Thesis for obtaining the title of Doctor of Engineering of the Faculties of Natural Sciences and

Technology of Saarland University

Statistical Learning Techniques for Text Categorization with Sparse Labeled Data

Georgiana Ifrim

Supervisor:
Prof. Dr.-Ing. Gerhard Weikum

Max-Planck Institute for Informatics
Saarbrücken, Germany
2009

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Saarbrücken, den 03. März 2009

(Unterschrift)

Acknowledgements

The work environment at the Max-Planck Institute for Informatics is truly special. One is surrounded by very smart and interesting people who are genuinely curious about difficult scientific problems, as well as difficult problems facing our current world.

First and foremost I would like to thank my supervisor Prof. Gerhard Weikum, who plays a very big role in making the MPII such an exciting work environment. He was always there listening to my ideas, giving guidance and support toward interesting and promising research directions, and most of all sharing his optimistic and mature approach toward solving problems, which made it so much easier to cope with the PhD life challenges.

I would also like to thank my office colleagues, Fabian Suchanek and Gjergji Kasneci. It was really a pleasure to share the office with you guys. I always enjoyed both our technical and our philosophical discussions, and most of all your openness toward approaching challenging problems. I am grateful to my husband and colleague Deepak Ajwani, for always patiently listening to my ideas and many times coming up with a fresh view on the issues. Many thanks to my collaborators Maya Ramanath and Gökhan Bakır, working with you was a very rewarding experience. Thanks to my colleagues Christos Tryfonopoulos, Ralitsa Angelova and Julia Luxenburger for their suggestions and support during the course of this work.

Last but not least, I am grateful to the International Max Planck Research School for its financial support during these years, and for always encouraging a productive research environment.

Abstract

Many applications involve learning a supervised classifier from very few explicitly labeled training examples, since the cost of manually labeling the training data is often prohibitively high. For instance, we expect a good classifier to learn our interests from a few example books or movies we like, and recommend similar ones in the future, or we expect a search engine to give more personalized search results based on whatever little it learned about our past queries and clicked documents.

There is thus a need for classification techniques capable of learning from sparse labeled data, by exploiting additional information about the classification task at hand (e.g., background knowledge) or by employing more sophisticated features (e.g., n-gram sequences, trees, graphs). In this thesis, we focus on two approaches for overcoming the bottleneck of sparse labeled data.

We first propose the **Inductive/Transductive Latent Model (ILM/TLM)**, which is a new generative model for text documents. ILM/TLM has various building blocks designed to facilitate the integration of background knowledge (e.g., unlabeled documents, ontologies of concepts, encyclopedia) into the process of learning from small training data. Our method can be used for inductive and transductive learning and achieves significant gains over state-of-the-art methods for very small training sets.

Second, we propose **Structured Logistic Regression (SLR)**, which is a new coordinate-wise gradient ascent technique for learning logistic regression in the space of all (word or character) sequences in the training data. SLR exploits the inherent structure of the n-gram feature space in order to automatically provide a compact set of highly discriminative n-gram features. Our detailed experimental study shows that while SLR achieves similar classification results to those of the state-of-the-art methods (which use all n-gram features given explicitly), it is more than an order of magnitude faster than its opponents.

The techniques presented in this thesis can be used to advance the technologies for automatically and efficiently building large training sets, therefore reducing the need for spending human computation on this task.

Kurzfassung

Viele Anwendungen benutzen Klassifikatoren, die auf dünn gesäten Trainingsdaten lernen müssen, da es oft aufwändig ist, Trainingsdaten zur Verfügung zu stellen. Ein Beispiel fuer solche Anwendungen sind Empfehlungssysteme, die auf der Basis von sehr wenigen Büchern oder Filmen die Interessen des Benutzers erraten müssen, um ihm ähnliche Bücher oder Filme zu empfehlen. Ein anderes Beispiel sind Suchmaschinen, die sich auf den Benutzer einzustellen versuchen, auch wenn sie bisher nur sehr wenig Information üeber den Benutzer in Form von gestellten Anfragen oder geklickten Dokumenten besitzen.

Wir benötigen also Klassifikationstechniken, die von dünn gesäten Trainingsdaten lernen können. Dies kann geschehen, indem zusätzliche Information über die Klassifikationsaufgabe ausgenutzt wird (z.B. mit Hintergrundwissen) oder indem raffiniertere Merkmale verwendet werden (z.B. n-Gram-Folgen, Bäume oder Graphen). In dieser Arbeit stellen wir zwei Ansätze vor, um das Problem der dünn gesäten Trainingsdaten anzugehen.

Als erstes schlagen wir das Induktiv-Transduktive Latente Modell (ILM/TLM) vor, ein neues generatives Modell für Text-Dokumente. Das ILM/TLM verfügt über mehrere Komponenten, die es erlauben, Hintergrundwissen (wie z.B. nicht Klassifizierte Dokumente, Konzeptontologien oder Enzyklopädien) in den Lernprozess mit einzubeziehen. Diese Methode kann sowohl für induktives als auch für transduktives Lernen eingesetzt werden. Sie schlägt die modernsten Alternativmethoden signifikant bei dünn gesäten Trainingsdaten.

Zweitens schlagen wir Strukturierte Logistische Regression (SLR) vor, ein neues Gradientenverfahren zum koordinatenweisen Lernen von logistischer Regression im Raum aller Wortfolgen oder Zeichenfolgen in den Trainingsdaten. SLR nutzt die inhärente Struktur des n-Gram-Raums aus, um automatisch hoch-diskriminative Merkmale zu finden. Unsere detaillierten Experimente zeigen, dass SLR ähnliche Ergebnisse erzielt wie die modernsten Konkurrenzmethoden, allerdings dabei um mehr als eine Größenordnung schneller ist.

Die in dieser Arbeit vorgestellten Techniken verbessern das Maschinelle Lernen auf dünn gesäten Trainingsdaten und verringern den Bedarf an manueller Arbeit.

Summary

Many applications involve learning a supervised classifier from very few explicitly labeled training examples, since the cost of manually labeling the training data is often prohibitively high. For instance, we expect a good classifier to learn our interests from a few example books or movies we like, and recommend similar ones in the future, or we expect a search engine to give more personalized search results based on whatever little it learned about our past queries and clicked documents.

There is thus a need for classification techniques capable of learning from sparse labeled data, by exploiting additional information about the classification task at hand (e.g., background knowledge) or by employing more sophisticated features (e.g., n-gram sequences, trees, graphs). In this thesis, we focus on two approaches for overcoming the bottleneck of sparse labeled data.

We first propose the **Inductive/Transductive Latent Model (ILM/TLM)**, which is a new generative model for text documents. ILM/TLM has various building blocks designed to facilitate the integration of background knowledge (related to the domain of the labeled data) into the process of learning from small training data. We advocate using external ontologies for instantiating the structure of our latent model, rather than selecting an appropriate structure by time-consuming model selection strategies. Such ontologies are currently growing and are freely available. We give an Expectation-Maximization algorithm for learning the parameters of ILM/TLM. The parameter space can be huge, but we propose pruning it by learning a prior on the model parameters based on background knowledge. For example, if the training data is too sparse for learning robust parameter values, a prior which relies on context-similarity in the given corpus and external sources, can help improve the final predictions by 10%. Additionally, background knowledge, e.g., encyclopedia such as Wikipedia, can be used to explicitly or implicitly extend the topic descriptions provided by the training set. Empirical results show that the additional flexibility of ILM/TLM offered by its various building blocks results in improved classification results for small training sets, as compared to other state-of-the-art classifiers. Additionally ILM/TLM has the advantage of interpretability and robustness to training/test distribution shifts. We analyze different ways of setting parameters, and the effect of each building block on the overall model performance. Last but not least, we show that taking advantage of unlabeled data, which is abundantly available in many applications, improves the results of our model.

Second, we propose **Structured Logistic Regression (SLR)**, which is a new coordinate-wise gradient ascent technique for learning logistic regression in the space of all (word or character) sequences in the training data. SLR 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 give theoretical bounds which quantify the "goodness" of the gradient for each n-gram candidate given its length-(n-1) prefix. We show that by using the proposed bounds, we can efficiently work with variable-length n-gram features both at the word-level and the character-level. Our detailed experimental study shows that while SLR achieves similar classification results to those of the state-of-the-art methods (which use all n-gram features given explicitly), it is more than an order of magnitude faster than its opponents. We also consider the problem of learning the tokenization of the input text, rather than explicitly fixing it in advance (as in the bag-of-words model). We show that SLR can be used to learn arbitrarily sized discriminative n-grams, rather than n-grams that are restricted to a hypothesized "good" size.

Given the flexibility of SLR for learning variable-length n-gram patterns, this model could be applied to supervised information extraction for learning patterns that are indicative of binary relations. Additionally, SLR can be applied to other domains such as biological sequence classification, where mining variable-length sequences is of particular importance. Theoretical results similar to those presented in this thesis for learning with sequences apply directly to trees or graph representations (for example for XML documents), with only a few implementation modifications. This is true because the simple monotonicity property needed by our proofs holds also in the case of more complex structures such as trees and graphs.

The techniques presented in this thesis can be used to advance the technologies for automatically and efficiently building large training sets, therefore reducing the need for spending human computation on this task.

Zusammenfassung

Viele Anwendungen benutzen Klassifikatoren, die auf dünn gesäten Trainingsdaten lernen müssen, da es oft aufwändig ist, Trainingsdaten zur Verfügung zu stellen. Ein Beispiel fuer solche Anwendungen sind Empfehlungssysteme, die auf der Basis von sehr wenigen Buechern oder Filmen die Interessen des Benutzers erraten müssen, um ihm ähnliche Bücher oder Filme zu empfehlen. Ein anderes Beispiel sind Suchmaschinen, die sich auf den Benutzer einzustellen versuchen, auch wenn sie bisher nur sehr wenig Information üeber den Benutzer in Form von gestellten Anfragen oder geklickten Dokumenten besitzen.

Wir benötigen also Klassifikationstechniken, die von dünn gesäten Trainingsdaten lernen können. Dies kann geschehen, indem zusätzliche Information über die Klassifikationsaufgabe ausgenutzt wird (z.B. mit Hintergrundwissen) oder indem raffiniertere Merkmale verwendet werden (z.B. n-Gram-Folgen, Bäume oder Graphen). In dieser Arbeit stellen wir zwei Ansätze vor, um das Problem der dünn gesäten Trainingsdaten anzugehen.

Als erstes schlagen wir das Induktiv-Transduktive Latente Modell (ILM/TLM) vor, ein neues generatives Modell für Text-Dokumente. Das ILM/TLM hat verschiedene Komponenten, die es erlauben, Hintergrundwissen in den Lernprozess mit einzubeziehen. Wir schlagen außerdem vor, externe Ontologien zur Initialisierung des latenten Modells zu verwenden, anstatt die passende Anzahl der verborgenen Variablen mit zeitaufwändigen Modellselektionsstrategien zu finden. Solche Ontologien sind frei verfügbar und werden immer größer. Wir beschreiben einen Expectation-Maximization-Algorithmus zum Lernen der Parameter des ILM/TLM. Der Parameterraum kann sehr groß werden, aber wir zeigen, wie man ihn einschränken kann, indem man ein A-Priori-Modell lernt, das durch das Hintergrundwissen bestimmt ist. Wenn beispielsweise die Trainingsdaten zu dünn gesät sind, um robuste Parameter zu lernen, so kann ein A-Priori-Modell, das auf kontextueller Ähnlichkeit im Korpus und externen Quellen beruht, die Vorhersagen um 10% verbessern. Zusätzlich kann Hintergrundwissen (z.B. in Form der Online-Enzyklopädie Wikipedia) benutzt werden, um die Trainingsdaten implizit oder explizit zu erweitern. Empirische Studien zeigen, dass die Flexibilität des ILM/TLM mit seinen vielfältigen Komponenten die Klassifikationsergebnisse im Vergleich zu anderen modernen Methoden bei dünn gesäten Trainingsdaten verbessert. Wir untersuchen verschiedene Konfigurationen des Systems und die Auswirkungen jeder einzelnen Komponente auf die Gesamtperformanz. Schließlich zeigen wir auch, dass nicht Klassifizierte Daten, welche in vielen Anwendungsfällen reichlich vorhanden sind, die Ergebnisse mit unserem Modell noch verbessern können.

Zweitens schlagen wir Strukturierte Logistische Regression (SLR) vor, ein neues Gradientenverfahren zum koordinatenweisen Lernen von logistischer Regression im Raum aller Wortfolgen oder Zeichenfolgen in den Trainingsdaten. SLR nutzt die inhärente Struktur des n-Gram-Raums aus, um automatisch hoch-diskriminative Merkmale zu finden. Wir beweisen theoretische Schranken für die Güte der Gradienten aller n-Gramme, die mit einem gegebenen (n-1)-stelligen Präfix anfangen. Mithilfe dieser Schranken kann unsere Methode effizient auf n-Grammen mit variabler Länge arbeiten – sowohl auf der Zeichen-Ebene als auch auf Wort-Ebene. Unsere detaillierten Experimente zeigen, dass SLR ähnliche Ergebnisse erzielt wie die modernsten Konkurrenzmethoden, allerdings dabei um mehr als eine Größenordnung schneller ist. Wir betrachten auch das Problem, die lexikalische Analyse des Eingabetexts zu lernen, anstatt sie im Vorhinein festzulegen (wie es beim Bag-of-Words-Modell geschieht). Wir zeigen, dass SLR beliebige diskriminative n-Gramme lernen kann, und nicht nur n-Gramme einer im Voraus festgelegten Maximal-länge.

Da SLR mit n-Grammen variabler Länge umgehen kann, könnte das Modell in der unüberwachten Informationsextraktion eingesetzt werden, um Textmuster zu lernen, die binäre Relationen ausdrücken. SLR kann auch in anderen Gebieten eingesetzt werden, z.B. bei der Klassifizierung von biologischen Sequenzen, wo das Auffinden von n-Grammen mit variabler Länge eine besondere Rolle spielt. Die hier vorgstellten theoretischen Ergebnisse lassen sich mit wenigen Änderungen auch auf Baum- oder Graph-Darstellungen (wie beispielsweise XML-Dokumente) übertragen, da die einfache Monotonie-Bedingung, auf der unsere Beweise basieren, ebenso für komplexere Strukturen wie Bäume und Graphen gilt.

Die in dieser Arbeit vorgestellten Techniken verbessern das Maschinelle Lernen auf dünn gesäten Trainingsdaten und verringern den Bedarf an manueller Arbeit.

Contents

Chapter 1

Introduction

1.1 Text Categorization: A Short Overview

Text Categorization (also known as Text Classification) is the process of organizing text documents into pre-defined categories according to the documents' content. More explicitly, given a set of example documents for each category, we want to *automatically* learn the characteristics of each category, or learn a model of each category, which we can then apply to assign categories to a new (unlabeled) document. The models of each category learned this way are also called *classifiers*.

Until the late 1980's, classifiers were built manually by a set of expert users, who needed to invest a huge amount of time into finding a comprehensive set of rules for characterizing the given set of categories. Each set of rules per category would then be used to decide whether the respective category should be assigned to an unlabeled document. With the sudden increase in the amount of digital information in the late 1990's, this type of manual system building became infeasible, since manually designing application-dependent rules is both slow and very expensive. The new approaches that took over around this time were based on *machine learning techniques* and relied on automatically learning characteristics of each category based on a set of labeled example documents.

Text categorization has many applications, in various fields and for various types of data. Many problems related to data storage, management and retrieval can be formulated in terms of text classification. The most common example of text classification is that of organizing news according to their topic, e.g. into Sports, Politics, Economics or categorizing large Web document collections into Web directories for easier browsing and retrieval, e.g. the Yahoo Directory [CM07], the Wikipedia free encyclopedia [Wik], etc. The use of text categorization is nevertheless much wider, ranging from automated or computer-aided patient diagnosis in hospitals [RRN$^+$06], where classifiers learned on the medical history of previous patients can help the doctor decide the type of treatment for a new patient, to automatically providing movies or book recommendations for on-line shopping portals such as Amazon [KB07].

From the wide spectrum of applications of text categorization we further mention:

- *Product categorization* [FM01]: Large on-line shopping portals such as eBay automatically organize their product descriptions into categories for facilitating the browsing of their products for the customers.

- *Search personalization* [KSG$^+$03, LEW08]: Many search engines keep track of queries and users browsing activities in order to improve their search experience, by matching retrieved information content to a user's profile. Advertisement classification is another important application. Classifying ads can help targeted advertisement, a very important source of revenue for most search engines.

- *Document filtering for digital libraries* [Seb01, Zha04]: Categorizing documents based on their content can facilitate the search and browsing of documents in digital libraries. Additionally, in publish-subscribe systems, users can register to be notified when new documents belonging to certain categories arrive in the system [MFP06].

- *Authorship detection* [HF95, SKF00, SKS07]: Typically employed for fraud detection or for texts which do not bare the name of the original author.

- *Product reviews classification* [PLV02, PL04, ZZ06, ZV06]: Organizing the user feedback regarding certain products or services into positive, negative and even neutral categories. Most users first read other users' reviews before deciding to purchase a certain product. Automatically organizing the user feedback can help both the company and the users to better assess the products on sale.

- *Analyzing public opinion* or *sentiment mining* [NH06, ZZ06, ZJ08, YNBN03, WBB$^+$03]: Mining data about companies, individual products and services or even political polls.

- *E-mail filtering* [KPA08, BS07, BFC$^+$06, SACY04, RZ04]: Automatically filtering out unsolicited e-mail (i.e., spam).

The most widely used document representation for text categorization is that of a *bag-of-words*, in which only the distinct words of each document are preserved, while discarding all the order information. For most topical classification tasks, e.g., applications in which there is a direct correlation between individual words and the category name or meaning, such a representation suffices for capturing enough of the semantics of the document for achieving high classification quality. For non-topical classification, where there is no direct relation between individual keywords and the categories of interest, such representation may be insufficient, and a lot of research has gone into investigating the use of other representations (e.g. n-gram representation) for better capturing the needs of such applications [HF95, KNS97, LM02, PSW04, ZL06, IBW08].

The simplest example of a fully automated classifier is the *Naive Bayes (NB) classifier* [MN98a], which uses frequency statistics of the words in the document set of each category, and a probabilistic model for assigning categories to documents. Naive Bayes makes assumptions about the way the input data is distributed: it assumes documents in each topic are generated according to a given distribution (e.g., Multinomial), and that words in a document are generated conditionally independent of other words, given the category label. Because NB makes assumptions about the input data distribution it belongs to the class of *generative classifiers*. On the positive side this classifier is very easy to implement, very fast, and reasonably accurate in practice, which led to its wide spread use in research on text classification [Lew98, MN98a, RSTK03]. On the negative side, its probability estimates are not accurate, and it is outperformed by more sophisticated classifiers [Joa98, CNM06].

Another type of classifiers which became the de facto method of choice because of its high accuracy are *Support Vector Machines (SVM)* [Vap98, Joa98]. SVM take a different approach to data classification, aiming at directly inferring a mapping between documents and their categories, without relying on intermediate assumptions about the data generation process. SVM can be regarded as a geometric approach in which documents are seen as vectors in a vector space spanned by all the distinct words in the collection. Learning an SVM classifier for the case of two categories is then equivalent to finding the hyperplane that separates the two categories with the largest margin. For more than two categories, several classifiers (one per category) are learned. SVM falls in the category of *discriminative classifiers*; it does not make any assumptions on or any attempt at modeling the input distribution. SVM proved to be more accurate than Naive Bayes in many classification tasks [Joa98], which led to the popular belief that discriminative classifiers are preferable over generative classifiers.

