 Datasets

We study three different applications<sup>1</sup> of text categorization that could benefit from learning variable-length n-grams. The first application is **movie genre classification**. We take a subset of movies from IMDB, which have a plot section, i.e. one or two paragraphs that describe the movie. IMDB contains information about the genre of each movie in the database. The classification task is to learn the genre of the movie from the short plot description associated with each movie. We take a subset of movies classified to either of the genres Crime or Drama. We select these two genres because they are close in terms of topicality, thus the classification task is harder than when learning to classify movies belonging to orthogonal genres, e.g. Crime versus Comedy. The dataset contains a total of 7,440 documents with 3,720 documents for each genre. There are 63,623 distinct word-unigrams (**IMDB dataset**).

The second application we study is **book reviews classification by genre**. We work with a dataset of editorial reviews of books from Amazon. The collection was first used in [14]. The editorial reviews are grouped into three genres: Biology, Mathematics and Physics. There are 5,634 reviews, with reviews of books about Biology and Mathematics having roughly 2,200 documents each, and those about Physics having about 1,300 documents. There are 55,764 distinct unigrams in this collection (**AMAZON dataset**).

The third application we analyze is **topic detection for Chinese text**. We considered the TREC-5 People’s Daily News dataset investigated in [12]. The dataset contains 6 classes: (1) Politics, Law and Society; (2) Literature and Arts; (3) Education, Science and Culture; (4) Sports; (5) Theory and Academy and (6) Economics, and it is split into training and test. Each class has 500 training documents and 100 test documents. The training set contains 4,961 distinct characters (**CHINESE dataset**).

No pre-processing of the first two datasets was carried out. For the Chinese dataset we removed the SGML tags. For the case of multiple classes, we convert the classification task into several binary classification tasks in the one-versus-all manner.

## 4.3 Parameter settings

We first compare the training running times for all methods, for each dataset and experiment, with a fixed parameter set. We also show various statistics on the models learned by each method. For reporting micro/macro-averaged F1 we also tune the most important (in terms of influence on classification performance) parameter for each method.

### 4.3.1 SLR

**Setting the number of iterations.** For our method the number of iterations directly influences the size of the

final model, since in each iteration we select a candidate feature to be included in the final model. In order to set this parameter, we consider the aggregated change in the score predictions, and if this is not above a fixed threshold, we stop the iterations. The threshold is set to 0.005. This is fixed across all datasets and experiments, for measuring running times. For tuning the number of iterations, we run cross-validation for values between 200 and 1,000 with steps of 100, and choose the best parameter. We set the final classification threshold to the value that minimizes training error. The data representation for our method is the original text, interpreted as a word-level or character-level n-gram sequence.

### 4.3.2 BBR

**Setting the regularization parameter.** For bayesian logistic regression [9] the regularization parameter is the most important (i.e. the prior variance for parameter values). This is initially set to the value recommended in [9], i.e. the ratio of the number of distinct features to the average euclidean norm of documents in the dataset. For tuning this parameter, we use the *autosearch* option, which automatically searches for the best cross-validated hyperparameter value. The other parameters are set as: -p 1 -t 1. The -p 1 parameter selects the Laplace prior distribution on the model parameters. We choose this prior due to the resulting sparse models. The Laplace (aka. lasso) logistic regression is also much faster and more accurate than the Gaussian (aka. ridge) logistic regression. The -t 1 parameter sets the final classification threshold to the value that minimizes the number of training errors. All other parameters are used with their default values. The data representation for BBR is sparse vectors, i.e. id and value for non-zero features.

### 4.3.3 SVM<sup>perf</sup>

**Setting the soft-margin  $C$  parameter.** For SVM<sup>perf</sup> [17] the soft-margin  $C$  parameter is the most important. [17] observed that any parameter value between 100 and 500 would be good across datasets. After some initial trials we set  $C = 100$ , for the fixed parameter experiments. For the tuning setting, we chose the  $C$  value which performed best in cross-validation on the training set, from several values selected from the range 100 to 500. All other parameters are kept with their default values (linear kernel is a default setting). The data representation for SVM is also sparse vectors.

## 4.4 Results

The experiments were run on a Linux machine with 1GByte memory and 2.8GHz Intel CPU. We show micro/macro-averaged F1 [31] as global measure of quality for each dataset and classifier.