Generative classifiers tend to be better when training data is small, and are more natural for treating missing data (learning from labeled and unlabeled data) or integrating assumptions about the model structure and prior parameter values. Discriminative classifiers are instead better when training data is plentiful, or when it is preferable to not make any assumptions about the way the input data was generated. At the same time discriminative approaches are more sensitive to noisy data [LB03]. Therefore a lot of work has investigated the benefits of combining these two type of approaches [JH99, BT04, RSNM03, NJ01].

1.2 Text Categorization with Sparse Labeled Data

Most existing classifiers can achieve very high classification accuracy when training data is plentiful (i.e., in the order of thousands). But, for most applications, labeled training data is difficult to provide in large numbers, since the labels are manually or at best semi-automatically acquired, which makes this process slow and costly. Moreover, digital data is growing at a very fast pace, making it harder to provide a representative set of labeled samples, to cover most aspects of the categories of interest. Therefore a lot of work has gone into studying which classifiers are more robust when training data is sparse and into various alternatives for enriching small amounts of labeled data. In this section we give a short summary of several existing approaches to overcoming the bottleneck of sparse labeled data.

Active Learning [MN98b, TKK00] tries to overcome the labeled data sparsity problem, by assuming that a large pool of unlabeled documents is available for the application of interest, and by allowing the learner to ask a human user to label a few carefully selected documents that are considered most relevant for learning a particular task. The learner's choice of documents to present to the user would typically prioritize documents for which the learner's label assignment is most uncertain, but can also include other criteria of importance. The parallel sometimes used in the literature between passive learning (where all the training data is labeled) and active learning, is that of a student that sits and listens to a teacher while an active learner is a student that asks the teacher questions, listens to the answers, and asks further questions based upon the teacher's response. Active learning was shown to significantly decrease the amount of labeled data necessary for achieving a certain classification accuracy [MN98b, TKK00, BBL06, Das04, DHM08, DH08].

Semi-supervised Learning [Zhu08] refers to the use of both labeled and unlabeled data for training. It contrasts supervised learning (all data is labeled) and unsupervised learning (all data is unlabeled). Labeled instances are often expensive while unlabeled data may be relatively easy to collect. Semi-supervised learning addresses this problem by using large amounts of unlabeled data, together with the labeled data, to build better classifiers. It has been shown that semi-supervised learning typically requires less human effort and can provide higher classification accuracy [NMTM00a, Zhu08, CSZ06] if there is a good match between the problem structure and the model assumptions. Semi-supervised learning methods use unlabeled data to either modify or re-prioritize hypotheses obtained from labeled data alone. EM (Expectation-Maximization) with generative mixture models, self-training, co-training, transductive support vector machines, and graph-based methods are a few examples of semi-supervised learning methods.

Transductive Learning [Zhu08, CSZ06] can be understood as a more specialized form of semi-supervised learning where the unlabeled test data is available during training. It is a simpler problem than *inductive learning* (where only the labeled training data, but no detailed knowledge of the test data is available during training). Research literature [Zhu05] has used the analogy that transductive learning is more like a take-home exam, while inductive learning is more like an in-class exam. An example of transductive learning arises in the context of information retrieval with relevance feedback [Bou02]. The unlabeled test set is stored in the database, and based on some labeled (i.e relevant and non-relevant) samples from the user, a classifier based on the labeled data and the entire unlabeled set can be built for finding more relevant documents. A lot of research work has investigated the benefits of transductive learning [Joa99b, Joa03, Zhu05, Bou02, EYG05, IW06].

Using *background knowledge* [ZH00, ZH01, ZH02, Zel02, IW06] for alleviating the labeled data sparsity falls in the semi-supervised learning type of approaches. For example, ontologies of concepts can help in coping with the ambiguity of words, by considering their usage in context; prior information (e.g. from an encyclopedia) about the categories of interest can help provide additional knowledge on model parameters [IW06]. Moreover, background knowledge can be used to bridge training and test samples [Zel02], i.e. training samples are related to test samples in the context of the background knowledge.

Transfer Learning [DYXY07, RMKD05, RK07, WD04] focuses on re-using labeled data from a different task, in order to reduce the need for labeled data for a new classification task, in a related domain. One example of transfer learning is that of learning prior distributions over classifier parameters. Achieving significant levels of transfer learning across tasks is a central problem in machine learning [tra, Car97]. However, transfer learning may actually hinder performance if the tasks are too dissimilar [RMKD05]. Therefore, one challenge for transfer learning research is to develop approaches that detect and avoid negative transfer using very little data from the target task.

Self-taught Learning [RBL$^+$07, DYXY08] is a new machine learning framework for using unlabeled data in supervised classification tasks. The main difference from semi-supervised learning is that the unlabeled data is not assumed to follow the same class label distribution as the labeled data. The main idea behind this type of learning paradigm, is that one uses the unlabeled data to discover basic features, which are then used for a better representation of the labeled data, together with a supervised classification algorithm. Therefore, one can use a large number of unlabeled images (or audio samples, or text documents) randomly downloaded from the Internet to improve performance on a given image (or audio, or text) classification task. Such unlabeled data is easier to obtain than in typical semi-supervised or transfer learning settings.

For many applications of text categorization in which individual words are not highly correlated with the categories of interest, changing the representation of documents from the simple bag-of-words to more sophisticated representations, e.g. n-grams of words, may be required. For example, in applications such as opinion mining [NH06, ZZ06] and spam filtering [KPA08, BS07], capturing sequences of words or characters rather than individual tokens (words or characters) may help improve the classification accuracy. Many studies have analyzed various document representations, ranging from the simple bag-of-words to n-gram representations [PS03, PSW04], syntactic parse trees [Mos06, KM04] or even more sophisticated semantically annotated structures [GM06b, BM07]. In a few cases it was noted that richer representations may help, but more often the simple bag-of-words representation turned out to be the most accurate. The reason for such results may have been that typically certain restrictions had to be imposed on the richness of the representation in order to avoid the combinatorial explosion of the feature space. Using unrestricted n-grams of variable length, or syntax parse trees results in very large feature spaces, and therefore it leads to computational challenges.

1.3 Contributions of this Thesis

Inductive and Transductive Text Classification using Explicit Knowledge Models

We present a generative model for inductive and transductive learning, coined **ILM/TLM (Inductive/Transductive Latent Model)**, which uses background knowledge for improving the quality of classification when the training data is sparse. Our algorithm uses ontologies of concepts in order to learn the structure of the model, and available unlabeled documents and encyclopedia in order to learn a prior distribution of model parameters.

Our generative model treats groups of related words as together describing a concept, while groups of concepts together describe a topic. In order to capture the dependencies between words, concepts and topics, ILM/TLM postulates the following generative process: words in documents are generated by concepts which in turn are generated by topics. This simulates the intuition that topics are described by finer grained concepts which in turn are described by several words. In order to estimate the word-concept and concept-topic mappings, we rely on an iterative Expectation-Maximization (EM) procedure for maximum likelihood estimation from the given training set. Additionally, for improving the parameter estimation, we use several background knowledge resources to infer a priori word-concept and concept-topic mappings. We show that using background knowledge for the application of interest improves classification accuracy, as compared to using just a small set of labeled training examples. We also show that our model is competitive to state-of-the-art classification techniques, and that for small training data our technique outperforms its competitors. This work has been published in [ITW05] and [IW06].

The process of creating ontologies was until recently a manual and therefore very tedious and slow process [Fel99]. We contributed to learning ontologies automatically by developing techniques for supervised information extraction from unstructured text [SIW06]. Once we know how to automatically create large and accurate ontologies, it is necessary to be able to retrieve information from such large structures in an efficient and meaningful way. In [KSI$^+$08b] and [KSI$^+$08a], we presented work on querying and retrieving ranked information from ontologies.

Transductive Learning for Topic-driven Clustering. Clustering algorithms typically find the structure present in a given dataset, in terms of groups of highly similar documents. The clusters found by such algorithms do not always correspond to the user needs, e.g. the user may desire to specify what type of grouping she is interested in and even specify a few keywords describing each of the groups or categories she would like to get from the clustering algorithm. This setting falls to some degree in the area of semi-supervised clustering approaches, but the problem such techniques may face is the extreme sparsity of explicitly labeled examples, since here the training data is given as very brief keyword descriptions of the categories of interest. In this setting, using auxiliary knowledge resources in order to enrich the original training set, can help improve the clustering quality compared to the solution that just relies on user provided keywords. This framework was coined *topic-driven clustering* in [ZK05]. We show that our transductive generative model can provide good results for such a task, in particular by retraining the model on its previous predictions, as a form of iterative self-training.

Structured Logistic Regression

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 stop-words removal or stemming. Furthermore, a feature selection step [YP97] 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 entails major tuning of the classifiers in order to find the right unigram features.

In this thesis we present a new learning algorithm for text classification which uses variable-length n-gram sequences as features. As opposed to prior work, we do not restrict the length of the n-gram features; in principle our n-gram features can be as long as the longest document. We consider both word-level n-grams to capture phrases, and character-level n-grams to capture morphological variations (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. 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 efficiently 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) [GLM06], SVM^{perf} [Joa06] and LIBLINEAR [Lin08]. 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 running time, *SLR* is more than one order of magnitude faster than its opponents. Open source code for *SLR* is available on-line at: `http://www.mpi-inf.mpg.de/~ifrim/slr`. This work is presented in [IBW08].

Spam Filtering. The task of blocking unsolicited e-mail is of vital importance. Text classifiers trained on previously labeled e-mails can be used to automate this process. We show an application of our SLR algorithm to spam filtering. Empirical results provide insight into the importance of using variable-length n-grams when training spam filters.

Explicit Regularization and Newton-style extensions for SLR. We investigate several extensions of SLR. First, we look into integrating explicit regularization of the objective function into our learning algorithm. We propose new theoretical bounds and discuss the challenges such modifications pose. We also give a theoretical analysis of using Newton-style updates for the optimization inside SLR, and discuss some preliminary results with this approach.

1.4 Outline of the Thesis

Chapter 1 introduced the notion of text categorization and the various learning approaches designed to solve the problem of learning robust classifiers when labeled training data is sparse. Chapter 2 discusses relevant prior work, and positions our contributions in the context of existing learning approaches. Chapter 3 presents our generative learning model for inductive and transductive learning, and gives a detailed analysis of the various building blocks of the model, and their interactions and role in achieving high classification accuracy. We present experimental results comparing our techniques to state-of-the-art classifiers and analyze their behavior for small training datasets. Chapter 4 presents our novel algorithm for learning classifiers in the space of all n-gram features in the training set. The same chapter presents theoretical and empirical evidence that our algorithm achieves classification accuracy comparable to that of state-of-the-art classifiers, while being orders of magnitude faster than existing algorithms. Chapter 5 concludes with a discussion of our findings, and future work directions opened up by this dissertation.

Chapter 2

Related Work

2.1 Background Knowledge for Text Classification

2.1.1 Inductive Learning

Inductive learning paradigms assume that a manually labeled training set is given as input to the text classifier. In this setting a model is learned on the given training set, in order to provide good predictions on new samples. An open research problem in inductive learning for text classification concerns reducing the training set size required for achieving good classification results. Recent research efforts addressing this problem focused on the usage of more complex kernel functions for text classifiers such as SVM (string kernels, [LSST$^+$01]), designing more effective document similarity measures based on prior knowledge, e.g. by including semantic knowledge from WordNet or Wikipedia in the kernel function [BCM05, CBM06, WD08]), better representations of documents, e.g., feature engineering, [BH04a, SM99], or supplying training with background knowledge from the Web (domain ontologies, encyclopedia, e.g. Wikipedia) [IW06], [DLM$^+$06].

The work presented in [FC04] investigates the behavior of several state-of-the-art text classifiers (Naive Bayes, Logistic Regression and Support Vector Machines) for small training set sizes. The authors study the effect of several factors on classification quality: training set size, feature selection, the mismatch between the training and test distributions and different ratios of positives versus negatives in the training set. Their work extends the work presented in [WP03] which studies the behavior of decision tree classifiers for varying distributions of the training samples over classes. In [WP03] the authors show that accounting for the mismatch between the training and the test distribution, can considerably improve the classification results. The work in [FC04] moves a bit further by analyzing several other factors, such as the training set size and feature selection. One of their main findings is that when the training set contains a low percentage of positive samples as compared to negative samples (for the two-class case) in the training set, the simple Naive Bayes classifier outperforms SVM. The latter classifier is known for providing high accuracy classification results when training data is plentiful and the number of positives is reasonably high (a few hundreds). This work also shows that SVM is more sensitive than Naive Bayes to a mismatch in data distribution between the training and the test set. When the percent of positives is high enough, SVM is the dominant classifier. Therefore, for small training sets with a few representatives from the positive set, Naive Bayes seems to be a better choice, while for large enough training sets with enough positive samples, SVM is the dominant classifier. Several other papers [NJ01, KT06, RSNM03, BT04, CYM08, LBM06] also pointed out the dominance of generative classifiers over discriminative classifiers for small training sets. Recently, a lot of research work has focused on the mismatch between the training and the test distribution [BBS07, SKM07, PSM07, SS07] which negatively affects most classifiers.

In Chapter 3 of this thesis we present a generative model for learning with small training data, which uses external resources such as ontologies and encyclopedia, in order to better estimate its parameters [Ifr05, ITW05]. Somewhat related to our model is the work on Latent Semantic Indexing (LSI) [DDF+90], Probabilistic Latent Semantic Analysis (PLSA) [Hof01] and Latent Dirichlet Analysis (LDA) [BNJ03]. These techniques aim at capturing *synonymy* (words sharing the same meaning) and *polysemy* (the contextual change of meaning of certain words) by treating groups of co-occuring words as a coherent unit of meaning, or so called *concept*. The concepts in these models are mathematical notions, and can be interpreted as dimensions of the original documents and terms, in a new concept-space which captures the variations of word usage. Typically the number of concepts has to be provided a priori, i.e., the number of dimensions of the new space has to be fixed in advance. Recent work has focused on techniques for selecting the number of concepts automatically [BGJT04, TJBB06]. LSI, PLSA, LDA and the models extending these approaches are typically unsupervised learning techniques, i.e. statistical models which capture the characteristics of text, but do not have the goal of mapping text to a given set of categories, and do not require labeled samples. In contrast to this work, our Inductive Latent Model (ILM) technique aims at learning a mapping from the given set of labeled examples to the given set of categories. ILM selects its latent model structure based on named concepts from a given ontology, which are carefully selected to reflect the domain of the training data. This helps both in learning a robust structure of the probabilistic model, but also in the interpretation of results, since the individual concepts and their relation to word features and the given set of categories can be investigated for understanding the decisions of the classifier.

Prior work on text classification with small training sets has also focused on exploring concepts from given ontologies in order to enrich the bag-of-words feature space [BH04a, SM99, TSW03]. After the feature extension step, the extended representation is given as input to a standard supervised classifier, e.g. SVM. The typical outcome of such feature engineering approaches is that they rarely outperform the simple bag-of-words representation. More sophisticated approaches which involve boosting weak hypotheses obtained using PLSA techniques [CH03] show some promise for the direction of extending the word feature space with domain specific concepts. The work of [GM05] uses a combination of feature generation, feature selection, and domain adaptation from a large web directory to improve classification of diverse documents.

Besides feature engineering efforts, another direction of work has investigated the usage of background knowledge for improving text classification results when training data is small [ZH00, ZH01, ZH02, DLM+06]. The work of [ZH00, ZH01, ZH02] investigates various ways of using unlabeled data or other external background knowledge resources for improving classifiers learned from small training data alone. For example, [ZH00] investigates the usage of unlabeled data for improving short text classification. Rather than attempting to label the unlabeled data and then add these samples to the training data, the authors propose treating the unlabeled data as a bridge between training and test examples. If certain training examples are similar to unlabeled documents which are similar to some test documents, than these training documents should contribute more to predicting the category of those test samples. In some sense the unlabeled data plays the role of capturing co-occurrence of words in free flowing text from a similar domain as the training and test data. A similar idea is used in [ZH01, ZH02], where background knowledge, i.e. large amounts of information available about a specific problem on the World Wide Web, is used for assessing document similarity therefore achieving a bridging effect on the training and test samples. What is needed for this approach is an unlabeled corpus of related data in addition to the training data. As a concrete example of learning with background knowledge, the authors mention the following classification task. Suppose that we wish to classify the names of companies by the industry that the company is part of. A company such as *Watson Pharmaceuticals Inc* would be classified with the label *drug*, and the company name *Walmart* would be classified as type *retail*. Although we many not have numerous training examples, and the training examples are very short, we can find other data that is related to this task. Such data could be articles from the business section of an on-line newspaper or information from company home pages.

Similar ideas are investigated in the work of [DLM$^+$06]. They analyze the benefit of using background knowledge for learning informative prior distributions for the parameters of a logistic regression model. The hypothesis is that when training data is small, relying on prior parameter distributions learned from data related to the given training data can be beneficial for the classification quality. As examples of background knowledge the authors mention category descriptions meant for manual indexers, reference materials on the topics of interest, lists of features chosen by a domain expert, etc. The authors also make the point that integrating background knowledge by specifying prior distributions using a Bayesian framework should be more effective than directly adding it as training samples, due to the diverse and non-document-like forms of the domain knowledge texts. Their empirical results also support this hypothesis: treating domain texts as artificial training examples had an erratic impact, sometimes improving and sometimes substantially harming effectiveness. On the other hand, converting the domain texts to priors, almost always improved effectiveness.

2.1.2 Transductive Learning

In the *Transductive* learning paradigms, the data collection to be automatically labeled (i.e. test data) is available during training time. Some examples of real-life applications of transductive text classification vary from organizing book descriptions into pre-defined categories (e.g. by the popular Amazon portal), to classifying crawled Web pages into topic directories for more convenient access (e.g. by search engines Google or Yahoo!), or categorizing encyclopedia articles for better search and browsing (e.g. the largest on-line encyclopedia Wikipedia). Similar to inductive learning, the bottleneck for transductive classification stems from the difficulty of providing sufficient manually labeled data. Being able to harness the feature distributions and relations among the available unlabeled documents is thus an important asset to improve the classifier. Another potentially beneficial asset is additional knowledge about concepts, words and phrases that express concepts, and the semantic relations among concepts (e.g., hyponymy). Such knowledge sources may be given in the form of an ontology or thesaurus, such as WordNet [Fel99], Yago [SKW08], Wikipedia [Wik], etc.

There are many approaches to transductive learning. Transductive SVM (TSVM) were introduced by [Vap98] and first applied to text classification by [Ben99, Joa99b]. They exploit the structure in both training and test data for better positioning the maximum margin hyperplane. Essentially TSVM use the unlabeled data additional to the training data to find lower density regions in the document space. Assuming that lower density regions correspond to separation between categories, TSVM find the maximum margin hyperplane passing through such a low density region. The TSVM implementation presented in [Joa99b] suffers from the dependency on the true ratio of positives and negatives in the test set. Typically an estimate of this ratio is computed from the training set, but a big mismatch between the training estimate and the true ratio can lead to poor classification performance. Recent work [CWD03] tried to avoid this dependency, by modifying the learning algorithm such that the ratio of positives to negatives in the test set is not required explicitly. The authors coin this algorithm Progressive Transductive SVM (PTSVM) and show empirically that by avoiding the requirement for explicit knowledge on the positive/negative ratio gives promising results. Nevertheless, PTSVM has to pay a bit in complexity for avoiding the need for knowing the correct ratio, therefore when the number of unlabeled data is large, it may be even slower than TSVM during training.

Since TSVM requires solving a combinatorial optimization problem, extensive research efforts have been devoted to efficiently finding an approximate solution. Recent work presented in [XJZ$^+$08] surveys the existing approaches for efficiently solving TSVM and proposes a new algorithm which works via a convex relaxation, which converts the NP-hard problem to a semi-definite programming scheme. We first mention a few of the previous approaches and then give some details on the algorithm presented in [XJZ$^+$08]. The popular version of TSVM proposed in [Joa99b] uses a label-switching-retraining procedure to speed up the computation. In [CZ05], the hinge loss in TSVM is replaced by a smooth loss function, and a gradient descent method is used to find the decision boundary in a region of low density. Chapelle et al. [CCZ06] employ an iterative approach for TSVM. It begins with minimizing an easy convex objective function, and then gradually approximates the objective of TSVM with more complicated functions. The solution of the simple function is used as the initialization for the solution to the complicated function. Other iterative methods, such as deterministic annealing [SKC06] and the concave-convex procedure (CCCP) method [CSWB06], are also employed to solve the optimization problem related to TSVM. The main drawback of the approximation methods listed above is that they are susceptible to local optima, and therefore are sensitive to the initialization of solutions. To address this problem, in [CSK07], a branch and-bound search method is developed to find the exact solution. In [XS05], the authors approximate TSVM by a semi-definite programming problem, which leads to a relaxation solution to TSVM (noted as RTSVM), to avoid the solution of local optimum. However, both approaches suffer from high computational costs and can only be applied to small sized data sets.

Compared with other semi-definite programming relaxation approaches for Transductive SVM, the algorithm proposed in [XJZ$^+$08] is computationally more efficient reducing the number of free parameters from $O(n^2)$ to $O(n)$ where n is the number of examples. The key idea of the presented method is to approximate the non-convex optimization problem of TSVM by its dual problem. Nevertheless, the worst-case computational complexity of their algorithm is $O(n^{4.5})$, which is prohibitly high for reasonably large datasets. Very recent work [ZWZ08] further advances the understanding of efficient solutions for TSVM. The authors of [ZWZ08] mention their solution is inspired by the work of Joachims on Linear (Inductive) SVM presented in [Joa06]. They propose a cutting plane semi-supervised support vector machine (CutS3VM) algorithm, to solve the semi-supervised SVM (S3VM) problem. The authors construct a nested sequence of successively tighter relaxations of the original S3VM problem, and efficiently solve each optimization problem in this sequence by using a constrained concave-convex procedure (CCCP). The authors prove theoretically that their CutS3VM algorithm takes $O(sn)$ time to converge, where n is the total number of samples in the dataset and s is the average number of non-zero features, i.e. the sparsity.

Also relevant is the work of [ZO00], which shows theoretically and empirically that learning a maximum margin from the unlabeled data in order to assign the labels (i.e. what TSVM does) may be unreliable. The authors argue that in order for the unlabeled data to be useful for a transductive learner, the data model $p(x)$ must be parameter dependent. In the case of SVM, a strong parameter dependency of $p(x)$ is not necessary in order to work well in the supervised setting. Therefore, the authors suggest that the prior success of TSVM may have been due to the fortunate empirical setup in which there was indeed a reasonably large margin between in-class and out-of-class members. For example, if the data is generated from two heavily overlapping Gaussian, the decision boundary would go right through the densest region, and methods such as TSVM would perform badly [Zhu08]. In this sitation, generative models which have a parameter dependent data distribution may benefit more from the use of unlabeled data. Thus, it is very important to match the problem structure with the model assumptions [Zhu08].

Generative approaches can exploit the information in the unlabeled collection for better estimating the generating distribution. Generative models are perhaps one of the oldest semi-supervised learning method [Zhu08]. They assume a model $p(x, y) = p(y)p(x|y)$ where x are document samples, y are categories or classes, and $p(x|y)$ is an identifiable mixture distribution, for example Gaussian mixture models. With large amount of unlabeled data, the mixture components can be identified. One can think of the mixture components as soft clusters. Nigam et al. [NMTM00a] apply the EM algorithm on mixture of multinomials for the task of text classification using labeled and unlabeled data. They showed the resulting classifiers perform better than those trained using only labeled data. If the mixture model assumption is correct, unlabeled data is guaranteed to improve accuracy [Zhu08, CC95]. However if the model is wrong, unlabeled data may actually hurt accuracy. This has been observed by multiple researchers. Cozman et al. [CCC03] give a formal derivation on how this might happen. It is thus important to carefully construct the mixture model to reflect reality. For example in text categorization a topic may contain several sub-topics, and will be better modeled by multiple multinomials instead of a single one ([NMTM00b]). Another solution is to down-weight unlabeled data ([CJ01]), which is also used by Nigam et al. [Nig01]. Even if the mixture model assumption is correct, in practice mixture components are identified by the Expectation-Maximization (EM) algorithm ([DLR77]). EM is prone to local maxima. If a local maximum is far from the global maximum, unlabeled data may again hurt learning. Remedies include smart choice of starting points by using informative priors [IW06] or by active learning ([Nig01]).

Other semi-supervised learning approaches represent the dataset as a graph and exploit the structure of the graph in search for mincuts [BC01] or for min average cuts [Joa03]. Graph-based semi-supervised methods define a graph where the nodes are labeled and unlabeled examples in the dataset, and edges (potentially weighted) reflect the similarity of examples. These methods usually assume label smoothness over the graph. Graph methods are typically nonparametric, discriminative, and transductive in nature [Zhu08]. Next, we give some examples of graph-based learning methods for text classification. Blum and Chawla ([BC01, BLRR04]) pose semi-supervised learning as a graph mincut problem. In the binary case, positive labels act as sources and negative labels act as sinks. The objective is to find a minimum set of edges whose removal blocks all flow from the sources to the sinks. The nodes connecting to the sources are then labeled positive, and those to the sinks are labeled negative. Pang and Lee [PL04] used graph mincuts to improve the classification of sentences into either 'objective' or 'subjective', with the assumption that sentences close to each other tend to have the same class. Discrete Markov Random Fields also called Boltzmann Machines are another approach to graph-based semi-supervised learning. Nevertheless, the inference problem posed in this framework is inherently difficult [ZG02, GSD06]. The Gaussian random fields and harmonic function methods introduced in [ZGL03] are a continuous relaxation to discrete Markov random fields and are known as label propagation techniques. Niu et al. [NJT05] applied the label propagation algorithm (aka harmonic functions) to word sense disambiguation. Goldberg and Zhu [GZ06] applied the algorithm to sentiment analysis for movie rating prediction. The Spectral Graph Transducer (SGT) [Joa03] is another example of graph-based learning methods for transductive classification. The SGT algorithm solves a normalized-cut (or ratio-cut) problem with additional constraints for the labeled examples using spectral methods. Pham et al. [PNL05] perform empirical experiments on word sense disambiguation, comparing variants of co-training [BM98] (training two separate classifiers with the labeled data, on two independent sub-feature sets) and spectral graph transducer. The authors notice that the spectral graph transducer with carefully constructed graphs produces good results.

Szummer and Jaakkola [SJ01] perform a t-step Markov random walk on the graph of labeled and unlabeled examples. The influence of one example to another example is proportional to how easy the random walk goes from one to the other. The graph construction is very important for all the graph-based methods previously mentioned. Several approaches have been investigated in the literature [Zhu08], e.g. using domain knowledge, using nearest neighbor graphs, etc.

Explicit knowledge sources like ontologies, thesauri, or dictionaries have been used in prior work on text classification mostly for feature engineering, e.g., constructing composite words or phrases as features based on a thesaurus or for constructing document similarity graphs. Most notably, the WordNet thesaurus [Fel99] has been leveraged in various feature engineering approaches [BH04b], [SM99]. In contrast, we propose integrating explicit knowledge about word-concept relationships into the learning procedure itself.

In Chapter 3 of this thesis we present a generative model for transductive learning, coined Transductive Latent Model (TLM) [IW06]. Our model relies on both available unlabeled data and external resources in order to learn the latent structure of the model, and for estimating parameters. In particular, unlabeled data plays an important role in estimating word-concept dependencies in our model, where individual words get mapped onto concepts that play the role of latent variables. Since capturing the contextual meaning of words (i.e. the word-concept mappings) only requires unlabeled data, we can take advantage of large quantities of unlabeled data to better estimate such dependencies, and therefore better estimate the model parameters. We show experiments comparing TLM to the state-of-the-art transductive classifiers Transductive Support Vector Machines [Joa99b] and Spectral Graph Transducer [Joa03].

Topic-driven clustering

Clustering algorithms typically find the structure in a given dataset, in terms of tight groups of similar documents. The clusters found by such algorithms do not always correspond to the user needs, e.g. the user may desire to specify what type of grouping he or she is interested in and even specify a few keywords describing each of the groups or categories he or she would like to get from the clustering algorithm. This setting falls to some degree into the group of semi-supervised clustering approaches [Zhu08], but the problem such techniques may face is the extreme sparseness of explicitly labeled examples, since the training data in this framework is just a sparse keyword description of the categories of interest, rather than several labeled training documents. The framework was coined *topic-driven clustering* in [ZK05]. This problem has been investigated by several researchers going back to 1999 [MN99]. We summarize some of these approaches in this section.

[ZK05] takes a clustering (as opposed to classification) view on the problem, and gives three clustering schemes which use the topic descriptions as prior knowledge for adjusting unsupervised clustering methods. They focus on a few clustering algorithms, among which a k-means style one. We discuss their methods and give an intuition using the k-means style algorithm.

In their work, both documents and topic descriptions are represented as vectors in a vector space. One of the approaches pursued by the authors is to seed the clustering algorithm with the topic descriptions and iteratively compute topic-weighted centroids, which are biased toward the topic descriptions.

Another technique the authors propose is to optimize a combined objective function, e.g. a simple supervised clustering, which maps documents onto topic descriptions by cosine similarity, with an unsupervised k-means solution, by means of a weight which settles the influence of the two methods $(\alpha f_{supervised} + (1 - \alpha) f_{unsupervised})$. The authors show their proposed techniques are superior to simple seeded k-means, but their evaluation methodology is problematic since the authors use the entropy of clustering as their evaluation measure. For this particular problem in which each cluster has an associated topic, using entropy to measure the quality of the clustering solution is not appropriate. The entropy captures how homogeneous the output clusters are, but does not capture the number of misclassifications corresponding to the clustering solution. Lets assume we are interested in two given topics and we obtain two clusters as a result from some clustering algorithm. The entropy is low (suggesting high quality of the clustering) in either of the two cases of 100% accuracy (the documents in each cluster are representatives of the corresponding topic) or 0% accuracy (the documents are entirely swaped, i.e. we find two clsuters but they correspond to the wrong topics), thus using entropy for measuring the quality of the clustering can be misleading. We therefore re-ran experiments with some of the techniques proposed in the original paper, in order to evaluate the quality of their clustering approaches using the standard F1 measure for evaluating classification solutions.

Previous approaches to topic-driven clustering mainly focused on bootstrapping training samples based on the keyword topic descriptions, and then applying standard classification techniques (e.g. Naive Bayes, SVM) for obtaining the desired clustering solution. Such examples include [MN99] which automatically labels some documents based on the presence of the topic keywords, and then uses a Naive Bayes classifier and an EM algorithm which takes advantage of the unlabeled documents for learning a classifier of the entire collection. In [GSD05], an unsupervised method is proposed for better selecting the initial labeled samples. The approach is based on LSI decomposition coupled with Gaussian mixtures for learning a document-topic similarity distribution, per topic. [LLLY04] addresses the issue of selecting good keywords for describing the topics of interest. They first cluster the collection and then use Mutual Information [MN98a] for selecting candidate keywords for each cluster. After this step a user selects good keywords for describing each cluster/topic of interest.

In contrast to previous approaches, we view this problem as a transductive classification problem, where we assume that each topic/category has a short textual description which is treated as explicitly labeled training data. The goal is to organize the unlabeled collection according to the user-defined topics. We employ our TLM model for learning from both the supervised and the unsupervised information available. We show that taking a transductive learning approach to this problem shows promising results. We also experiment with other inductive and transductive classifiers, and comment on the behavior of each of the compared techniques for solving the topic-driven clustering problem.

2.2 Rich Document Representations for Text Classification

2.2.1 N-gram Features, Parse Trees, Semantic Kernels

A lot of research has investigated the hypothesis that using document representations which are richer than the simple bag-of-words, (e.g. syntactic and semantic representations) can improve text classification results. Most of this work stems from the observation that words alone do not always represent true atomic units of meaning [SM99]. David D. Lewis undertook a major study of the use of noun phrases for statistical classification as part of his Ph.D. thesis [Lew92], but concluded that more sophisticated phrase-based representations did not improve the results obtained using the bag-of-words representation. In [SM99] the authors attempt to integrate more complex information in the RIPPER rule-based learner. They focus on using some alternative ways to represent text based on syntactic and semantic relationships between words (i.e. phrases, synonyms and hypernyms from Wordnet). Their finding is that using this additional information directly by extending the set of features does not improve results, but combining classifiers based on different representations using a majority voting technique seems more promising. Nevertheless, their study separates feature selection from the actual learning algorithm. Additionally, since using all the phrases or possible feature combinations leads to feature space explosion, the authors select only a subset of features, therefore potentially eliminating some good combinations of features. The work of [JM00] expands the previous study, by analyzing careful combinations of various conceptual and contextual features (e.g. synonyms, hypernyms, term frequency, and bigrams of nouns, synonyms and hypernyms) and various classifiers (Coordinate matching, TF*IDF, and Naive Bayes). The authors mention that when using an automated scheme to determine which features to use, based upon the data set in question, the results improve considerably as compared to using just the bag-of-words representation. They also give some guidelines for feature combinations and type of features which typically lead to improvement of results. According to the authors, hypernyms always improve results and they usually improve over the use of synonyms alone; stemming helps when a large percentage of terms do not exist in WordNet; bigrams can help dramatically, but the corresponding individual terms should be removed from the feature set, etc. Many other works rely on thesauri such as WordNet or Wikipedia for extending the word feature space with concepts [BH04a, GM05, GM06a, GM07, WHZ$^+$07, WHZC08]. The main idea in [WHZ$^+$07] for example, is to use Wikipedia to build a general thesaurus for word feature enrichment. Based on the filtered Wikipedia concept index, they first search candidate concepts mentioned in each text document, and then add the synonyms, polysems, hyponyms, etc., of these candidate concepts to the original documents. On a similar note [GM07] enhances the feature space with features generated from domain-specific and common-sense knowledge. This knowledge is represented by ontologies that contain hundreds of thousands of concepts (e.g. Open Directory Project [Dmo]), further enriched through controlled Web crawling. Prior to text categorization, a feature generator analyzes the documents and maps them onto appropriate ontology concepts that augment the bag of words used in simple supervised learning.

Other studies focus on using more sophisticated syntactic or semantic information for representing documents. The work of [BMP00, Mos03] analyzes various avenues for exploiting syntactic information for text classification. In his Ph.D. thesis Moschitti [Mos03] refers to the following necessary pre-processing steps. First, the author points at the importance of the separation between content words (i.e. open syntactic classes such as nouns, verbs and adjectives) and other less relevant information (e.g., functional classes like prepositions or complex functional expressions *as far as* or *in order to*). The need for this separation was known since the early research in Information Retrieval [Sal89] which motivated the use of stoplists or stop-words. Another important aspect is the identification of the syntactic role of each word in its corresponding context: for example verbal from nominal uses of a lemma can be distinguished (*ready to land* vs. *suitable public lands*). The syntactic role allows to select the more informative class of words, i.e. nouns, and to perform a first level of word disambiguation, e.g., *book* and *to book*. The syntactic category of the word *book*, clearly, decides which is the most suitable choice between categories like *Book Sales* and *Travel Agency*. Furthermore, the author points at the identification of linguistically motivated structures such as complex Proper Nouns (e.g., *Shell Transport & Trading Co. PLC*), as they should not be modeled similarly to common nouns.

Further work uses more complex information such as syntactic parse trees of individual sentences or document paragraphs as features. In [Mos06] the authors study the use of tree kernels to encode syntactic parsing information for semantic role labeling and question answering. Their experiments with SVM on the PropBank and FrameNet predicate argument structures show that kernels defined using syntactic information are generally more accurate. The work presented in [BM07] proposes a generalized framework consisting of a family of kernels that jointly incorporates syntax and semantics. This is done by considering information about the syntactic structure of the input and by incorporating knowledge about the semantic similarity of term features. The applications considered by the authors are the classification of natural language questions from a TREC question answering dataset and the automated assignment of ICD-9 (International Classification of Diseases, Version 9) categories to short textual fragments of medical diagnoses.

2.2.2 Learning with Variable-length N-grams

Previous research has shown that for most topical text classification applications (e.g. categorizing news into categories such as Politics, Sports, Science), the bag-of-words representation suffices for capturing the characteristics of each category. This is possible because there is typically a high correlation between the category profile and individual keywords in documents [KM04].

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 classifier itself is very powerful [HF95, KNS97, LM02, PSW04, ZL06]. 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 natural-language parse trees. Using syntactic or semantic kernels typically comes at a price in computational complexity, since computing more complex similarity measures usually results in considerable running time costs. Kudo et al. [KM04] observed that the classification 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.

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., [GLM06, Joa06, ZO01]), 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 [CYHH07]. 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., [KB06]), but this in turn puts the burden on the users by requiring a potentially large amount of human attention.

Recent advances in efficient, regularized learning algorithms, such as SVM [Joa06, HCL$^+$08] and sparse logistic regression [GLM06] have reduced the need for explicitly modifying the input feature space (e.g. by doing feature selection), 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 [FCW00, Ros00], which are generative approaches, or SVM with string kernel [SS02], a discriminative approach.
Markov chain models can be in fixed order/memory or variable order/memory [MS00, ZL06]. The Markov chain models in fixed order n are usually called n-gram language models [Goo01, Ros00]. Recently, [PSW04] 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 [PSW04, ZL06]. 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) [DSSS04], text compression [BCW90] methods such as PPM (prediction by partial matching) and PPM* [CT97] 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 [DPHS98, Joa98, YL99, ZL06]. 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 [LSST$^+$01, ZL06]. Recent work [ZL06] 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 [GLM06]) 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 [NW06]. 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 [NW06], conjugate gradient [NW06], and hybrids of conjugate gradient with other methods [KM03], 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. [EHJT04] 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 [NW06], 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 [JYZH03], and cyclic coordinate descent [SK03, ZO01]. Some of these algorithms have already shown promise on text categorization or other language processing tasks. One of the methods we use for comparison, Bayesian Logistic Regression (BBR) [GLM06], is an efficient implementation of regularized cyclic coordinate descent logistic regression.

In Chapter 4 we present our technique for efficiently learning a logistic regression classifier in the space of all n-grams present in the training set. 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) [GLM06] and SVM^{perf} [Joa06]. 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.

Chapter 3

Background Knowledge for Text Classification and Clustering

3.1 Introduction

The process of gathering clean and representative labeled data for text classification is typically very slow and costly. Therefore, techniques that can learn from a small set of labeled data and rich sources of background knowledge can be highly useful for many different applications. For example, if we are interested in automatically organizing the technical literature (conference papers or other type of documents) on our computers into pre-defined categories, e.g. *text classification, information extraction, information retrieval, etc.* and we are willing to label a few examples that (sparsely) describe the categories of interest, we can learn a classifier from the provided examples, and use it to label the remaining collection of documents. Learning a classifier on such small training data is likely to provide very low accuracy when automatically predicting the labels of new documents, since many features (terms) may not even be represented in the small training set. We could instead use other sources of information relevant to this particular learning task in order to improve the accuracy of the classifier. Such examples of *background knowledge* could be:

- Ontologies of concepts, such as WordNet [Fel99], Yago [SKW08], etc., from which we can learn about word-concept or concept-concept relations, phrases that express concepts.
- Enclyclopedia, such as Wikipedia, which provide broad descriptions of the categories of interest.
- Unlabeled documents, which give information about the structure of the input collection of documents, and capture knowledge about the feature distributions, etc.