### 4.4.1 IMDB dataset

The IMDB dataset is not explicitly split into training and test, thus we run 5-fold cross validation and report the micro/macro-averaged F1. In Table 1 we report the training running time for all methods. We observe that our method is much faster than both BBR and SVM. For word n-grams, our running time stays almost constant with increasing n-gram length (0.35 minutes even with unrestricted length), while BBR goes from 0.3 minutes for unigrams to 4 minutes for (up to) 5-grams. SVM shows a similar trend as BBR, its running time increases from 0.3 minutes for unigrams to 15 minutes for 5-grams. The time reported is an average

<sup>1</sup>All datasets are available at:  
<http://www.mpi-inf.mpg.de/~ifrim/data>

Table 1: IMDB training running times. Micro/Macro-averaged F1 for varying  $n$ .

	word n-grams				char n-grams			
	n=1	n=3	n=5	n unrestricted	n=1	n=3	n=5	n unrestricted
max n-gram length								
# overall features	63,623	838,620	1,922,942	N/A	135	41,433	704,440	N/A
# iterations SVM	500	1,130	1,600	N/A	490	1,721	2,912	N/A
# iterations BBR	250	370	374	N/A	85	265	370	N/A
# iterations SLR	175	180	184	190	325	670	649	620
# features in final model BBR	3,200	3,750	4,000	N/A	97	3,000	4,000	N/A
# features in final model SLR	120	130	135	135	47	440	420	420
Running Time SVM	0.3 min	6 min	15 min	N/A	0.2 min	3.2 min	50 min	N/A
Running Time BBR	0.3 min	2.2 min	4 min	N/A	0.1 min	2.4 min	8.4 min	N/A
Running Time SLR	0.25 min	0.3 min	0.35 min	0.35 min	0.3 min	1.7 min	2.4 min	2.4 min
Time for generating patterns	1 min	2.5 min	3.5 min	N/A	1 min	3 min	5 min	N/A
Total time SVM	1.3 min	6.5 min	18.5 min	N/A	1.2 min	6.2 min	55 min	N/A
Total time BBR	1.3 min	4.7 min	7.5 min	N/A	1.1 min	5.4 min	13.4 min	N/A
<b>Total Time SLR</b>	<b>0.25 min</b>	<b>0.3 min</b>	<b>0.35 min</b>	<b>0.35 min</b>	<b>0.3 min</b>	<b>1.7 min</b>	<b>2.4 min</b>	<b>2.4 min</b>
microavgF1 SVM	69.08%	71.11%	70.71%	N/A	57.13%	65.27%	69.48%	N/A
microavgF1 BBR	67.54%	67.97%	68.21%	N/A	56.32%	65.21%	67.10%	N/A
<b>microavgF1 SLR</b>	<b>72.90%</b>	<b>72.89%</b>	<b>72.64%</b>	<b>73.17%</b>	<b>55.99%</b>	<b>72.86%</b>	<b>73.84%</b>	<b>73.94%</b>
macroavgF1 SVM	69.08%	71.13%	70.74%	N/A	57.16%	65.27%	69.48%	N/A
macroavgF1 BBR	67.56%	68.02%	68.24%	N/A	56.43%	65.21%	67.08%	N/A
<b>macroavgF1 SLR</b>	<b>72.95%</b>	<b>72.94%</b>	<b>72.72%</b>	<b>73.28%</b>	<b>56.09%</b>	<b>72.90%</b>	<b>73.89%</b>	<b>74.00%</b>

Table 2: AMAZON training running times. Micro/Macro-averaged F1 for varying  $n$ .