This type of background knowledge can be used for solving two difficult problems typically associated with learning parametric statistical models: learning an appropriate structure for the statistical model employed and learning a prior distribution for the model parameters. In Section 3.2 we present a latent model which builds on different types of background knowledge for solving these type of problems. The structure of the model (i.e. the number of latent variables and their initialization) is decided based on the training set and an external ontology of concepts. An informative prior for the model parameters is learned from the training corpus and other available resources, in order to improve the parameter estimation based on the (sparse) training set alone. In particular, we analyze the behavior of this model for two different learning paradigms, *inductive* and *transductive* learning. We conclude with an overview of the proposed approach and discuss future research directions.

3.2 Latent Model

3.2.1 Introduction

For text classification, the standard representation of documents is a set or bag-of-words, i.e. a word vector, also called *feature vector*. Features can be all the distinct terms in the input collection, or a subset or combinations of the terms. More sophisticated pre-processing at both syntactic and semantic level can be done, e.g. using part-of-speech tagging or word sense disambiguation. Typically some light term pre-processing is preferred, such as stemming, stop-words removal, etc. The most widely used features are simply words or their morphological normal forms. Here, we use the name *features* and *words* interchangeably.

A simple classifier, such as Naive Bayes, learns frequency statistics associated with each feature and topic, from the training set. Even if this type of classifier makes a strong assumption about the input documents, i.e. it assumes that given the topic label, the features of the document are independent, it performs surprisingly well, many times being competitive to much more sophisticated classifiers such as Support Vector Machines [RSTK03].

The independence assumption of Naive Bayes does not badly affect the final classification predictions [RSTK03, MRS08] since the classifier manages to estimate well the most likely topic, even if the probability estimate may be wrong. In other words, as long as Naive Bayes correctly predicts the ranking of topics with respect to a given document, having wrong estimates for the actual probabilities of topics for the given document does not influence the final classification accuracy.

Nevertheless, since Naive Bayes relies on simply counting words in the training set to estimate the probability of a topic given a word, for small training sets the count estimates may be unreliable, which has a direct effect on the final predictions.

For avoiding these type of effects that can affect Naive Bayes, we can instead try to capture more information about the training set, when estimating parameters. For example, we can capture groups of related words, as together describing a concept, while groups of concepts together describe a topic. This essentially breaks the problem into three rather than two layers: a layer of *words*, a layer of *concepts* and a layer of *topics*. Groups of words from the word layer can be understood as ways to describe a common concept, while groups of concepts from the concept layer can be understood as finer grained topics, which together describe a more general topic. This has the advantage of capturing *synonymy* effects, i.e., words that have the same meaning (synonyms) and *correlation* effects, i.e., words that tend to co-occur. Furthermore, it can deal with *polysemy* effects, i.e., words that have various meanings or interpretations depending on the context.

For example, if a text describes a topic using different words, the frequency of each word may not be enough in itself to correctly discriminate between topics of a given document which contains some of these words, but if we use the implicit grouping of related words, the frequency estimates get in some sense focused into one group parameter, and thus the predictive power of the group can improve the predictions.

For getting the relations between the word, concept and topic layers, we first need to instantiate each of the layers and then need to learn the groupings in a meaningful way. The words and topics are provided by the training set, while for instantiating the concepts we use an external ontology.

For the scope of this work, we specifically used the WordNet ontology [Fel99], which at the time contained around 150,000 concepts. WordNet was for a long time the most prominent and freely available such resource and is simply an example of the explicit concept collections that could be leveraged for better text representation. Currently, there is a lot of work on automatically building ontologies by mining concepts from encyclopedia like Wikipedia or directly from unstructured text from the Web [SKW08, WHW08, Sar08, SS04].

In order to capture the dependencies between the three layers, we develop a generative model for text documents, which postulates that words are generated by concepts which in turn are generated by topics. This simulates the intuition that topics are describes by finer grained concepts which in turn are described by several words. In order to estimate the word-concept and concept-topic dependencies (we also refer to them as mappings), we rely on an iterative EM (expectation-maximization) procedure for maximum likelihood estimation from the given training set. Additionally, for helping the parameter estimation, we use several background knowledge resources to infer a priori word-concept and concept-topic mappings. For the a priori estimation, we use context based similarity heuristics which rely on:

- labeled (training set) and unlabeled documents (if the test set is available a priori or related collections of unlabeled documents) for estimating a context for the usage of words
- ontology information in the form of concept descriptions and neighborhoods, to infer a context describing concepts
- encyclopedia (e.g. Wikipedia) information, to infer broad descriptions of the topics of interest

The a priori knowledge coming from the background resources plays a very important role in making the learning and estimation process robust and practically viable, by removing unlikely combinations of word-concept and concept-topic pairs.

In the next subsection we describe the generative model and the estimation process in detail. In particular, we discuss the building blocks of the model, and the modeling choices we have made. In Section 3.3 and Section 3.4 we analyze the effect of different building blocks for inductive and transductive learning. We conclude this chapter with a discussion of experimental results for the two different learning paradigms.

3.2.2 Generative Model

In this section we introduce the framework and the theoretical model proposed. Depending on the learning strategy used for estimating parameters we coin our model **Inductive Latent Model (ILM)** or **Transductive Latent Model (TLM)**.

We consider as input:

- A set of pre-defined categories or *topics* $T = \{t_1, \ldots, t_k\}$.
- A *document collection*, $D = \{d_1, \ldots, d_n\}$, which is split into training (documents with known topics) and test data (documents with unknown topics).
- A set of *features*, $F = \{f_1, \ldots, f_m\}$, that can be observed in documents (*words or phrases*).
- An *ontology graph of concepts*, $C = \{c_1, \ldots, c_r\}$, where each concept has a short textual description, and is related to other concepts by semantic edges (e.g. hypernym/hyponym relations).

Given a document d with observed features (and possibly unknown topics), we want to automatically predict $P[t|d]$ for every topic t or find $argmax_t P[t|d]$. Let (f, t) be observation pairs from the training set. Our generative model for feature-topic co-occurrence can be described as:

1. Select a topic t with probability $P[t]$;
2. Pick a latent variable c with probability $P[c|t]$, the probability that concept c describes topic t;
3. Generate a feature f with probability $P[f|c]$, the probability that feature f describes concept c.

The pairs (f, t) can be directly observed, while concepts are implicit and are treated here as latent variables. Figure 3.1 shows a graphical representation of our generative model. This is a latent

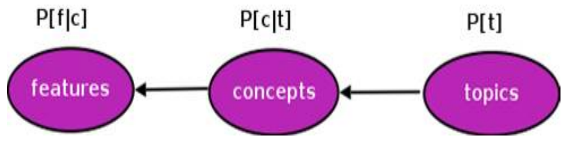

Figure 3.1: Graphical model representation of the generative model.

variable model for co-occurrence data which associates an unobserved variable $c \in \{c_1 \ldots c_r\}$ with each observation pair (f, t).

3.2.3 Learning Model Parameters

For tractability reasons, the model is based on two independence assumptions. The different observation pairs (f, t) are generated independently and the features f and topics t are conditionally independent given the latent variable c: $P[(f, t)|c] = P[f|c] \cdot P[t|c]$. To describe the generative process of an observation (f, t) we sum up over all the possible values that the latent variables might take

$$P[f, t] = \sum_c P[c] \cdot P[(f, t)|c]. \tag{3.1}$$

This representation is also called *mixture distribution*, since each concept c gives a probability of generating the pair (f,t) from a particular distribution $P[(f,t)|c]$.

Since we are working with count data, we naturally assume the (f,t) pairs are generated from a multinomial distribution. The likelihood of the observed (f,t) samples can therefore be expressed as:

$$\begin{aligned} L &= \prod_{f\in F, t\in T} P[f,t]^{n(f,t)} \\ &= \prod_{f\in F, t\in T} (\sum_{c\in C} P[c]\cdot P[(f,t)|c])^{n(f,t)} \\ &= \prod_{f\in F, t\in T} (\sum_{c\in C} P[f|c]\cdot P[c|t]\cdot P[t])^{n(f,t)} \end{aligned} \tag{3.2}$$

$n(f,t)$ is the number of occurrences of feature f in the training set of topic t.
The learning problem can be formulated as estimating the parameters ($P[f|c]$, $P[c|t]$, $P[t]$) of the generating distribution, given the observed (f,t) samples from the training set. This can be formally expressed as a maximization of the *observed data log-likelihood*:

$$\begin{aligned} l &= \sum_{(f,t)} n(f,t)\cdot log(P[f,t]) \\ &= \sum_{(f,t)} n(f,t)\cdot log(\sum_{c} P[c]\cdot P[(f,t)|c]) \end{aligned} \tag{3.3}$$

Due to the sum inside the logarithm in Equation 3.3, direct maximization of the log-likelihood by partial derivatives is infeasible. But if we knew for each pair (f,t) exactly by which of the latent variables it was generated, we could express the *complete log-likelihood* without a log of sums, because only one term inside the sum would be non-zero. We introduce indicator variables

$$\Delta_{c,f,t} = \begin{cases} 1 & \text{if the pair } (f,t) \text{ was generated by concept } c \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

The joint density becomes:

$$P[f,t,\Delta_{c,f,t}] = \prod_{c} (P[c]\cdot P[f,t|c])^{\Delta_{c,f,t}} \tag{3.5}$$

and the *complete data likelihood* is:

$$L^{comp} = \Pi_{f,t} P[f,t,\Delta_{c,f,t}]^{n(f,t)}. \tag{3.6}$$

Essentially $P[f=1, t=1, \Delta_{c,f,t}=1]$ is the probability of a certain (feature, topic) pair to have been generated from a given concept. Now we can express the complete log-likelihood of the data by:

$$l^{comp} = \sum_{(f,t)} n(f,t)\cdot log(P[f,t,\Delta_{c,f,t}]) \tag{3.7}$$

$$l^{comp} = \sum_{(f,t)} n(f,t) \cdot log(\prod_{c} (P[c] \cdot P[f,t|c])^{\Delta_{c,f,t}}) \tag{3.8}$$

$$l^{comp} = \sum_{(f,t)} n(f,t) \sum_{c \in C} \Delta_{c,f,t} \cdot log(P[c] \cdot P[f,t|c]) \tag{3.9}$$

If we replace the missing variables $\Delta_{c,f,t}$ by their expected values, using Jensen's inequality (e.g. $E[log(X)] \geq log(E[X])$) we can show that the complete data log-likelihood in Equation (3.9), bounds from below the *incomplete log-likelihood*, Equation (3.3). This means that we can concentrate on the easier problem, that of maximizing the *expected complete data log-likelihood* (where the expectation is taken over the unknown variables $\Delta_{c,f,t}$). We compute the expectation over the missing variables using the current values of the model parameters:

$$P[\Delta_{c,f,t}|f,t] = \frac{P[f,t,\Delta_{c,f,t}]}{P[f,t]} = \frac{\Pi_c (P[c] \cdot P[f,t|c])^{\Delta_{c,f,t}}}{\sum_c P[c] \cdot P[f,t|c]} \tag{3.10}$$

$$\begin{aligned} E[\Delta_{c,f,t}|f,t] &= 1 \cdot P(\Delta_{c,f,t} = 1|f,t) + 0 \cdot P(\Delta_{c,f,t} = 0|f,t) && (3.11) \\ &= P(\Delta_{c,f,t} = 1|f,t) + 0 \cdot P(\Delta_{c,f,t} = 0|f,t) \\ &= \frac{P[c] \cdot P[f,t|c]}{\sum_{c \in C} P[c] \cdot P[f,t|c]} = P[c|f,t] \\ & && (3.12) \end{aligned}$$

$$E[l^{comp}] = \sum_{t \in T} \sum_{f \in F} n(f,t) \sum_{c \in C} P[c|f,t] \cdot log(P[c] \cdot P[f,t|c]) \tag{3.13}$$

$$E[l^{comp}] = \sum_{t \in T} \sum_{f \in F} n(f,t) \sum_{c \in C} P[c|f,t] \cdot log(P[c] \cdot P[f|c] \cdot P[t|c]) \tag{3.14}$$

$$E[l^{comp}] = \sum_{t \in T} \sum_{f \in F} n(f,t) \sum_{c \in C} P[c|f,t] \cdot log(P[t] \cdot P[f|c] \cdot P[c|t]) \tag{3.15}$$

This type of approach in which the maximization of the likelihood is analytically intractable, but made easier by enlarging the sample with latent data falls in the category of *Expectation-Maximization* (EM) techniques [DLR77].

Typically EM algorithms take two iterative steps:

- **E-Step:** Expectation step, in which posterior probabilities are estimated for the latent variables, taking as evidence the observed data (current estimates of the model parameters). For calculating the probabilities of the E-step, we use Bayes' formula:

$$\begin{aligned} P[c|f,t] &= \frac{P[c,f,t]}{P[f,t]} && (3.16)\\ &= \frac{P[f,t|c]\cdot P[c]}{\sum_{c'\in C} P[f,t|c]\cdot P[c]} = \frac{P[f|c]\cdot P[t|c]\cdot P[c]}{\sum_{c'\in C} P[f|c]\cdot P[t|c]\cdot P[c]}\\ &= \frac{P[f|c]\cdot P[c|t]\cdot P[t]}{\sum_{c'\in C} P[f|c]\cdot P[c|t]\cdot P[t]}\\ P[c|f,t] &= \frac{P[f|c]\cdot P[c|t]}{\sum_{c'\in C} P[f|c]\cdot P[c|t]} && (3.17) \end{aligned}$$

- **M-Step:** Maximization step, in which the current parameters are updated based on the expected complete data log-likelihood which depends on the posterior probabilities estimated in the E-Step. We compute the parameters $P[f|c], P[c|t]$ and $P[t]$ that maximize the expected complete data log-likelihood $E[l^{comp}]$.

$$E[l^{comp}] = \sum_{t\in T}\sum_{f\in F} n(f,t) \sum_{c\in C} P[c|f,t]\cdot log(P[t]\cdot P[f|c]\cdot P[c|t]) \quad (3.18)$$

For the maximization criterion, we need to also consider the *normalization constraints*:

$$\sum_{t\in T} P[t] = 1;\ \sum_{f\in F} P[f|c] = 1,\ for\ each\ c\in C;\ \sum_{c\in C} P[c|t] = 1,\ for\ each\ t\in T \quad (3.19)$$

The M-step parameter estimates are:

$$P[f|c] = \frac{\sum_{t\in T} n(f,t)P[c|(f,t)]}{\sum_{f'\in F}\sum_{t'\in T} n(f,t)P[c|(f,t)]} \quad (3.20)$$

$$P[c|t] = \frac{\sum_{f\in F} n(f,t)P[c|(f,t)]}{\sum_{c'\in C}\sum_{f'\in F} n(f,t)P[c|(f,t)]} \quad (3.21)$$

$$P[t] = \frac{\sum_{f\in F, c\in C} n(f,t)P[c|(f,t)]}{\sum_{t'\in T}\sum_{f'\in F, c'\in C} n(f,t)P[c|(f,t)]} \quad (3.22)$$

The E-step and M-step are iterated until some stopping criterion is met. The number of EM iterations is a parameter of the model. Typically, a threshold on the log-likelihood improvement (change in the function given in Equation 3.3) is used as a stopping criterion.

The algorithm presented above estimates the parameters $P[f|c], P[c|t]$ and $P[t]$ based on the postulated model. Our goal is nevertheless to predict the $P[t|d]$ for a given document d. For this we use Bayes formula to reverse the generative model and predict a distribution over topics for a given document. We first use the estimated parameters to compute $P[d|t]]$.

$$\begin{aligned} P[d|t] &= \prod_{f\in d} P[f|t]^{n(f,t)} = \prod_{f\in d}\left(\frac{P[f,t]}{P[t]}\right)^{n(f,t)} && (3.23)\\ &= \prod_{f\in d}(\sum_{c\in C} P[f|c]\cdot P[c|t])^{n(f,t)} \end{aligned}$$

We can than substitute 3.23 into 3.24 to compute $P[t|d]$.

$$P[t|d] = \frac{P[d|t] \cdot P[t]}{P[d]} = \frac{P[d|t] \cdot P[t]}{\sum_t P[d|t] \cdot P[t]} \tag{3.24}$$

3.2.4 Problems and Solutions

The EM algorithm faces two major problems:

- The combinatorial explosion of the parameter space, since the number of parameters is directly proportional to the cross-product of the number of features, concepts and topics. These parameters are sparsely represented in the observed training data.
- The possibility of converging to a local maximum of the likelihood function (i.e. not finding the global maximum).

For the first problem, it is desirable to **prune the parameter space** to reflect only the meaningful parameter combinations. For the second problem, it is desirable to **pre-initialize the model parameters** to good values.

Pruning the Parameter Space

We propose using *feature selection* for reducing the size of the feature space down to a subset of important features and using *concept selection* for reducing the size of the concept set selected from the ontology, down to a subset relevant to the given input collection.

1. Feature Selection

   From the given collection we extract a set of features using two approaches. *Supervised feature selection* from the labeled training set retains the features with the highest average Mutual Information with the topic variable [MN98a]. *Unsupervised feature selection* from the entire collection of documents (without use of topic labels), selects features with high $tf \cdot idf$ [Cha03, Kra05] weights.

2. Concept Set Selection

   We use the WordNet thesaurus [Fel99] as the basis for our ontology graph. WordNet contains around 150,000 concepts (word senses) linked by hierarchical relations. Using the entire set of concepts can result in a high computational overhead and a high amount of noise. A better approach is to select from the ontology only a subset of concepts that reflects the semantics of the input collection. We call this the *candidate set of concepts*. This set is selected in a pre-processing step, before running the EM algorithm, and serves the purpose of instantiating the layer of latent variables in our model.

   There are several strategies for selecting the concept set, based on context similarity measures between the concepts from the ontology and the features from the input text collection. These strategies are highly related to the pre-initialization of the model parameters, and thus we give more details on this in the next paragraph. The advantage of such a concept selection step is that the size of this subset is some orders of magnitude lower than the size of the entire ontology. It also pre-determines the number of latent variables of the model.

Pre-initializing the Model Parameters

EM is theoretically guaranteed to find a local maximum of the objective function. Therefore, the standard way of using this algorithm is to randomly initialize the model parameters and iterate the algorithm until convergence. This is done repeatedly and the values of the parameters that give the highest value of the likelihood are retained.

For avoiding slow or suboptimal convergence of the EM algorithm, it is desirable to **pre-initialize the model parameters** to good values.

In order to get a good initialization for the parameters $P[f|c]$ and $P[c|t]$ we use a similarity-based mapping approach. The technique consists of mapping features to concepts and concepts to topics, based on *context similarity.* Let f be a feature (word) that we want to map to the ontological concepts. For each occurrence of word f in the text collection, its local context is a text window around its offset. We can gather all these local contexts into a global context over the entire collection. We note that the context of the feature does not require labeled data (supervised information), but can be computed on the entire (possibly unlabeled) collection.

For gathering concept contexts we rely on the ontology graph. The WordNet thesaurus is a directed acyclic graph (DAG) where the nodes are the concepts and the edges are semantic relationships [Fel99]. First, we query WordNet for the possible meanings of word f. For example, if we query WordNet for the word *mouse* we get:

- *The* **noun** *mouse has 2 senses (concepts) in WordNet.*

  1. *mouse – (any of numerous small rodents...)*
  2. *mouse, computer mouse – (a hand-operated electronic device...)*

- *The* **verb** *mouse has 2 senses (concepts) in WordNet.*

  1. *sneak, mouse, creep, steal, pussyfoot – (to go stealthily or furtively)*
  2. *mouse – (manipulate the mouse of a computer)*

Let $\{c_1, \ldots, c_l\}$ be the set of meanings associated with f. By taking also the synonyms of these word senses, we can form *synsets* for each of the word meanings. Additionally, for each sense c_i we use the neighborhood formed by its hypernyms, hyponyms, holonyms and their short textual descriptions to form the context. We restrict the neighborhood to depth two, in order to avoid introducing too much noise.

Next, we apply a light word sense disambiguation step by computing the overlap between the context of the feature and that of each of its possible meanings. This type of approach is commonly used in the word sense disambiguation literature.

For each of the candidate senses c_i, we compute the cosine similarity between the tf vectors of $context(f)$ and $context(c_i)$, $i \in \{1, \ldots, l\}$.

There are several strategies for deciding which concepts should be selected for the final set. One strategy keeps all the concepts gathered by querying the ontology for each feature. The size of this set is typically several times the size of the feature set depending on how many different meanings a feature maps to in the ontology.

A second strategy, only keeps the most dominant concept (meaning) of f. For features having multiple meanings, our hypothesis is that "secondary" meanings are introduced by their corresponding features, e.g., for the feature *Java* having meanings *island* and *coffee*, if *island* is selected as the main meaning of *Java*, the second meaning *coffee* can still be introduced in the concept space by occurrences of words like *espresso, latte, coffee beans*, etc. For this strategy, the size of the concept set is at most as large as that of the feature set. We call this strategy *Discriminative Concept Selection*.

We found out experimentally that both strategies give similar classification accuracy, but the second approach improves running time and scalability.

For initializing the $P[c|t]$ parameters we use a similar context-similarity strategy. For gathering the context of a topic, we can use two different strategies. The *supervised approach*, uses the top features as selected by decreasing Mutual Information (MI) [MN98a] value from the training set. For our implementation, we used the top 50 features with respect to MI rank. The *unsupervised approach*, relies on external resources such as encyclopedia, for gathering a broad description of the topic of interest. This can help particularly when training data is sparse. For example, we can improve the training description of the topic *Biology* by using a better description from an encyclopedia, e.g. the Biology page from Wikipedia. We note that this methodology of using background knowledge makes our model robust to variations in vocabulary or data distribution, problems that can occur when directly adding background data to the training set.

Once we have computed all the similarities for the (feature, concept) and (concept, topic) pairs, we normalize them, and interpret them as estimates of the probabilities $P[f|c]$ and $P[c|t]$. In the $sim(f, c)$ and $sim(c, t)$ computations, we only consider the (f, c) and (c, t) pairs in the pruned parameter space. The computed values are then used for initializing the parameters of the EM algorithm.

Enhanced EM Algorithm

The mapping step presented in the previous paragraph can be interpreted as building a prior probability distribution on the model parameters $P[f|c]$, $P[c|t]$. We can interpret the similarity-based estimates as pseudo-counts, and consider them in the inference process. This can be exploited in a Maximum A Posteriori (MAP) estimation of parameters, rather than simple maximum likelihood estimation. We denote by $\theta = (\theta_{f|c}, \theta_{c|t}, \theta_t)$, $\theta_{f|c} = P[f|c]$, $\theta_{c|t} = P[c|t]$, $\theta_t = P[t]$ our model parameters. Let $\theta_{f|c} \sim Dirichlet(\alpha_f^c)$ and $\theta_{c|t} \sim Dirichlet(\beta_c^t)$, where α_f^c is set to $sim(f, c)$ for each $f \in F$ and $c \in C$ and β_c^t is set to $sim(c, t)$ for each $c \in C$ and $t \in T$. Let θ_t be uniformly distributed, with density $g(\theta_t) = \frac{1}{|T|}$. The corresponding densities for $\theta_{f|c}$ and $\theta_{c|t}$ are:

$$g(\theta_{f|c}) \sim \prod_{f \in F} {\theta_{f|c}}^{\alpha_f^c},\ \theta_{f|c} \geq 0,\ \alpha_f^c \geq 0,\ \sum_{f \in F} \theta_{f|c} = 1,\ for\ each\ c \in C \tag{3.25}$$

$$g(\theta_{c|t}) \sim \prod_{c \in C} {\theta_{c|t}}^{\beta_c^t},\ \theta_{c|t} \geq 0,\ \beta_c^t \geq 0,\ \sum_{c \in C} \theta_{c|t} = 1,\ for\ each\ t \in T \tag{3.26}$$

Because $\theta_{f|c}$, $\theta_{c|t}$ and θ_t are independent random variables, their joint density can be written as: $g(\theta) = g(\theta_{f|c}, \theta_{c|t}, \theta_t) = g(\theta_{f|c}) \cdot g(\theta_{c|t}) \cdot g(\theta_t)$. Let $x = (f, t)$ be an observation from a multinomial distribution. Let $F(\theta|x) = g(\theta) \cdot L(x|\theta)$, where $g(\theta)$ defines a prior distribution on the parameters and $L(x|\theta)$ is the likelihood of the (f,t) samples (as in Section 2.2). We want to compute the MAP estimate

$$\theta = argmax_\theta \; F(\theta|x) \tag{3.27}$$

As $g(\theta_t)$ is a constant function, it does not influence the maximization, and we leave it out in the following estimations. Let

$$F(\theta|x) = (\prod_{c,f} \theta_{f|c}{}^{\alpha^c_f}) \cdot (\prod_{t,c} \theta_{c|t}{}^{\beta^t_c}) \cdot [\prod_{f,t} (\sum_{c} \theta_{f|c} \cdot \theta_{c|t} \cdot \theta_t)^{n(f,t)}] \tag{3.28}$$

For maximizing the above function we employ an EM algorithm (similar to the estimation in Section 3.2). The **E-Step** does not change. The parameter estimates $\theta_{f|c}$ and $\theta_{c|t}$ for the **M-Step** become:

$$\theta_{f|c} = P[f|c] = \frac{\alpha^c_f + \sum_{t\in T} n(f,t)P[c|(f,t)]}{\sum_{f'\in F}(\alpha^c_{f'} + \sum_{t\in T} n(f',t)P[c|(f',t)])} \tag{3.29}$$

$$\theta_{c|t} = P[c|t] = \frac{\beta^t_c + \sum_{f\in F} n(f,t)P[c|(f,t)]}{\sum_{c'\in C}(\beta^t_{c'} + \sum_{f\in F} n(f,t)P[c'|(f,t)])} \tag{3.30}$$

Combining the similarity-based estimates with the estimates based on training counts strengthens the model robustness. We empirically analyze the effect of using a context-similarity prior on model parameters in the next sections.

3.3 Inductive Learning

In the *inductive learning* paradigm, we are given a labeled set of documents and the objective is to learn a classification model from the labeled set which can accurately predict the labels of new documents. Typically one splits the labeled set into a training set and a test set. The training set is used for learning the classifier model. The test set is not seen by the classifier during training, and is used for measuring the performance of the classifier, by directly comparing for each document the predicted labels to the true labels. Furthermore, one can decide to split the labeled set into training, validation and test sets, where the validation set serves the purpose of evaluating different models (e.g. different parameter values and configurations), and the best model is selected for predicting the labels of the test set.

In this work we take the first approach, i.e., we split the labeled set into training and test sets. If training data is already very small, splitting it further for validation purpose is unlikely to give any insight or benefit. If training is enough (some hundreds of labeled samples), we can use cross-validation techniques for model selection.

3.4 Transductive Learning

In the *transductive learning* paradigm, the data collection to be automatically labeled is available before hand. The goal is to predict labels for the entire collection, using a few labeled samples provided by the user. The bottleneck is again the small size of the training set (labeled set), but one can take advantage of the unlabeled data in order to learn more about the nature of the input corpus, e.g. cluster substructure, feature distributions, etc. Moreover, transductive classification can be used as a building block for preparing the large training sets required by inductive learning techniques.

Both learning paradigms (inductive and transductive) can benefit from background knowledge about the problem at hand.

In order to analyze the various building blocks of the latent model presented in the previous section, we design several experiments, in which we test the influence of each of the building blocks on model performance. For the purpose of parameter analysis for the latent model, we compare the results to a Naive Bayes classifier as a baseline. In order to understand the benefit of using our latent model for improving classification accuracy, we compare the classification performance to several inductive and transductive classifiers.

3.5 Experimental Results

3.5.1 Methodology

The experimental setup is the following. We are given a collection of labeled documents which we split into training, with labels known to the classifier, and test, with labels unknown to the classifier. The goal is to automatically predict the labels of the documents from the test set. Furthermore, we want to study how much labeled data is needed for obtaining a reasonable classification accuracy, and what classifier methods perform best.

We analyze the performance of both inductive and transductive learning techniques in solving this problem. The inductive techniques learn based on the training data alone and have access to external background knowledge (e.g. ontologies, encyclopedia, but no unlabeled examples). The transductive techniques learn from both training (labeled) and test (unlabeled) data, as well as from external background knowledge. For analyzing the impact of various parameters on the performance of our latent model (Sections 3.5.4 and 3.5.5), we randomly split the collection into 1% training and remaining 99% test. For comparing several classifiers (Section 3.5.7), we average over 5 repetitions with random splits into training and test data, with the size of the splits ranging from 0.25% training data and 99.75% test data, up to 10% – 90% training-test splits.

3.5.2 Test Collections

We evaluate our techniques on three different text corpora. For all three collections, we pre-process the text by stemming all the words and removing stop-words from the standard SMART list [SMA].

The **Reuters-21578** dataset is part of the Reuters newswire collected in 1987 [Lew]. Of the 135 potential categories we select the top 10 categories in terms of training set size (so that each topic has a reasonable amount of labeled documents) and we keep only documents labeled with a unique topic. We pool together all the labeled documents for each topic. This results in a total of 8,024 documents. Per topic statistics are given in Table 3.1.

The **Amazon** dataset is a collection of editorial reviews of books organized into categories, extracted from `http://www.amazon.com`. From the available categorization, we selected all the

Table 3.1: Reuters-21578 corpus description.

Category Name	Labeled set size
earn	3,923
acq	2,292
crude	374
trade	327
money-fx	309
interest	272
money-supply	149
ship	144
sugar	122
coffee	112

editorial reviews of books under the categories *Biological Sciences, Mathematics, and Physics.* This amounts to 5,634 labeled documents. The corpus is described in Table 3.2.

Table 3.2: Amazon corpus description.

Category Name	Labeled set size
Mathematics	2,258
Biological Sciences	2,047
Physics	1,329

Table 3.3: Wikipedia corpus description.

Category Name	Labeled set size
Politics	3,923
Computer Science	2,292
Mathematics	975
Geography	919
Physics	513
Biology	435
Chemistry	371

The **Wikipedia** collection was obtained by a topic-focused crawl of `http://en.wikipedia.org`. The selected topics were *Politics, Computer Science, Physics, Chemistry, Biology, Mathematics, and Geography.* The crawl was started from the main pages of each of these classes and topic-specific words in the anchor text were used as indicators for whether an outgoing link should be followed. The dataset gathered this way contains 5,384 documents. Table 3.3 gives detailed corpus statistics. Due to the crawling procedure, this dataset contains a certain amount of noise. All datasets are available online[1].

3.5.3 Performance Measures

For evaluating the classification quality, we use the following measures commonly employed in evaluating text classifiers [MS00, Cha03]:

- Per topic:
  1. *Precision:* $\frac{\text{number of correct positive predictions}}{\text{number of positive predictions}}$
  2. *Recall:* $\frac{\text{number of correct positive predictions}}{\text{number of positive examples}}$
  3. *F1-measure:* the harmonic mean of Precision and Recall

$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

Typically a per topic *confusion matrix* is created which summarizes the classifier decisions. Precision and Recall are then computed from this matrix. Table 3.4 shows how the confusion matrix is defined. Using the confusion matrix Precision and Recall are defined as: $Precision = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$.

Table 3.4: Classifier evaluation based on the per topic confusion matrix.

		Predicted	
		Positive	Negative
True	Positive	TP	FN
	Negative	FP	TN

- Aggregation over all topics:
  1. *Micro-averaged Precision/Recall:* Precision/Recall computed on the confusion matrix formed by summing up all the per topic matrices.
  2. *Micro-averaged F1 measure:* The harmonic mean of Micro-averaged Precision and Micro-averaged Recall.
  3. *Macro-averaged Precision/Recall:* Averaged Precision/Recall over all topics.
  4. *Macro-averaged F1 measure:* The harmonic mean of Macro-averaged Precision and Macro-averaged Recall.

[1]Test collections available at `http://www.mpi-sb.mpg.de/~ifrim/pkdd06-datasets.zip`

Table 3.5: Details on the training/test per collection.

Collection	Training size	Test size	Features
Reuters21578	80	7,944	1,385
Amazon	56	5,578	2,041
Wikipedia	54	5,330	5,170

Micro-averaging makes the overall Precision and Recall depend most on the classes with the largest number of (positive) documents: the classification results can be poor for classes with few positive examples without affecting much the overall numbers. Thus, micro-averaged measures pay equal importance to each document [Cha03]. Macro-averaging pays equal importance to each topic, that is macro-averaged results reflect how well the documents in each topic were classified.

3.5.4 Parameter Analysis of the Inductive Latent Model

We first analyze the *inductively learned latent model* (**ILM**), i.e. the classifier only sees the labeled training data, but does not see any (unlabeled) test data. The main parameters of the latent model are:

1. the number of features,
2. the number of concepts (latent variables),
3. the number of EM iterations,
4. $P[f|c]$ (the probability of feature f given concept c),
5. $P[c|t]$ (the probability of concept c given topic t).

For the experiments in this section, we randomly select 1% of the collection as training data, and the rest as test data. We use Mutual Information [MN98a] for selecting features. We also show results for a Naive Bayes classifier (**NB**) tested in similar conditions as our **Inductive Latent Model** (**ILM**). For evaluating **ILM**, we design the following experiments.

Influence of the number of EM iterations

In this experiment we analyze the model behavior as a function of the number of EM iterations.

We take all the distinct terms in the training set as features. The number of concepts is set using the *Discriminative Concept Selection* strategy presented in Section 3.2.4. We vary the number of EM iterations (By 0 EM iterations we refer to using only the mapping estimates of $P[f|c]$ and $P[c|t]$, without $n(f,t)$ training counts) between 0 and 20 and study the effect of this parameter on classification quality. We show results with both random initialization (**rand-init**) and context-similarity initialization (**sim-init**) of $P[f|c]$ and $P[c|t]$. Table 3.5 shows statistics on the input data.

In Table 3.6 we show classification results for **Naive Bayes** over all three collections. We use Naive Bayes results as a baseline for comparison with **ILM**.

Table 3.6: Results on all collections using Naive Bayes.

Collection	NB	
	micro-avg F1	macro-avg F1
Reuters21578	79.7	60.3
Amazon	74.9	73.2
Wikipedia	70.2	70.3

Table 3.7: **Reuters21578**. Influence of the number of EM iterations on micro/macro-averaged F1 using **rand-init** for $P[f|c]$ and $P[c|t]$.

EM iteration	ILM	
	micro-avg F1	macro-avg F1
0	28.4	4.8
1	76.2	54.5
2	79.5	60.7
3	**81.0**	61.4
4	79.6	59.1
5	79.4	58.9
10	74.9	50.6
15	73.4	46.3
20	72.6	44.0

Table 3.8: **Reuters21578**. Influence of the number of EM iterations on micro/macro-averaged F1 using **sim-init** for $P[f|c]$ and $P[c|t]$.

EM iteration	ILM	
	micro-avg F1	macro-avg F1
0	69.8	50.5
1	75.9	55.1
2	**77.5**	55.8
3	77.3	55.0
4	77.1	54.5
5	76.7	54.2
10	75.7	52.8
15	75.5	51.8
20	75.5	51.8

Table 3.7 and Table 3.8 present results on the Reuters collection for **ILM**. We give in bold the best micro-averaged F1 result across EM iterations. We observe that for both random and similarity-based initialization (denoted by **rand-init** and **sim-init**) of $P[f|c]$ and $P[c|t]$, the classification quality (in terms of micro-averaged F1) peaks at 2-3 EM iterations, and then starts decreasing with more iterations, most likely due to overfitting. We also notice that **ILM** outperforms **NB** on this collection, when the $P[f|c]$ and $P[c|t]$ parameters are randomly initialized.

Table 3.9: **Amazon**. Influence of the number of EM iterations on micro/macro-averaged F1 using **rand-init** for $P[f|c]$ and $P[c|t]$.

EM ITERATION	ILM	
	MICRO-AVG F1	MACRO-AVG F1
0	36.2	26.9
1	**74.4**	72.9
2	73.9	72.7
3	73.8	72.6
4	73.7	72.3
5	73.4	71.8
10	72.1	69.7
15	70.9	68.2
20	70.5	67.7

Table 3.10: **Amazon**. Influence of the number of EM iterations on micro/macro-averaged F1 using **sim-init** for $P[f|c]$ and $P[c|t]$.

EM ITERATION	ILM	
	MICRO-AVG F1	MACRO-AVG F1
0	61.9	63.0
1	71.8	71.7
2	72.1	71.9
3	72.4	72.0
4	72.6	71.9
5	73.0	72.0
10	**73.4**	71.6
15	73.1	71.0
20	72.9	70.7

On Amazon (Table 3.9 and Table 3.10) the best number of EM iterations differs depending on the initialization. For **rand-init**, the micro-averaged F1 is highest in the first EM iteration, while for **sim-init**, the peak is reached with about 10 iterations. In Table 3.11 and Table 3.12 we summarize the results on the Wikipedia dataset. Again, the classification quality varies with the number of EM iterations. For **rand-init** micro-averaged F1 reaches its peak on the third EM iteration, while for **sim-init** it peaks on the first iteration. This experiment shows that the classification quality is influenced by the number of EM iterations. Throughout the next experiments we fix the number of EM iterations to the value that resulted in highest micro-averaged F1 for each dataset and parameters configuration. We summarize these values in Table 3.13.

Table 3.11: **Wikipedia**. Influence of the number of EM iterations on micro-averaged F1 using **rand-init** for $P[f|c]$ and $P[c|t]$.

EM ITERATION	ILM	
	MICRO-AVG F1	MACRO-AVG F1
0	17.7	13.6
1	69.2	69.7
2	70.2	70.4
3	**70.4**	70.5
4	70.3	70.3
5	69.6	69.7
10	65.2	65.8
15	62.6	63.5
20	61.7	62.8

Table 3.12: **Wikipedia**. Influence of the number of EM iterations on micro/macro-averaged F1 using **sim-init** for $P[f|c]$ and $P[c|t]$.

EM ITERATION	ILM	
	MICRO-AVG F1	MACRO-AVG F1
0	66.7	65.2
1	**68.3**	68.0
2	67.7	67.7
3	67.6	67.3
4	67.1	66.7
5	66.8	66.6
10	66.0	65.8
15	65.7	65.3
20	65.5	65.1

Table 3.13: Best number of EM iterations on all collections..

COLLECTION	ILM	
	RAND-INIT	SIM-INIT
REUTERS21578	3	2
AMAZON	1	10
WIKIPEDIA	3	1

Influence of the number of concepts

In this section, we test various strategies for selecting concepts (i.e. initializing the model structure), including the two strategies presented in Section 3.2.4. All the distinct terms in the training set are taken as features. The number of EM iterations is set as in Table 3.13. We initialize $P[f|c]$

and $P[c|t]$ to random values. Tables 3.14 to 3.16 show results for varying number of concepts.

Table 3.14: **Reuters21578**. Influence of the number of concepts on micro/macro-averaged F1.

Concepts	ILM	
	micro-avg F1	macro-avg F1
10	75.3	41.5
50	79.4	53.7
100	79.4	54.8
300	80.4	60.2
500	80.6	60.3
1,000	81.0	60.6
2,000	80.9	60.1
5,000	**81.3**	62.3
ConceptSel-All: 6,969	81.2	62.0
ConceptSel-Discriminative: 1,107	**81.0**	61.4

Table 3.15: **Amazon**. Influence of the number of concepts on micro/macro-averaged F1.