	word n-grams				char n-grams			
	n=1	n=3	n=5	n unrestricted	n=1	n=3	n=5	n unrestricted
max n-gram length								
# overall features	55,764	1,036,693	2,454,205	N/A	95	50,422	700,993	N/A
# iterations SVM	479	1089	1,722	N/A	592	1,578	2,597	N/A
# iterations BBR	270	390	386	N/A	77	259	320	N/A
# iterations SLR	87	93	95	98	216	347	318	294
# features in final model BBR	1,000	1,250	1,329	N/A	79	1,157	1,332	N/A
# features in final model SLR	75	83	77	86	43	258	259	243
Running Time SVM	0.2 min	6 min	24 min	N/A	0.5 min	2.5 min	55 min	N/A
Running Time BBR	0.4 min	2.7 min	5.4 min	N/A	0.05 min	2.75 min	9.6 min	N/A
Running Time SLR	0.13 min	0.16 min	0.18 min	0.2 min	0.26 min	0.95 min	1.3 min	1.5 min
Time for generating patterns	0.1 min	1 min	4.0 min	N/A	1 min	5 min	8 min	N/A
Total time SVM	0.3 min	7 min	28 min	N/A	1.5 min	7.5 min	63 min	N/A
Total time BBR	0.5 min	3.7 min	9.4 min	N/A	1.05 min	7.75 min	17.6 min	N/A
<b>Total Time SLR</b>	<b>0.13 min</b>	<b>0.16 min</b>	<b>0.18 min</b>	<b>0.2 min</b>	<b>0.26 min</b>	<b>0.95 min</b>	<b>1.3 min</b>	<b>1.5 min</b>
microavgF1 SVM	81.75%	80.42%	78.05%	N/A	39.01%	79.09%	81.88%	N/A
microavgF1 BBR	79.08%	79.23%	80.13%	N/A	42.70%	78.37%	80.45%	N/A
<b>microavgF1 SLR</b>	<b>82.13%</b>	<b>81.80%</b>	<b>82.15%</b>	<b>81.89%</b>	<b>43.73%</b>	<b>82.75%</b>	<b>83.90%</b>	<b>84.49%</b>
macroavgF1 SVM	80.24%	78.40%	75.31%	N/A	30.93%	77.37%	80.52%	N/A
macroavgF1 BBR	77.49%	77.50%	78.58%	N/A	39.82%	76.75%	78.92%	N/A
<b>macroavgF1 SLR</b>	<b>80.64%</b>	<b>80.30%</b>	<b>80.65%</b>	<b>80.40%</b>	<b>40.92%</b>	<b>81.31%</b>	<b>82.52%</b>	<b>83.20%</b>

running time per cross-validation split averaged across topics. The trend for char-level n-grams is similar, our method stays almost constant at 1.7 minutes for 3-grams and 2.4 minutes for unrestricted n-gram length. BBR and SVM go from 2.7 and 2.5 minutes respectively for character 3-grams, to 8.4 and 50 minutes respectively for 5-grams. Thus, SLR is orders of magnitude faster than the state-of-the-art methods SVM and BBR. This difference in running time is due to the way our technique deals with large feature spaces, by its branch-and-bound search strategy. We also notice that SLR selects much fewer features in the final model, as compared to BBR. For word n-grams, out of 190 iterations for unrestricted n-gram size, it selects 135 distinct features in the final model. BBR runs for 374 iterations in the space of 5-grams, and selects 4,000 distinct features in the final model. For char n-grams, our model runs for 620 iterations and selects 420 distinct features, while BBR runs for 370 iterations for 5-grams and selects 4,000 features. In terms of micro/macro-averaged F1 (Table 1, Table 4), we

observe that our method is as good as the state-of-the-art regarding generalization ability, while being orders of magnitude faster. For the case of fixed parameters, our method is 3 to 5% better than both BBR and SVM. In the case of tuned parameters, for word n-grams our method achieves 74.04% macro-averaged F1, while BBR achieves 73.94% and SVM 71.24%. Regarding performance with char n-grams, our method achieves the best macro-averaged F1 (74.88%), while BBR achieves 74.72% and SVM 70.43%. In Table 5 we show top 5 word-level and char-level n-gram features for the positive (Crime) and the negative (Drama) class. We can observe the following effects comparing the top word-level n-grams with the char n-grams in Table 5. The top-5 word-level features are highly discriminative unigrams, for both the positive and the negative class. The character-level n-grams extract characteristic substrings (word stems, syllables, etc.) of words and provide robustness to morphological variation of wordings and misspellings. For example, the n-gram *urde* a potential substring of *murder*, *murderer*,

Table 3: CHINESE training running times. Micro/Macro-averaged F1 for varying  $n$ .