Concepts	ILM	
	micro-avg F1	macro-avg F1
3	48.6	48.7
50	73.1	70.4
100	**74.6**	72.3
300	73.5	72.3
500	74.3	72.5
1,000	74.2	72.7
2,000	74.2	72.7
5,000	74.4	72.7
ConceptSel-All: 9,715	74.2	72.7
ConceptSel-Discriminative: 1,723	**74.4**	73.0

This experiment shows that the number of latent variables in the model affects the quality of classification. It also shows that initializing the structure of the latent model based on the ontology using the strategies presented in Section 3.2.4 is a good approach. The advantage of using the techniques proposed in Section 3.2.4 is that the latent variables are explicitly represented in the ontology and their selection is driven by the training set, without the need for further cross-validation or other model selection techniques.

Influence of the number of features

In this section we vary the number of features in the model, and measure the classification performance. We initialize $P[f|c]$ and $P[c|t]$ using random values and context-similarity estimates. For

Table 3.16: **Wikipedia**. Influence of the number of concepts on micro/macro-averaged F1.

Concepts	ILM	
	micro-avg F1	macro-avg F1
7	52.4	52.2
50	70.0	69.3
100	70.4	70.2
300	69.9	70.2
500	69.9	70.1
1,000	70.2	70.4
2,000	70.1	70.2
5,000	70.4	70.5
ConceptSel-All: 18,426	70.4	70.5
ConceptSel-Discriminative: 4,024	**70.4**	70.5

all the remaining experiments, we set the number of concepts using our *Discriminative Concept Selection* strategy. The number of EM iterations is set as in Table 3.13. We show results with varying number of features for both **NB** and **ILM**.

The NB classifier results improve with increased number of features. On the Reuters collection (Table 3.17) it achieves the peak of micro-averaged F1 with 1,000 features, while on the second collection (Table 3.19) it achievs maximum classification quality when using all the distinct terms in the training set as features. On the Wikipedia collection (Table 3.21), it achieves the best

Table 3.17: **Reuters21578**. Influence of the number of features on micro/macro-averaged F1. NB versus **rand-init** ILM.

Features	NB		ILM	
	micro-avg F1	macro-avg F1	micro-avg F1	macro-avg F1
100	75.1	57.5	75.3	56.7
500	79.7	61.4	79.8	59.1
1,000	**80.1**	61.1	80.3	60.1
1,385	79.7	60.3	**81.0**	61.4

Table 3.18: **Reuters21578**. Influence of the number of features on micro/macro-averaged F1. NB versus **sim-init** ILM.

Features	NB		ILM	
	micro-avg F1	macro-avg F1	micro-avg F1	macro-avg F1
100	75.1	57.5	68.4	39.3
500	79.7	61.4	76.1	56.0
1,000	**80.1**	61.1	**78.3**	56.7
1,385	79.7	60.3	77.5	55.8

Table 3.19: **Amazon**. Influence of the number of features on micro/macro-averaged F1. NB versus **rand-init** ILM.

Features	NB		ILM	
	micro-avg F1	macro-avg F1	micro-avg F1	macro-avg F1
100	69.7	69.6	69.9	67.7
500	72.8	72.1	72.9	71.4
1,000	74.2	72.4	**74.9**	73.3
1,500	74.8	73.1	73.9	72.7
2,000	74.7	72.9	73.8	72.4
2,041	**74.9**	73.2	74.4	72.9

Table 3.20: **Amazon**. Influence of the number of features on micro/macro-averaged F1. NB versus **sim-init** ILM.

Features	NB		ILM	
	micro-avg F1	macro-avg F1	micro-avg F1	macro-avg F1
100	69.7	69.6	69.0	68.5
500	72.8	72.1	72.1	71.1
1,000	74.2	72.4	73.3	71.9
1,500	74.8	73.1	72.8	71.8
2,000	74.7	72.9	73.3	71.7
2,041	**74.9**	73.2	**73.4**	71.6

Table 3.21: **Wikipedia**. Influence of the number of features on micro/macro-averaged F1. NB versus **rand-init** ILM.

Features	NB		ILM	
	micro-avg F1	macro-avg F1	micro-avg F1	macro-avg F1
100	63.4	62.9	62.7	62.3
500	69.3	68.9	69.2	68.5
1,000	69.9	70.0	69.6	69.5
1,500	70.1	70.1	69.8	69.8
2,000	**70.5**	70.6	70.1	69.9
5,000	70.0	70.1	70.0	70.0
5,170	70.2	70.3	**70.4**	70.5

performance with 2,000 features out of 5,170 distinct terms. On Reuters21578 (Table 3.17), ILM achieves highest micro-averaged F1 using all the distinct terms in the training set (1,385 features). ILM with rand-init is slightly better (+1.3%) than NB on this dataset. ILM with sim-init is slightly worse than ILM with rand-init (-1.7%). On Amazon (Table 3.19), ILM achieves the best micro-averaged F1 with 1,000 features (out of 2,041 distinct terms in training). For rand-init ILM has micro-averaged F1 which is the same as for NB (NB: 74.9%, ILM: 74.9%). For sim-

Table 3.22: **Wikipedia**. Influence of the number of features on micro/macro-averaged F1. NB versus **sim-init** ILM.

Features	NB		ILM	
	micro-avg F1	macro-avg F1	micro-avg F1	macro-avg F1
100	63.4	62.9	63.3	62.8
500	69.3	68.9	68.7	68.5
1,000	69.9	70.0	69.2	69.6
1,500	70.1	70.1	**69.7**	69.8
2,000	**70.5**	70.6	69.6	69.7
5,000	70.0	70.1	68.6	68.2
5,170	70.2	70.3	68.3	68.0

init, ILM behaves slightly worse (micro-averaged F1: 73.4%). On Wikipedia (Table 3.21), ILM achieves highest micro-averaged F1 using all the features (5,170 distinct terms) with rand-init, and using only 2,000 features with sim-init of the $P[f|c]$ and $P[c|t]$ parameters. Compared to NB, the micro-averaged F1 results are similar (NB: 70.5%, ILM: 70.4%).

We note that using the similarity-based prior does not help in improving the classification accuracy of ILM as compared to its randomly initialized counterpart. On all three collections, the random initialization of these parameters resulted in better classification quality. This suggests the training may be too small and of potentially too low quality to be used alone for inferring the similarity-based prior. We summarize the best number of features for each collection and parameter initialization in Table 3.23.

Table 3.23: Best number of features on all collections..

Collection	ILM	
	rand-init	sim-init
Reuters21578	1,385	1,000
Amazon	1,000	2,041
Wikipedia	5,170	1,500

Influence of P[f|c]

In this experiment, we test the influence of the similarity-based prior distribution of $P[f|c]$ on the classification quality of the latent model. We fix the number of features and the number of EM iterations to the values that achieved best micro-averaged F1 for each collection (Table 3.23, Table 3.13). We initialize $P[f|c]$ using the context-similarity heuristic presented in Section 3.2.4. We initialize the $P[c|t]$ to random values.

Tables 3.24 and 3.25 show results for ILM with different initializations of the parameters $P[f|c]$. From this experiment we see that the similarity-initialization as a prior on $P[f|c]$ does not help the overall classification quality of ILM. This can be explained by the quality and size of the

Table 3.24: Results with best parameter settings on all collections using rand-init for both $P[f|c]$ and $P[c|t]$.

Collection	ILM	
	rand-init P[f|c], rand-init P[c|t]	
	EM iter	micro-avgF1
Reuters21578	3	81.0
Amazon	1	74.9
Wikipedia	3	70.4

Table 3.25: Influence of sim-init for P[f|c].

Collection	ILM			
	sim-init P[f|c], rand-init P[c|t]			
	EM iter	micro-avg F1	EM iter	micro-avg F1
Reuters21578	3	77.9	2	78.1
Amazon	1	73.0	10	73.9
Wikipedia	3	69.7	1	69.0

Table 3.26: Influence of sim-init for P[c|t].

Collection	ILM			
	rand-init P[f|c], sim-init P[c|t]			
	EM iter	micro-avg F1	EM iter	micro-avg F1
Reuters21578	3	78.7	2	79.0
Amazon	1	73.8	10	73.3
Wikipedia	3	68.9	1	69.7

training data, e.g. when the training data is small, features tend to occur sparsely in the training set, therefore inferring a prior mapping between features and concepts based on training contexts alone, does not boost performance as compared to random initialization.

Influence of P[c|t]

We fix the number of features and the number of EM iterations to the values that achieve best micro-averaged F1 for each collection (Table 3.23, Table 3.13). We initialize $P[f|c]$ to random values and $P[c|t]$ using the context-similarity heuristic presented in Section 3.2.4.

From Table 3.26 we observe the same effect when using the context-similarity prior for P[c|t], as we noted for $P[f|c]$. When training is sparse, the similarity-based prior does not improve over random initialization of $P[f|c]$ and $P[c|t]$.

3.5.5 Parameter Analysis of the Transductive Latent Model

For testing the *transductively learned latent model* (**TLM**), we investigate the effect of using unlabeled examples and external background knowledge for selecting features and for estimating the prior on $P[f|c]$ and $P[c|t]$.

Influence of unlabeled examples

We fix the number of features to 5,000, selected based on $tf \cdot idf$ ranking computed on the full (labeled and unlabeled) collection. The number of concepts is selected by our *Discriminative Concept Selection* technique. We vary the number of EM iterations from 0 to 10. We initialize the $P[c|t]$ parameters to random values. We test both the random initialization of P[f|c] and the similarity-based estimate of $P[f|c]$ from *labeled and unlabeled examples.* In this section we try to answer several questions.

1. *Does enlarging the feature set alone, without estimating a context-similarity prior on $P[f|c]$ and $P[c|t]$, increase classification accuracy?*

Table 3.27: Influence of unlabeled features.

Collection	TLM		
	rand-init P[f|c], rand-init P[c|t]		
	EM iter	micro-avg F1	macro-avg F1
Reuters21578	0	28.3	4.4
	1	77.2	51.5
	2	80.2	60.5
	3	81.5	61.2
	4	**81.8**	61.8
	5	81.6	61.9
	10	76.2	52.8
Amazon	0	36.9	31.4
	1	**73.9**	72.5
	2	73.4	72.5
	3	73.0	72.2
	4	73.1	72.1
	5	72.8	71.6
	10	71.4	69.4
Wikipedia	0	14.2	12.3
	1	68.9	69.4
	2	69.8	69.7
	3	70.1	69.9
	4	**70.2**	70.1
	5	70.1	69.9
	10	66.1	66.7

Simply enlarging the feature set by selecting features from the entire collection rather than training alone does not help. Lets have a closer look at the effect of extending the feature set by adding *unlabeled features*, i.e. features of the collection which may not occur in the labeled training set. From equations Equation 3.17 and 3.29 we observe that unlabeled features can only affect the estimation of $P[c|f,t]$ directly. The estimates of $P[f|c]$ and $P[c|t]$ for *unlabeled features* does not change during EM iterations, but they directly influence $P[c|f,t]$ and thus indirectly influence the estimates of $P[f|c]$ and $P[c|t]$ for the *labeled features* (features with non-zero occurrences in the training set). Since we initialize the $P[f|c], P[c|t]$ parameters with random values, they cannot influence the $P[c|f,t]$ in a meaningful way, and thus the model using both labeled and unlabeled features does not gain much accuracy over its counter part which only involves labeled features. The classification results of our **Transductive Latent Model (TLM)** are summarized on Table 3.27. On Reuters, the micro-averaged F1 goes up from 81.0% to 81.8%, on Amazon it goes down from 74.9% to 73.9%, and on Wikipedia it goes down from 70.4% to 70.2%.

2. *Does enlarging the feature set and inferring a similarity-based prior on $P[f|c]$ increase classification accuracy?*

Table 3.28: Influence of sim-init for P[f|c] from labeled and unlabeled documents.

Collection	TLM		
	sim-init P[f|c], rand-init P[c|t]		
	EM iter	micro-avg F1	macro-avg F1
Reuters21578	0	27.0	5.6
	1	83.1	62.4
	2	83.8	65.7
	3	**85.1**	69.6
	4	84.6	68.8
	5	82.1	64.2
	10	79.9	59.1
Amazon	0	33.9	26.7
	1	77.9	76.0
	2	**77.9**	76.6
	3	76.7	76.1
	4	75.0	75.3
	5	74.5	74.9
	10	73.6	72.2
Wikipedia	0	13.5	9.5
	1	72.9	73.1
	2	73.7	73.8
	3	**74.3**	74.2
	4	74.2	74.0
	5	73.2	73.0
	10	68.6	68.6

Since we initialize the $P[f|c]$ based on the context-similarity prior, the parameters corresponding to unlabeled features positively affect the overall parameter estimation. Table 3.28 present TLM results for all three collections. On the Reuters corpus, TLM achieves a 4% improvement in terms of micro-averaged F1 (from 81.0% for ILM to 85.1% for TLM), and 3-4% improvement on Amazon and Wikipedia respectively (Amazon: from 74.9% for ILM to 77.9% for TLM, Wikipedia: from 70.4% for ILM to 74.3% for TLM).

3. *Does enlarging the feature set and estimating a similarity prior on both $P[f|c]$ and $P[c|t]$ increase classification accuracy?*

Table 3.29: Influence of sim-init for P[f|c] and P[c|t] from labeled and unlabeled documents.

Collection	TLM		
	sim-init P[f|c], sim-init P[c|t]		
	EM iter	micro-avg F1	macro-avg F1
Reuters21578	0	81.6	58.0
	1	82.3	62.3
	2	84.1	66.2
	3	**84.8**	67.9
	4	84.6	67.3
	5	84.0	66.0
	10	82.5	62.7
Amazon	0	73.4	73.4
	1	73.0	75.5
	2	71.9	74.7
	3	72.4	74.6
	4	73.6	74.8
	5	74.5	74.9
	10	**75.9**	74.5
Wikipedia	0	**74.6**	74.3
	1	74.1	74.1
	2	73.3	73.6
	3	72.6	73.2
	4	71.8	72.2
	5	71.3	71.5
	10	70.7	70.5

Table 3.29 shows that using the similarity prior on both $P[f|c]$ and $P[c|t]$ helps classification accuracy as compared to ILM, but not as compared to TLM with sim-init $P[f|c]$ and rand-init $P[c|t]$. The reason for the latter observation is that the estimate for $P[f|c]$ can take advantage of unlabeled data while the estimate for $P[c|t]$ relies on labeled data alone, thus it is better to rely on the prior only for $P[f|c]$ and not for $P[c|t]$. As compared to TLM with sim-init $P[f|c]$ and rand-init $P[c|t]$, TLM with sim-init for both parameters gives the following results: on Reuters, micro-averaged F1 decreases from 85.1% to 84.4%, on Amazon from 77.9% to 75.9% and on Wikipedia

it increases slightly from 74.3% to 74.6%.

Influence of external background knowledge

For this experiment, we set the number of features to 5,000, selected based on $tf \cdot idf$ ranking computed on the full (labeled and unlabeled) collection. The number of concepts is selected by our discriminative Concept Selection technique. We vary the number of EM iterations from 0 to 10. We run two experiments, in order to check the influence of the $P[c|t]$ component. The first experiment initializes $P[f|c]$ to random value, while the second estimates $P[f|c]$ from labeled and unlabeled examples. For both experiments we initialize $P[c|t]$ using both training and Wikipedia pages to build contexts for topics. The Wikipedia pages are treated as explicit training data.

Table 3.30 shows the results of the first experiment. Using sim-init for $P[c|t]$ alone (with rand-init $P[f|c]$) has a positive effect on classification accuracy as compared to initializing both parameters with random values. Nevertheless, using context-similarity initialization for both $P[f|c]$ and $P[c|t]$ parameters gives the best results.

Table 3.30: Influence of sim-init for P[c|t] from training and Wikipedia pages.

Collection	TLM		
	rand-init P[f|c], sim-init P[c|t]		
	EM iter	micro-avg F1	macro-avg F1
Reuters21578	0	7.6	4.2
	1	79.3	57.5
	2	82.0	59.9
	3	**82.8**	59.1
	4	82.4	58.6
	5	81.6	58.5
	10	80.3	59.7
Amazon	0	35.5	33.7
	1	81.7	80.6
	2	81.9	80.8
	3	82.0	81.0
	4	82.3	81.1
	5	**82.4**	81.2
	10	81.4	79.9
Wikipedia	0	13.5	13.0
	1	**76.0**	75.3
	2	74.8	75.0
	3	73.1	74.4
	4	73.3	74.3
	5	73.7	74.3
	10	73.5	73.7

Table 3.31 shows **TLM** results for the second experiment. We note that the background knowledge from Wikipedia has a strong positive effect on the classification quality on the Amazon and Wikipedia corpora, while for the Reuters corpus it leads to a slight decrease as compared to the model learned with no Wikipedia data. Nevertheless, we observe that for all collections including external background knowledge leads to a *faster peak in the micro-averaged F1*, e.g. on all collections the highest micro-averaged F1 value is reached within the first three EM iterations. The results with respect to micro-averaged F1 without and with using Wikipedia as an extension for training are: on Reuters 85.1% versus 84.2%, on Amazon 77.9%versus 84.2% and on Wikipedia 74.3% versus 79.6%.

For comparison we also show NB results on the Wikipedia-extended training set. Table 3.32 summarizes the details about the extended training data. The NB results are shown in Table 3.33.

Table 3.31: Influence of sim-init for P[f|c] from labeled and unlabeled documents, sim-init for P[c|t] from training and Wikipedia pages.

Collection	TLM		
	sim-init P[f|c], sim-init P[c|t]		
	EM iter	micro-avg F1	macro-avg F1
Reuters21578	0	57.7	53.1
	1	81.6	60.0
	2	83.7	61.6
	3	**84.2**	62.6
	4	83.6	61.7
	5	83.3	61.4
	10	82.7	63.3
Amazon	0	82.5	82.0
	1	**84.2**	83.3
	2	84.1	83.1
	3	83.9	82.8
	4	83.9	82.8
	5	83.7	82.5
	10	83.0	81.6
Wikipedia	0	**79.6**	78.4
	1	79.2	78.6
	2	77.1	77.5
	3	75.9	76.5
	4	75.6	76.1
	5	75.2	75.7
	10	74.8	75.0

Table 3.32: Details on the training/test per collection. Training extended with Wikipedia pages.

Collection	Extended Training size	Test size	Extended Training Features
Reuters21578	90	7,944	5,082
Amazon	59	5,578	3,500
Wikipedia	61	5,330	6,802

Table 3.33: Results on all collections using Naive Bayes. Training extended with Wikipedia pages.

Collection	NB		NB-wikiext	
	micro-avg F1	macro-avg F1	micro-avg F1	macro-avg F1
Reuters21578	79.7	60.3	80.1	61.7
Amazon	74.9	73.2	83.3	82.2
Wikipedia	70.2	70.3	76.9	76.5

3.5.6 Parameter Analysis of the Latent Model: Discussion

In the previous experiments we analyzed the building blocks of the latent model proposed, and observed the influence of each of these building blocks on the overall model performance. We have noted the following effects:

1. The *number of EM iterations* affects the classification results. Typically the model reaches the peak of classification quality within the first 10 iterations.

2. Using an external ontology of *concepts* for initializing the *model structure* is beneficial and avoids the need for using other model selection techniques.

3. The *number of features* in the model influences the classification performance. A few thousand features seem to be enough for achieving good classification quality. Models learned transductively, e.g. with labeled and unlabeled features, provide better results than models learned inductively.

4. Using a *context-similarity heuristic for initializing* $P[f|c]$ affects the classification performance. If the similarity-based prior on $P[f|c]$ is learned only from little training, this may negatively affect the results. In this case initializing these parameters to random values can give better results. If instead the prior is learned from a large collection of labeled and unlabeled documents, the prior can improve micro and macro-averaged F1 results by up to 5%.

5. Using a *context-similarity heuristic for initializing* $P[c|t]$ affects the classification performance. If the similarity-based prior on $P[c|t]$ is learned only from little training, this may negatively affect the results. If instead the prior is learned from training and other sources of background knowledge about the topics, the prior can improve micro and macro-averaged F1 results by up to 8%.

6. Using a *context-similarity prior on both* $P[f|c]$ *and* $P[c|t]$ parameters positively affects the classification quality and can improve the results by up to 10%.

3.5.7 Comparison to Other Techniques

In the previous sections we have analyzed the model behavior depending on various parameter values. In this section we present results comparing our technique to other state-of-the-art (inductive as well as transductive) classifiers. We compare the following classifiers:

- (Inductive) Multinomial Naive Bayes (NB) [MN98a]
- Inductive/Transductive Latent Model (ILM, TLM) developed in this thesis
- Inductive/Transductive Support Vector Machines (ISVM, TSVM) [Joa98] [Joa99b]
- (Transductive) Spectral Graph Transducer (SGT) [Joa03]

To evaluate the classification performance of all the methods compared, we focus on micro/macro-averaged F1 [Seb02] as our main performance metrics. The results are averages over 5 repetitions with random splits into training and test data, with the size of the splits ranging from 0.25% training data and 99.75% test data, up to 10% − 90% training-test splits. When learning classifiers from sparse training data, parameter tuning is unreliable, i.e. it is difficult to perform meaningful cross-validation or to provide held-out data for model selection. In order to set the hyperparameters for each of the compared methods we tune them directly *on the test set*, using the same training/test data employed in the previous section for the analysis of ILM/TLM parameters (Section 3.5.4, Section 3.5.5). The results presented can thus be interpreted as an upper bound on the realistic performance these methods will deliver in practice. We show experiments for two settings: the first setting does not use external information from Wikipedia to improve the description of the topics. In the second setting (coined **wikiext**) we use Wikipedia pages as broad descriptions of the topics of interest (and add them as explicit training for all the methods).

Parameter tuning for all methods

In this paragraph we summarize the decisions we took for setting the hyperparameters for all the methods compared. For ILM/TLM we rely on the empirical study presented in the previous section in order to set the number of EM iterations and whether we turn on or off the similarity prior for the model parameters $P[f|c]$ and $P[c|t]$. We observed that using more than 10,000 features did not improve the classification performance of NB, ILM and TLM, therefore we limit the number of features to 10,000 for these methods. The features are selected from the training set using Mutual Information for NB and ILM, and from the entire collection by $tf \cdot idf$ rank for TLM. The number of concepts is set according to our *Discriminative Concept Selection* strategy. Table 3.34 and Table 3.35 summarize the parameter settings used for ILM and TLM, both with and without Wikipedia as background data.

As suggested in [Joa98, Joa99b, Joa03] we do not use feature selection for ISVM, TSVM and SGT, i.e. we use all the distinct terms in the collection as features. We use SVMLight [Joa98] with linear kernel. In order to set the hyperparameter of SVM (parameter C: the trade-off between training error and margin), we run experiments on the same data as in Section 3.5.4 and choose the best parameter according to the performance on the test set. Table 3.36 and Table 3.37 show results for varying the hyperparameter of SVM. For TSVM we limited the number of values tried for the C parameter due to the high training time. For C larger than 0.01, it took more than 2 days to train TSVM. Furthermore, the classification quality kept on decreasing for values of C lower than 1.

The SGT hyperparameter is the number of nearest neighbors (denoted by k) for each node in the neighborhood graph. Similar to the hyperparameter of SVM, we set the SGT main parameter to the *best performing value on the test set.* Table 3.38 and Table 3.39 show results for varying the hyperparameter of SGT.

Table 3.34: Parameter settings for ILM.

Collection	ILM	ILM-wikiext
	rand-init $P[f\|c]$, $P[c\|t]$	rand-init $P[f\|c]$, sim-init $P[c\|t]$
	EM iterations	EM iterations
Reuters21578	3	3
Amazon	3	3
Wikipedia	3	2

Table 3.35: Parameter settings for TLM.

Collection	TLM	TLM-wikiext
	sim-init $P[f\|c]$, rand-init $P[c\|t]$	sim-init $P[f\|c]$, sim-init $P[c\|t]$
	EM iterations	EM iterations
Reuters21578	3	3
Amazon	2	1
Wikipedia	3	1

Table 3.36: Influence of the SVM hyperparameter on micro/macro-averaged F1.

Collection	C	ISVM		TSVM	
		micro-avg F1	macro-avg F1	micro-avg F1	macro-avg F1
Reuters21578	def	73.4	34.0	81.2	54.8
	10	74.3	39.4	81.2	50.4
	1	73.4	34.0	**81.4**	55.0
	0.1	73.6	31.6	79.6	51.5
	0.01	73.3	31.5	68.7	51.9
	0.001	**78.5**	55.8	57.5	53.2
	0.0001	77.3	54.4	N/A	N/A
Amazon	def	72.7	69.3	73.3	71.2
	10	74.3	71.3	72.4	70.5
	1	72.7	69.3	**73.7**	71.5
	0.1	72.7	69.3	63.5	61.7
	0.01	72.0	68.6	52.9	53.8
	0.001	**74.1**	71.9	N/A	N/A
	0.0001	73.7	71.5	N/A	N/A
Wikipedia	def	65.4	66.2	70.5	67.6
	10	**68.8**	68.4	70.2	68.1
	1	65.4	66.2	**70.7**	67.8
	0.1	65.2	66.0	61.7	57.0
	0.01	68.0	68.0	59.6	52.7
	0.001	62.7	63.7	N/A	N/A
	0.0001	63.2	64.9	N/A	N/A

Table 3.37: Influence of the SVM hyperparameter on micro/macro-averaged F1 with Wikipedia pages.

Collection	C	ISVM		TSVM	
		micro-avg F1	macro-avg F1	micro-avg F1	macro-avg F1
Reuters21578	def	73.9	36.0	80.4	56.8
	10	75.2	46.6	80.6	57.1
	1	73.9	36.0	**80.7**	57.5
	0.1	74.0	34.1	78.5	52.5
	0.01	74.2	34.1	N/A	N/A
	0.001	**78.8**	55.5	N/A	N/A
	0.0001	78.8	55.2	N/A	N/A
Amazon	def	75.2	72.7	73.0	71.2
	10	77.4	75.1	**73.4**	71.7
	1	75.2	72.7	73.0	71.2
	0.1	75.2	72.7	66.0	64.4
	0.01	73.7	70.9	N/A	N/A
	0.001	76.1	74.4	N/A	N/A
	0.0001	**77.7**	75.9	N/A	N/A
Wikipedia	def	70.5	71.3	70.9	68.3
	10	73.1	73.0	71.2	70.2
	1	70.5	71.3	**71.5**	69.0
	0.1	70.4	71.2	60.1	54.8
	0.01	68.6	70.7	N/A	N/A
	0.001	**73.6**	73.7	N/A	N/A
	0.0001	60.5	64.4	N/A	N/A

Table 3.38: Influence of the SGT hyperparameter on micro/macro-averaged F1.

Collection	k	SGT	
		micro-avg F1	macro-avg F1
Reuters21578	10	64.9	62.6
	50	76.4	70.3
	100	79.9	72.8
	300	84.6	74.6
	500	**84.8**	73.0
	800	84.2	70.9
	1000	84.0	69.2
Amazon	10	84.9	83.8
	50	86.1	84.9
	100	86.3	85.4
	300	**86.6**	85.9
	500	85.7	85.0
	800	85.6	84.9
	1000	86.0	85.1
Wikipedia	10	**80.7**	80.5
	50	78.2	78.8
	100	77.8	78.5
	300	79.8	79.1
	500	79.7	79.0
	800	80.2	79.6
	1000	79.9	79.3

Table 3.39: Influence of the SGT hyperparameter on micro/macro-averaged F1 with Wikipedia pages.

Collection	k	SGT	
		micro-avg F1	macro-avg F1
Reuters21578	10	64.8	62.8
	50	78.6	74.5
	100	83.1	76.2
	300	87.8	79.1
	500	88.2	79.1
	800	**88.4**	78.2
	1000	87.6	75.7
Amazon	10	85.1	84.1
	50	86.2	84.9
	100	86.5	85.6
	300	**86.6**	85.8
	500	85.9	85.2
	800	85.9	85.2
	1000	85.8	85.0
Wikipedia	10	**81.9**	81.2
	50	79.8	79.9
	100	79.7	79.8
	300	79.7	79.2
	500	80.3	79.3
	800	80.5	79.9
	1000	79.7	79.3

Results without Wikipedia background data

Tables 3.40 to 3.42 summarize the results for all the methods compared across the three datasets, in the experimental setting in which no Wikipedia pages are used to overcome the training sparsity.

Behavior of Inductive Methods. Analyzing the behavior of the inductive methods, we observe that ILM gives better results than NB on Reuters and Wikipedia (Table 3.40, Table 3.42) and it is comparable to NB on Amazon (Table 3.41). For small training sets ISVM is slightly worse than NB and ILM, and becomes better for larger training data.

Behavior of Transductive Methods. On all collections the transductive methods (TSVM, SGT, TLM) perform better than the inductive methods (NB, ILM, ISVM). This is normal, since these methods can take advantage of the information encoded in the unlabeled data available to the classifier. Among the transductive methods, TLM and SGT behaves best.

Table 3.40 shows results for the Reuters dataset. For small training (up to 40 training documents or 0.5% of the total dataset), TLM is about 4% better than SGT with respect to micro-averaged F1 (i.e 79.26% versus 75.40%), while for larger training sets SGT is about 1% better than TLM in terms of micro-averaged F1.

Table 3.40: **Reuters21578.** Micro/macro-averaged F1 for different training set sizes.

F1	Training (%)	NB	ILM	ISVM	TSVM	SGT	TLM
Micro-avg F1	33 (0.4%)	64.34	64.96	63.28	49.72	75.40	79.26
	40 (0.5%)	77.06	77.72	71.72	78.20	83.20	83.42
	80 (1%)	81.44	82.66	75.88	85.92	87.40	86.76
	160 (2%)	84.30	86.04	66.24	89.50	90.60	88.90
	400 (5%)	89.82	90.02	85.66	90.90	92.28	91.68
	800 (10%)	91.92	91.16	77.32	93.58	92.92	92.54
Macro-avg F1	33 (0.4%)	59.20	58.10	60.12	40.84	72.58	69.40
	40 (0.5%)	55.76	60.16	49.14	49.84	68.22	62.36
	80 (1%)	61.20	63.70	54.78	65.06	76.16	71.50
	160 (2%)	64.28	71.24	36.92	74.70	82.76	75.94
	400 (5%)	78.38	80.62	73.26	82.30	86.04	83.52
	800 (10%)	83.34	83.36	65.14	87.16	87.50	85.76

On the Amazon dataset (Table 3.41), TSVM is outperformed by both SGT and TLM. Comparing TLM to SGT, SGT provides better results. We hypothesize that in this case the TLM prior is not strong enough to improve parameter estimation.

The results for the Wikipedia dataset are shown in Table 3.42. For very small training data TLM is slightly better than SGT, but for larger training the SGT results become a bit better than those of TLM. For the latter two datasets, using background data in order to learn a strong prior on the TLM parameters is therefore an important aspect for making TLM a robust classifier. Additionally, the SGT hyperparameter is tuned directly on the test data. Therefore one of its main assumptions, that the training and test distributions are similar, does not have a strong impact on its performance. Thus, the results we present should be interpreted as an upper bound on

Table 3.41: **Amazon.** Micro-averaged F1 for different training set sizes

F1	Training (%)	NB	ILM	ISVM	TSVM	SGT	TLM
Micro-avg F1	14 (0.25%)	57.26	56.46	54.42	61.28	74.78	67.18
	28 (0.5%)	65.36	65.64	64.44	69.08	83.66	73.38
	56 (1%)	73.40	72.66	74.56	72.98	86.00	77.28
	112 (2%)	78.98	78.32	76.40	77.82	86.30	80.78
	280 (5%)	82.50	82.44	83.78	84.24	88.14	83.78
	560 (10%)	84.14	84.12	81.70	86.70	88.32	85.08
Macro-avg F1	14 (0.25%)	57.18	56.50	49.70	57.96	74.00	65.26
	28 (0.5%)	63.34	63.52	63.86	64.96	82.48	71.20
	56 (1%)	72.54	71.80	72.80	71.20	84.96	76.24
	112 (2%)	77.72	76.90	74.82	76.90	85.48	79.38
	280 (5%)	81.64	81.38	82.54	83.94	87.34	82.74
	560 (10%)	83.00	82.92	80.26	85.94	87.56	83.92

Table 3.42: **Wikipedia.** Micro-averaged F_1 for different training set sizes

F1	Training	NB	ILM	ISVM	TSVM	SGT	TLM
Micro-avg F1	14 (0.25%)	53.36	53.58	29.64	52.96	67.30	67.62
	27 (0.5%)	61.74	62.00	53.62	61.06	73.98	71.70
	53 (1%)	70.38	71.42	70.72	74.06	80.74	77.84
	108 (2%)	76.80	77.14	78.82	71.72	81.88	80.48
	265 (5%)	82.54	83.00	83.54	79.56	83.64	83.34
	530 (10%)	85.32	85.56	85.68	82.80	84.44	85.42
Macro-avg F1	14 (0.25%)	50.78	51.64	38.44	54.40	67.36	65.46
	27 (0.5%)	59.64	59.80	53.68	56.38	70.40	68.46
	53 (1%)	68.52	69.12	68.96	71.22	79.94	76.44
	108 (2%)	75.56	75.30	76.94	68.82	80.92	78.44
	265 (5%)	80.46	81.02	81.66	78.04	82.60	81.76
	530 (10%)	83.60	83.90	83.88	81.12	83.38	83.82

the SGT true classification performance. In practice, when training data is very sparse, tuning parameters can become infeasible and therefore SGT's performance could be affected considerably. For example, by tuning the parameters on the test data rather than using default values, the SGT results on small training sets improve by as much as 6-7%.

Results using Wikipedia background data

For the second experimental setting, in which we use Wikipedia pages to extend the topics descriptions, we show results in Table 3.43, Table 3.44 and Table 3.45. As we can observe from these experiments, the Wikipedia background data helps all methods, and in particular improves the

Table 3.43: **Reuters21578-wikiext.** Micro/macro-averaged F1 for different training set sizes

F1	Training (%)	NB	ILM	ISVM	TSVM	SGT	TLM
Micro-avg F1	33 (0.4%)	63.88	71.44	66.08	46.88	76.64	80.66
	40 (0.5%)	76.46	79.20	71.06	75.46	85.72	85.88
	80 (1%)	83.38	83.50	75.64	85.08	89.34	87.38
	160 (2%)	85.46	85.70	69.52	90.16	90.30	88.72
	400 (5%)	89.60	90.02	85.66	90.72	91.86	91.26
	800 (10%)	91.38	91.44	78.90	93.66	92.74	92.18
Macro-avg F1	33 (0.4%)	64.88	62.14	66.38	40.60	73.04	68.10
	40 (0.5%)	64.60	61.82	51.14	49.96	74.48	65.28
	80 (1%)	67.40	65.54	55.72	64.62	79.54	67.94
	160 (2%)	69.48	70.18	48.16	75.90	90.30	72.76
	400 (5%)	78.44	80.42	73.46	82.28	84.98	82.02
	800 (10%)	82.40	84.36	66.76	87.52	86.98	84.82

Table 3.44: **Amazon-wikiext.** Micro-averaged F1 for different training set sizes

F1	Training (%)	NB	ILM	ISVM	TSVM	SGT	TLM
Micro-avg F1	14 (0.25%)	77.30	74.92	66.42	65.42	78.04	82.80
	28 (0.5%)	79.18	77.66	72.40	70.52	83.76	83.16
	56 (1%)	81.12	80.36	77.04	69.98	85.32	83.12
	112 (2%)	83.34	83.08	77.56	79.14	86.42	85.18
	280 (5%)	84.30	84.68	83.06	83.38	87.74	85.78
	560 (10%)	85.36	85.78	81.68	85.18	88.32	86.46
Macro-avg F1	14 (0.25%)	76.96	75.06	66.38	63.06	76.78	82.04
	28 (0.5%)	78.28	77.02	71.24	67.34	82.58	82.22
	56 (1%)	80.42	79.74	75.54	68.24	84.30	82.32
	112 (2%)	82.32	82.06	76.14	78.40	85.56	84.14
	280 (5%)	83.50	83.80	81.76	83.18	86.76	84.88
	560 (10%)	84.36	84.82	80.18	84.42	87.48	85.48

results of TLM, which can benefit from the strong prior on its model parameters.

On all three collections TLM and SGT are the best performing methods. TLM outperforms SGT for the small training sets on all three datasets. For larger training data, the two methods are comparable.

Directly adding external data to the training can have a negative impact on the classification accuracy, since the original training data and the external background knowledge may have different vocabulary and different term distributions per topic. With ILM/TLM we can consider the external knowledge only for the computation of the similarity-prior on model parameters, and thus overcome a potential negative effect potentially induced by directly extending the training

Table 3.45: **Wikipedia-wikiext.** Micro-averaged F_1 for different training set sizes

F1	Training	NB	ILM	ISVM	TSVM	SGT	TLM
Micro-avg F1	14 (0.25%)	72.76	72.74	51.98	62.70	78.26	80.70
	27 (0.5%)	75.94	76.04	64.32	64.96	78.26	81.36
	53 (1%)	77.96	77.88	71.18	74.46	82.46	81.34
	108 (2%)	80.90	81.08	70.84	71.78	82.22	82.82
	265 (5%)	83.74	83.76	81.34	79.22	83.90	83.74
	530 (10%)	85.90	85.48	83.40	82.82	84.88	85.46
Macro-avg F1	14 (0.25%)	73.08	73.04	59.50	62.42	78.16	79.70
	27 (0.5%)	75.36	75.24	65.66	61.08	77.18	80.16
	53 (1%)	76.96	76.62	71.60	72.08	81.46	80.06
	108 (2%)	79.80	79.74	72.40	69.46	81.24	81.66
	265 (5%)	81.92	82.02	80.14	77.74	82.76	82.42
	530 (10%)	84.20	83.96	81.84	81.08	83.64	83.88

data. We observed empirically that using the Wikipedia extension only for computing the prior for ILM/TLM gave similar results as when adding it directly to the training data. TLM has the advantage of interpretability, since the mapping of features to concepts and concepts to topics can be directly displayed to the user. Furthermore, techniques such as re-training which we employ in the next section, can help improve TLM results by a significant percent.

Running time. We give the training running times for all methods on the data used for tuning hyperparameters, i.e. a single 1%-99% training/test split for each collection. Table 3.46 summarizes the training running time for all the methods compared. NB and ISVM have very low training times, followed by SGT. TSVM takes a considerable amount of time, in particular the choice of the C hyperparameter can considerably affect the training time. We show here the training times for the best value of the hyperparameters on the test set.

ILM needs around 10 minutes for training, while TLM requires somewhat longer (40 minutes) due to the estimation of the prior on the unlabeled part of the collection. During training, selecting the concepts and estimating the prior on $P[f|c]$ and $P[c|t]$ takes about three quarters of the training

Table 3.46: Total training running time for all methods compared (minutes).

	NB	ILM	ISVM	TSVM	SGT	TLM
Reuters21578	0.5	4	1.5	60	3.5	35
Amazon	0.5	3	1	65	3	20
Wikipedia	0.5	8	7	150	6	35
Reuters21578-wikiext	0.6	8	2	240	5	37
Amazon-wikiext	0.7	4	1	75	3	25
Wikipedia-wikiext	1	10	8	170	6.5	40

time, but this needs to be done only once before starting the EM algorithm. The rest of the training time goes into re-computing the parameters in the EM iterations.