max char n-gram length	n=1	n=3	n=5	n unrestricted
# overall features	4,961	1,588,488	6,107,182	N/A
# iterations SVM	146	116	148	N/A
# iterations BBR	223	222	226	N/A
# iterations SLR	187	184	184	184
# features in final model BBR	551	687	701	N/A
# features in final model SLR	120	131	131	131
Running Time SVM	0.1 min	1.5 min	5 min	N/A
Running Time BBR	0.5 min	4.4 min	11 min	N/A
Running Time SLR	0.25 min	0.5 min	0.5 min	0.5 min
Time for generating patterns	1 min	5 min	9 min	N/A
Total time SVM	1.1 min	6.5 min	14 min	N/A
Total time BBR	1.5 min	6.9 min	20 min	N/A
<b>Total Time SLR</b>	<b>0.25 min</b>	<b>0.5 min</b>	<b>0.5 min</b>	<b>0.5 min</b>
microavgF1 SVM	72.82%	80.93%	78.96%	N/A
microavgF1 BBR	73.18%	76.65%	76.85%	N/A
<b>microavgF1 SLR</b>	<b>77.39%</b>	<b>78.87%</b>	<b>78.87%</b>	<b>78.87%</b>
macroavgF1 SVM	73.29%	80.66%	78.60%	N/A
macroavgF1 BBR	73.74%	77.43%	77.52%	N/A
<b>macroavgF1 SLR</b>	<b>77.69%</b>	<b>78.54%</b>	<b>78.54%</b>	<b>78.54%</b>

Table 4: All collections. Tuned parameters. Micro/Macro-averaged F1 for varying  $n$ .

Collection	max n-gram length	word n-grams				char n-grams			
		n=1	n=3	n=5	n unrestricted	n=1	n=3	n=5	n unrestricted
IMDB	microavgF1 SVM	69.21%	71.11%	71.22%	N/A	57.47%	65.65%	70.41%	N/A
	microavgF1 BBR	73.70%	73.58%	73.92%	N/A	56.97%	73.54%	74.71%	N/A
	<b>microavgF1 SLR</b>	<b>74.02%</b>	<b>73.74%</b>	<b>73.70%</b>	<b>73.86%</b>	<b>58.44%</b>	<b>73.82%</b>	<b>74.44%</b>	<b>74.87%</b>
	macroavgF1 SVM	69.21%	71.13%	71.24%	N/A	57.47%	65.66%	70.43%	N/A
	macroavgF1 BBR	73.76%	73.63%	73.94%	N/A	57.07%	73.56%	74.72%	N/A
	<b>macroavgF1 SLR</b>	<b>74.04%</b>	<b>73.77%</b>	<b>73.72%</b>	<b>73.91%</b>	<b>58.43%</b>	<b>73.89%</b>	<b>74.47%</b>	<b>74.88%</b>
AMAZON	microavgF1 SVM	82.09%	80.60%	78.49%	N/A	39.32%	79.14%	81.88%	N/A
	microavgF1 BBR	84.58%	84.14%	83.99%	N/A	43.62%	82.98%	84.21%	N/A
	<b>microavgF1 SLR</b>	<b>84.22%</b>	<b>83.64%</b>	<b>83.75%</b>	<b>83.57%</b>	<b>45.54%</b>	<b>83.05%</b>	<b>84.21%</b>	<b>84.27%</b>
	macroavgF1 SVM	80.68%	78.54%	75.79%	N/A	34.91%	77.47%	80.52%	N/A
	macroavgF1 BBR	83.44%	82.95%	82.81%	N/A	40.62%	81.77%	82.97%	N/A
	<b>macroavgF1 SLR</b>	<b>83.16%</b>	<b>82.43%</b>	<b>82.55%</b>	<b>82.30%</b>	<b>42.61%</b>	<b>81.78%</b>	<b>82.94%</b>	<b>83.01%</b>
CHINESE	microavgF1 SVM	N/A	N/A	N/A	N/A	72.82%	80.93%	78.96%	N/A
	microavgF1 BBR	N/A	N/A	N/A	N/A	78.36%	78.88%	78.58%	N/A
	<b>microavgF1 SLR</b>	<b>N/A</b>	<b>N/A</b>	<b>N/A</b>	<b>N/A</b>	<b>76.18%</b>	<b>78.71%</b>	<b>78.30%</b>	<b>79.17%</b>
	macroavgF1 SVM	N/A	N/A	N/A	N/A	73.29%	80.66%	78.60%	N/A
	macroavgF1 BBR	N/A	N/A	N/A	N/A	78.17%	79.63%	79.36%	N/A
	<b>macroavgF1 SLR</b>	<b>N/A</b>	<b>N/A</b>	<b>N/A</b>	<b>N/A</b>	<b>76.09%</b>	<b>79.35%</b>	<b>79.06%</b>	<b>79.75%</b>

*murdering*, is selected as a positive feature for the Crime class (962 times in Crime vs 303 times in Drama). Note that *urde* rather than *murde* is selected because adding the  $m$  does not increase (in this particular case) the discrimination power of this feature. Other examples are the prefix *lov*, from *love*, *loving*, *loveable*, a feature much more frequent in the Drama movie plots (925 times), than the Crime plots (470 times). Similarly, substrings such as *choo* from *school*, *schools*, *schooling*, etc. are chosen as characteristics of the Drama class (300 times in Drama, 110 times in Crime). The macro-averaged F1 for the word n-gram and the char n-gram models is comparable, e.g. 74.04% versus 74.88% both achieved with SLR.

#### 4.4.2 AMAZON dataset

Similarly, we run 5-fold cross validation on the AMAZON dataset and report the micro/macro-averaged F1 results (averaged over cross-validation splits). In Table 2 we present training running times on this dataset for all methods. Again, SLR is much faster than BBR and SVM, with highest running time of 0.2 minutes for unrestricted word n-grams, and 1.5 minutes for unrestricted char n-grams. BBR

takes 5.4 minutes for learning a word-level 5-gram model and 9.6 minutes for learning a 5-gram char model. SVM takes somewhat longer with 24 minutes average runtime for word-level 5-gram model and 55 minutes for char-level 5-gram model. In terms of classification quality, for the fixed parameters setting (Table 1) SLR is better by 2-3% than either BBR or SVM. In the tuned setting (Table 4), SLR is comparable to BBR (word-level: 83.16% versus 83.44%; char-level: 83.01% versus 82.97%) and SVM (word-level: 80.68%; char-level: 80.52%).

In Table 5 we show top-5 positive and negative n-grams for this dataset. We observe the same effect of implicit stemming for the character n-grams as in IMDB. For example, our model selects features such as *bio*, instead of *biology*, or *mat* instead of *mathematics*. These features carry already enough information for discriminating the given topics, thus there is no need for selecting the entire word. Note the features such as *olo* which seem unexpected at a first glance. The reason for selecting *olo* is that it is contained in words such as *biology*, *biologist*, *biological*, *ecology*, etc., thus it is by itself already good for discriminating between Biology and

Table 5: Top 5 features IMDB and AMAZON. Max n-gram length  $n=5$ .

Methods	IMDB Crime(pos) vs Drama(neg)		AMAZON Biology (pos) vs Mathematics-Physics(neg)	
	SLR	BBR	SLR	BBR
top-5 pos word n-grams	0.139 crime 0.125 police 0.105 detective 0.077 murder 0.068 gang	9.673 has done 8.348 postwar 7.631 extortion 7.562 recorded 7.203 dependent on	0.093 species 0.086 human 0.081 biological 0.074 biology 0.072 ecology	8.148 which they can 7.484 ecology 7.108 biochemistry 7.067 bioinformatics 6.542 ecological
top-5 neg word n-grams	-0.057 school -0.050 war -0.049 family -0.046 love -0.035 her	-8.094 a wild -7.803 his way to -6.888 is the owner -6.624 life in -6.419 onto his	-0.092 physics -0.084 mathematics -0.077 theory -0.071 mathematical -0.057 statistics	-6.100 calculus -5.402 physics -5.106 stars -4.623 geometry -4.408 chaos
top-5 pos char n-grams	0.040 u r d e 0.039 g a n g 0.038 o l i c e 0.034 c r i m 0.021 b o s	3.244 c h t o t 3.238 b w 3.217 a a r 3.077 b y o n 2.956 e B o	0.064 B i o 0.060 o l o 0.048 b i o 0.046 p e c i e 0.045 i o l o g	3.050 E c o l 2.478 i o l 2.301 c o l o 2.273 B i 2.153 n i m
top-5 neg char n-grams	-0.020 l o v -0.015 t i o n -0.013 r i n -0.011 c h o o -0.011 l d	-3.150 c h u n -2.635 a b r o -2.289 a t s e r -2.278 t h e r n -2.262 i n i s t	-0.076 e m a t -0.060 i c s -0.043 h e o r -0.031 s i c -0.030 M a t	-2.792 M a t -2.667 S t a t i -2.334 T e -2.144 M a t h -1.797 a l c u

Mathematics-Physics. This sort of features may seem prone to overfitting, but our learning method is robust enough to decide on inclusion of only highly discriminative features. Examples of syllable extraction are features like *ics* instead of *physics*, *mathematics*, *statistics*, etc. Misspellings, such as *physics* versus *pyhsics*, can influence the features much less with this kind of representation. Char n-grams could also potentially capture other effects of language use, such as re-named entities, e.g. *Alon Halevy*, *A. Halewi*, *A. Halevy*, slang, e.g. *Eire for Ireland*, and abbreviations, e.g. *math* instead of *mathematics*.

#### 4.4.3 CHINESE dataset

This dataset is split into training and test, thus we train all models on the training set and report micro/macro-averaged F1 on the test set. The training set contains 4,961 distinct characters, and about 6,000,000 char n-grams up to size 5. Table 3 shows training run-times for this dataset. SLR takes 0.5 minutes for learning a model in the space of char-level n-grams of unrestricted length. BBR needs 11 minutes, while SVM needs 5 minutes for learning 5-gram models. These running times do not include the time required for pattern generation for BBR and SVM (9 extra minutes). Regarding prediction quality, SLR achieves 79.75% macro-averaged F1 which compares well to the 80.66% achieved by SVM and 79.63% achieved by BBR.

Tables 1 to 4 show micro/macro-averaged F1 behavior for all methods and corpora for varying maximum n-gram length  $n$ . For word n-grams, we note that a higher order  $n$  improves the quality of the models in the case of fixed parameters, but not in the case of tuned parameters. For char n-grams, we observe that all methods benefit from using higher order n-gram features, also in the tuned setting. Overall, char n-gram models seem to be at least as good and often better than word n-gram models. Furthermore, the accuracy of SLR models with fixed parameter is close to that of tuned SLR models, thus reducing the need for careful tuning of the number of iterations. Although the space of variable length n-grams is very large, our method can ef-

ficiently learn accurate models, thus avoiding the need for additional pre-processing, such as feature selection or word-segmentation.

## 5. CONCLUSION

In this paper we present a coordinate-wise gradient ascent technique for learning logistic regression in the space of all (word or character) n-gram sequences in the training data. Our method exploits the inherent structure of the n-gram feature space in order to automatically provide a compact set of highly discriminative n-gram features.

We propose a theoretical bound which quantifies the “goodness” of the gradient for each n-gram candidate given its length- $(n - 1)$  prefix. We show that due to the proposed bound, we can efficiently work with variable-length n-gram features both at the word-level and the character-level.

We present experiments that compare our *SLR* method against the state-of-the-art classifiers *BBR* (a logistic regression method) [9] and *SVM<sup>perf</sup>* [17]. We show that while *SLR* generalizes as good as the state-of-the-art methods, it is more than one order of magnitude faster than its opponents.

With the method presented in this paper we study the problem of learning the tokenization of the input text, rather than explicitly fixing it in advance (as in the bag of words model). The tokens learned by *SLR* can be arbitrary sized, rather than restricted to a hypothesized “good” size. This opens interesting research directions, such as supervised entity extraction and unsupervised or semi-supervised text clustering. Furthermore, our technique can be applied to other domains such as gene sequence classification, where mining variable-length sequences is of particular importance.

Open source code for *SLR* is available on-line at: <http://www.mpi-inf.mpg.de/~ifrim/slr>.

## 6. ACKNOWLEDGMENTS

We thank the reviewers for their helpful comments and Fernando Pereira for suggesting the relevant work on prediction suffix trees [5].

## 7. REFERENCES

- [1] K. Abe, S. Kawasoe, T. Asai, H. Arimura, and S. Arikawa. Optimized substructure discovery for semi-structured data. In *Proceedings of PKDD*. Springer-Verlag, 2002.
- [2] T. C. Bell, J. G. Cleary, and I. H. Witten. *Text Compression*. Prentice-Hall, 1990.
- [3] H. Cheng, X. Yan, J. Han, and C.-W. Hsu. Discriminative frequent pattern analysis for effective classification. In *Proceedings of ICDE*, pages 716–725, 2007.
- [4] J. G. Cleary and W. J. Teahan. Unbounded length contexts for PPM. *Computer Journal*, 3(40):67–75, 1997.
- [5] O. Dekel, S. Shalev-Shwartz, and Y. Singer. The power of selective memory: Self-bounded learning of prediction suffix trees. In *Proceedings of NIPS*, Vancouver, Canada, 2004.
- [6] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of CIKM*, pages 148–155. ACM, 1998.
- [7] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2)(4047499), 2004.
- [8] E. Frank, C. Chui, and I. H. Witten. Text categorization using compression models. In *Proceedings of the Data Compression Conference*, page 555, Snowbird, UT, July 2000.
- [9] A. Genkin, D. Lewis, and D. Madigan. Large-scale Bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304, 2007.
- [10] J. Goodman. A bit of progress in language modeling. In *Technical report*. Microsoft Research, 2001.
- [11] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics, 2003.
- [12] J. He, A.-H. Tan, and C.-L. Tan. On machine learning methods for Chinese documents classification. *Applied Intelligence*, 18(3):311–322, 2003.
- [13] D. Holmes and R. Forsyth. The Federalist revisited: New directions in authorship attribution. *Literary and Linguistic Computing*, 2(10):111–127, 1995.
- [14] G. Ifrim and G. Weikum. Transductive learning for text classification using explicit knowledge models. In *Proceedings of PKDD*, Springer Lecture Notes in Artificial Intelligence, pages 223–234, Berlin, Germany, 2006.
- [15] R. Jin, R. Yan, J. Zhang, and A. Hauptmann. A faster iterative scaling algorithm for conditional exponential model. In *Proceedings of ICML*, 2003.
- [16] T. Joachims. Text categorization with Support Vector Machines: learning with many relevant features. In C. Nédellec and C. Rouveiol, editors, *Proceedings of ECML*, pages 137–142, Chemnitz, DE, 1998. Springer Verlag.
- [17] T. Joachims. Training linear SVMs in linear time. In *Proceedings of SIGKDD*, pages 217–226, New York, NY, 2006. ACM Press.
- [18] B. Kessler, G. Nunberg, and H. Schtze. Automatic detection of text genre. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 1997.
- [19] P. Komarek and A. Moore. Fast robust logistic regression for large sparse datasets with binary outputs. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, New York, NY, 2003.
- [20] A. C. König and E. Brill. Reducing the human overhead in text categorization. In *Proceedings of SIGKDD*, pages 598–603, New York, NY, USA, 2006. ACM.
- [21] T. Kudo. An implementation of freqt (frequent tree miner). <http://chasen.org/~taku/software/freqt/>, 2003.
- [22] T. Kudo and Y. Matsumoto. A boosting algorithm for classification of semi-structured text. In *Proceedings of EMNLP*, pages 301–308, Barcelona, Spain, July 2004. Association for Computational Linguistics.
- [23] Y.-B. Lee and S. H. Myaeng. Text genre classification with genre-revealing and subject-revealing features. In *Proceedings of SIGIR*, pages 145–150, Tampere, Finland, 2002.
- [24] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. In *Journal of Machine Learning Research*, pages 419–444, 2001.
- [25] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, 1999.
- [26] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operation Research and Financial Engineering, 2006.
- [27] F. Peng, D. Schuurmans, and S. Wang. Augmenting Naive Bayes text classifier with statistical language models. *Information Retrieval*, 3(7):317–245, 2004.
- [28] R. Rosenfeld. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88(8):1270–1278, 2000.
- [29] B. Scholkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.
- [30] S. K. Shevade and S. S. Keerthi. A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics*, 19:2246–2253, 2003.
- [31] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of SIGIR*, pages 42–49, Berkeley, CA, 1999.
- [32] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of ICML*, pages 412–420, Nashville, TN, 1997.
- [33] M. Zaki. Efficiently mining frequent trees in a forest. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, 2002.
- [34] D. Zhang and W. S. Lee. Extracting key-substring-group features for text classification. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 474–483, New York, NY, USA, 2006. ACM.
- [35] T. Zhang and F. Oles. Text categorization based on regularized linear classifiers. *Information Retrieval*, 4(1):5–31, 2001.