Complexity. In this paragraph we discuss the time and memory complexity of each of the methods compared. NB is simple to implement and has time and memory complexity of $O(|F| \cdot |T|)$. The SVMLight implementation has time complexity of $O(q \cdot L \cdot f)$ for ISVM, where L is the number of labeled examples, q is a constant much smaller than L, and f is the maximum number of non-zero features in any of the training examples [Joa99a]. The memory complexity is $O(q \cdot L)$ [Joa99a].

TSVM has time complexity of $O(2^U)$ where U is the number of unlabeled examples [Joa99b]. SGT relies on finding minimum average cuts also called normalized cuts in the nearest neighbor graph of all documents in the dataset, which in turn relies on finding the second smallest eigen vector of the Laplacian graph. This can be done using the Lanczos method in $O(m \cdot n)$, where n is the number of nodes in the graph and m was empirically observed to be less than $O(n^{\frac{1}{2}})$ [SM00].

Our algorithm for learning ILM/TLM has $O(|F| \cdot |C| \cdot |T|)$ time and $O(|F| \cdot |C|) + O(|C| \cdot |T|)$ memory complexity, where $|F|$ is the number of features in the model, $|C|$ is the number of concepts and $|T|$ is the number of topics. These factors result from the fact that we need to keep track of the $P[f|c]$ and $P[c|t]$ parameters. Due to our concept selection strategy, the number of concepts is at most as high as the number of features, and we assume that the number of topics is a fairly small constant. This complexity can be quite high if we work with a large number of features, but as observed from our experiments using a few thousand features already leads to good classification results. All experiments were ran on a Red Hat Linux release 3.0 machine, with 8GByte memory and 2.3GHz AMD Opteron CPU.

3.5.8 Other Aplications: Topic-driven Clustering

Organizing large document collections into meaningful clusters has attracted a lot of interest from the research community. Clustering is usually an unsupervised process, but if prior knowledge about the clusters of interest is available, the clustering techniques should be able to exploit this information for producing more desired clusterings.

For example, domain experts or normal users could describe the major topics that the collection covers. Most importantly, they would like the clustering algorithms to produce clustering solutions that are consistent with their cognition models [ZK05]. For example, the user may want to specify the granularity level of the clusters, together with a short description for each cluster, and would expect the clustering algorithm to find groups of documents according to the specified criteria.

Depending on the application at hand, providing labeled samples for each of the topics of interest could be very costly, particularly if the number of desired topics/clusters is large. Instead, short keyword descriptions could be provided for the topics of interest. Since the available descriptions may only contain a few keywords, in order to produce good clusterings additional type of knowledge must be considered by the algorithm, e.g. the unlabeled documents or other sources of domain knowledge.

The problem of organizing document collections into clusters according to a user-given set of categories was coined by [ZK05] as *topic-driven clustering*.

We view this problem as a transductive classification problem, where we assume that each topic/category has a short textual description which is treated as explicitly labeled training data.

Table 3.47: Details on the training/test per collection.

Collection	Topics	Source	Documents	Terms	Categories
Trec6	301-350	FT & LATimes	2,559	37,821	37
Trec7	351-400	FT & LATimes	2,785	39,572	45
Trec8	401-450	FT & LATimes	2,745	41,019	43

The goal is to organize the unlabeled collection according to the user-defined topics.

Test Collections

We work with three document collections selected from the TREC-6, TREC-7, and TREC-8 ad-hoc retrieval tasks which were also used in [ZK05]. These are news articles from the Financial Times Limited and Los Angeles Times newswires. The user specified topics for these datasets are taken to be the TREC queries for which at least 10 relevant documents are available. Each query has a title and a short textual description. We only use the query title as a topic description, thus each topic of interest is described by a few keywords. The number of queries/topics varies from 37 for the TREC6 dataset to 45 for the TREC8 dataset. The number of documents is close to 3,000 for each collection. Table 3.47 gives exact statistics on these datasets. All three collections are available online.[2]

Results

We compare the performance of several inductive and transductive classifiers on this task. The methods compared are: Naive Bayes, Support Vector Machines, Spectral Graph Transducer and our Transductive Latent Model. We observed empirically that on this task Transductive Support Vector Machines (TSVM) performed worse than Inductive Support Vector Machines (ISVM). For this reason, we only show results for ISVM. We as well present results using one of the techniques presented in [ZK05], e.g. a modified form of k-means which considers the constraints imposed by the user while searching for clusters. We use micro/macro-averaged F1 for evaluating the classification results.

Since the training data is very sparse the prediction quality can be quite low. For this reason we decided to investigate whether re-training (aka. self-training [Zhu05]) of all the classifiers can help improve the classification quality. During re-training each classifier selects some of the documents for which it predicted labels with high confidence, and adds those documents to the training set. Then the classifier is re-trained on the extended training set. This process needs to be carefully designed, since extending the training with wrongly classified documents can increase rather than decrease the quality of the classifier's predictions. For re-training we considered several strategies for selecting documents for iteratively extending the training set. We observed empirically that sorting the documents by prediction confidence and extending the training by taking the top 20%, gave the best results. We thus use this simple technique for retraining all the classifiers compared.

[2]Test collections available at `http://www.mpi-sb.mpg.de/∼ifrim/trec678-datasets.zip`

Table 3.48: TREC678. Micro/macro-averaged F_1 for different training set sizes.

F1	Collection	NB	Re-NB	ISVM	Re-ISVM	SGT	Re-SGT	TLM	Re-TLM
Micro-avg F1	Trec6	56.5	58.2	64.2	64.2	68.7	68.7	76.1	78.0
	Trec7	43.6	44.0	51.4	55.6	63.1	63.1	53.9	58.0
	Trec8	52.1	55.6	58.9	65.0	57.2	57.2	67.6	74.5
Macro-avg F1	Trec6	N/A	N/A	63.0	63.0	N/A	N/A	69.5	73.4
	Trec7	53.3	N/A	59.5	61.7	N/A	N/A	58.5	63.8
	Trec8	54.4	N/A	59.4	61.5	N/A	N/A	66.9	73.5

Table 3.48 presents results for the classifier methods compared. For TLM we set the number of features to 10,000 and the number of EM iterations to 1 across collections. For all the other compared methods we set their hyperparameters based on the *performance on the test set.* These results are therefore an upper bound on the classification performance in a realistic scenario. We use the prefix *RE* in front of the name of each classifier to denote the re-trained classifier. If re-training did not improve results we show the result without re-training. For the macro-averaged F1 results, we denote by N/A the case in which one or more topics were not discovered by the classifier, i.e., the classifier did not correctly identify any test examples belonging to one or more topics. We first note that the inductive classifiers NB and ISVM give fairly low results, since the training data is extremely sparse, thus the classifiers need to use additional data to relax the constraint impose by the training data. Re-training these two classifiers improved the results slightly (by 2-3% micro-averaged F1), but did not outperform the results of the transductive classifiers. The transductive classifiers can benefit from exploring the unlabeled data in order to learn more about the structure of the dataset. Comparing the results of the two transductive classifiers (SGT and TLM), TLM outperformed SGT on two out of three datasets. Re-training improves the results of TLM but did not improve the classification results of SGT.

Table 3.49: TREC6. Micro/macro-averaged F_1 for different training set sizes.

F1	Collection	Constrained-KMeans
Micro-avg F1	Trec6	77.7
	Trec7	62.4
	Trec8	73.7
Macro-avg F1	Trec6	73.2
	Trec7	72.0
	Trec8	67.8

Table 3.49 shows results using the constrained-k-means method presented in [ZK05]. Compared to the constrained clustering approach, TLM was better in two out of the three test collections.

Given that training data is so sparse for this particular problem, TLM relied mainly on the context-similarity prior on the model parameters, rather than the explicit training information in order to learn a robust model. Nevertheless, the constrained clustering technique performs quite well and may be better suited for such problems where the explicitly labeled training data is in the form of a few keywords rather than training samples.

3.6 Conclusion

In this chapter we introduced a new generative model for text documents, which uses various building blocks designed to facilitate the integration of background knowledge about the problem at hand in the process of learning from small training data.

We proposed using external ontologies for instantiating the structure of the latent model, rather than selecting an appropriate structure by time consuming model selection strategies. Such ontologies are currently growing and are freely available [SKW07, WHW08]. Furthermore, information extraction techniques have considerably advanced [SIW06, WW08, WHW08], thus automatically building domain-specific ontologies has become a fruitful area.

We have given an Expectation-Maximization algorithm for learning the parameters of our Inductive/Transductive Latent Model. The parameter space can be huge, but we propose several techniques for learning a prior on the model parameters based on background knowledge. For example, if training is too sparse for learning robust parameter values, a prior which relies on context-similarity in the given corpus and external sources, can help improve the final predictions by 10%. Additionally, background knowledge, e.g., encyclopedia such as Wikipedia, can be used to explicitly or implicitly extend the topic descriptions provided by the training set.

We have empirically observed that the additional flexibility of ILM/TLM offered by its various building blocks results in improved classification results for small training sets, as compared to other state-of-the-art classifiers. We have analyzed various ways of setting parameters, and what type of effect each building block has on the overall model performance. We have seen that different model components can be turned on or off depending on how much we trust the training data or the background knowledge to be useful for learning an accurate classifier. Most of all, we have observed that taking advantage of unlabeled data which is plentiful in many applications improves the results of our model.

In the next chapter we study another way of approaching the problem of learning with sparse labeled data. Rather than using a richer model and additional background knowledge, we use a simpler logistic regression model and focus on learning with more complex features, such as word or character *sequences*. Thus, we focus on more complex representations of the input data, rather than using external information sources. We show that for applications in which the individual keywords are not highly correlated with the pre-specified categories, learning sequences rather than individual words can result in increased classification accuracy.

Chapter 4

Rich Input Representations: Learning with Variable-Length N-gram Features

4.1 Introduction

Approaching text categorization from a Machine Learning perspective involves several typical steps. Given a training set of labeled samples, feature selection is applied in order to reduce the size of the feature space or remove noisy features. A classifier, typically a parametric statistical model, is learned from the labeled samples represented in the reduced feature space. Finally, the induced classifier is used for labeling new test samples. Typically both the training and test samples are represented as a set-of-words or a bag-of-words. This representation has the advantage of being compact, since it only requires statistics about the distinct words in the collection, but has the great disadvantage of sometimes not being expressive enough. The decrease in expressivity is a result of discarding the structure, order, and context of the original documents' contents, which comes at a considerable loss of information, in favor of having a simple and compact representation. Since the bag-of-words representation is the most widely used for text classification, this could mean that in general such loss of information does not negatively affect the classification results. Therefore, is there a need for using richer representations of the input text, when the simple bag-of-words representation seems to be enough? For answering this question, consider the following example.

Let us assume that a company wants to automatically categorize its customer feedback, which comes in the form of user product reviews, into positive or negative reviews. For example, given the two labeled samples in Table 4.1 for the positive and the negative category, we would like the classifier to learn discriminative features, such as *not bad, not good, actually quite good* and *actually quite bad* which would be good discriminators for the positive versus the negative category. A bag-of-words representation of these input samples would only capture the individual words, out of context, such as *good, bad, actually*, etc., completely missing out on the difference between the two statements. Thus a simple swapping of the words *good* and *bad* causes this representation to even

Table 4.1: Bag-of-words fails to accurately represent the input data.

Category	Training sample
+1	This product is not **bad**, it is actually quite **good**.
−1	This product is not **good**, it is actually quite **bad**.

Table 4.2: Bag-of-words representation for the two training samples.

Category	Training sample
+1	(actually, bad, is, it, good, product, This)
−1	(actually, bad, is, it, good, product, This)

fail to accurately represent the input. The bag-of-words representation for both training samples is given in Table 4.2.

Ideally we should not alter the original input text, but work with the original *text sequence* and use a representation that can take advantage of all subsequences (e.g. n-grams) in the text, as features. The n-gram representation for the two samples above is shown in Table 4.3. Nevertheless, working with the full space of n-gram features is a highly challenging problem, due to the huge size of such a feature space, e.g. with u distinct unigrams in the input training corpus, the n-gram feature space can have as many as $O(u^n)$ distinct n-grams.

Table 4.3: N-gram representation for the two training samples.

Category	Training sample
+1	(This, product, is, not, bad, it, is, actually, quite, good, This product, product is, is not, **not bad**, ..., **quite good**, This product is, product is not, is not bad, ..., it is actually quite good,...)
−1	(This, product, is, not, good, it, is, actually, quite, bad, This product, product is, is not, **not good**, ..., **quite bad**, This product is, product is not, is not good, ..., it is actually quite bad, ...)

Since the bag-of-words representation is quite compact, we could try to fix its expressiveness handicap by simply adding some n-gram features for a fixed n, or adding some combinations of bigram and trigram features, thus essentially resorting to some type of feature engineering or feature selection. But, unless done by a domain expert or by a user with considerable knowledge of the task and corpus at hand, this type of pre-processing heuristics may still miss important information. Furthermore, feature selection is often highly dependent on several other factors: the specific application, the domain of the text (topicality, using common or scientific terms), the language of the corpus (Western versus Asian language), and even the size of the training corpus (when the training corpus increases, the choice and the number of features may be very different, e.g. selecting unigrams and bigrams versus unigrams, bigrams and trigrams, selecting 1,000 features versus 5,000 features, etc.).

In this chapter we present a new logistic regression algorithm, coined **Structured Logistic Regression** (or **SLR**), for efficiently learning a classifier in the space of all n-gram features in the training set. The **SLR** algorithm selects only a small set of discriminative n-gram features from the full space of n-grams, and produces features such as *not bad, not good, actually quite good* which can easily discriminate the examples presented in Table 4.1.

The main idea behind our algorithm is to modify the gradient based optimization techniques typically used for solving logistic regression (which are at least linear in the feature space size), so they do not require seeing the full feature space for updating the model parameters. Instead of iteratively re-computing the full gradient vector of the log-likelihood function, we use a different ascent direction which is extremely sparse, i.e. a vector with only one non-zero coordinate corresponding to the the n-gram feature with the largest magnitude of the gradient value. This approach works for the following reasons. First, the chosen direction is an *ascent direction* because it forms an angle of less than 90 degrees with the original gradient vector, therefore theoretically guaranteeing ascent in the objective function [NW06]. Second, choosing the n-gram feature with the largest gradient value reflects a greedy advance toward the largest change in the objective function (conditional log-likelihood); thus it captures a notion of goodness of the n-gram feature for the discriminative power of the model.

This transforms the learning problem into a search problem, which can be solved efficiently by pruning large parts of the search space, using an upper bound on the quality (i.e. gradient value) of each n-gram feature, based on its prefix. The upper bound exploits the structure of the n-gram feature space (i.e. the fact that the set of occurrences of an n-gram is a subset of the set of occurrences of its prefix) in order to quantify the growing behavior of the gradient value of any n-gram feature based on its prefix.

The advantage of such a technique is that we can treat the input simply as a sequence of words or characters, or even more general, as a *sequence of bits*, and let the model learn a compact set of discriminative subsequences. Thus, model learning and feature selection are interleaved, eliminating the need for expert knowledge about the classification problem at hand. Furthermore, the tokenization of the text is not fixed in advance, as in the bag-of-words model, but it is rather learned by the logistic regression model. This has the advantage of providing robustness to sparse training data, since arbitrary length sequences of characters are considered as features, rather than restricting the input representation to the level of words. In the next sections we discuss the proposed model in detail and show that it is competitive to state-of-the-art classification techniques in terms of prediction quality, while being at least one order of magnitude faster than its competitors.

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 (e.g. logistic regression), without any noticeable penalty on the size of the feature space and computational cost of the training.

4.2 Structured Logistic Regression

4.2.1 Logistic Regression Model

Let $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ 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 $x_i = (x_{i1}, \ldots x_{ij}, \ldots x_{id})^T,\ x_{ij} \in \{0, 1\},\ i = 1 \ldots N.$ $y_i \in \{0, 1\}$ are class labels encoding membership (1) or non-membership (0) of the training samples in the category. We focus here on binary classification and treat multi-class classification as several binary classification problems. Let X be the set of all samples $x \in \{0, 1\}^d$ in the given collection. Let $\beta = (\beta_1, \ldots, \beta_j, \ldots, \beta_d) \in \mathbb{R}^d$ be a parameter vector (which can be interpreted as a weight vector for the feature dimensions). Under the logistic regression model, the probability of a sample belonging to class $y = 1$ is ([HTF03]):

$$p(y_i = 1; x_i, \beta) = \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}}.$$

The goal is to learn a mapping $f : X \to \{0, 1\}$ 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 a mapping for logistic regression is achieved by finding the parameter vector β, that maximizes the conditional log-likelihood of the training set. The conditional log-likelihood of the training set for the logistic regression model is ([HTF03]):

$$l(\beta) = \sum_{i=1}^{N} [y_i \cdot \beta^T \cdot x_i - log(1 + e^{\beta^T \cdot x_i})] \tag{4.1}$$

We take an optimization approach for solving this problem. In the next section 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.

4.2.2 Coordinate-wise gradient ascent in the space of all n-gram sequences

All prior optimization methods for solving logistic regression require at least $O(d)$ memory where d is the size of the overall feature space. Since with u distinct unigrams there are $O(u^n)$ potential n-grams in the training set, for large u and n, this would result in 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).

The simplest optimization approach to solving logistic regression is the steepest (gradient) ascent technique [NW06]. It is linear in the number of features and it iteratively updates the parameter vector β by always moving in the direction of the gradient of the log-likelihood function.

Equation 4.2 gives the parameter update details.

$$\begin{aligned}\beta^{(n+1)} &= \beta^{(n)} + \epsilon \cdot \frac{\partial l}{\partial \beta}(\beta^{(n)}) \qquad (4.2)\\ (\beta_1^{(n+1)}, \beta_2^{(n+1)}, \dots, \beta_d^{(n+1)}) &= (\beta_1^{(n)}, \beta_2^{(n)}, \dots, \beta_d^{(n)})\\ &+ \epsilon \cdot [\frac{\partial l}{\partial \beta_1}(\beta^{(n)}), \frac{\partial l}{\partial \beta_2}(\beta^{(n)}), \dots, \frac{\partial l}{\partial \beta_d}(\beta^{(n)})]\end{aligned}$$

This essentially involves iteratively recomputing the full gradient vector, which has as many coordinates as features in the feature space. For the extremely large space of all n-grams, simply enumerating all the coordinates of the gradient vector is infeasible. Therefore, rather than computing the full gradient vector (Equation 4.3) we propose taking a different update direction, which is still an ascent direction, but is much sparser than the original gradient (Equation 4.4).

$$\frac{\partial l}{\partial \beta}(\beta^n) = [\frac{\partial l}{\partial \beta_1}(\beta^n), \frac{\partial l}{\partial \beta_2}(\beta^n), \dots, \frac{\partial l}{\partial \beta_d}(\beta^n)] \qquad (4.3)$$

$$\left[\frac{\partial l}{\partial \beta}(\beta^n)\right]^{sparse} = [0, 0, \dots, \frac{\partial l}{\partial \beta_j}(\beta^n), \dots, 0] \qquad (4.4)$$

$$where,\ \beta_j = \underset{\beta_j, j \in \{1,\dots,d\}}{argmax} \frac{\partial l}{\partial \beta_j}(\beta^n)$$

This step transforms the learning problem into a search problem, since in each iteration we have to search for the n-gram feature with the largest magnitude of the gradient. This problem is still hard, given the huge space that we need to search to find this feature, but can be made easier by giving a search-space-pruning-strategy. In our approach, we iteratively search for the best n-gram feature and prune parts of the search space which cannot improve the currently best solution. For designing an efficient pruning strategy we exploit the *structure of the n-gram feature space*: the fact that the set of occurrences of an n-gram is a subset of the set of occurrences of its prefix. This consideration leads us to an upper bound on the gradient value of any n-gram feature, based on the occurrences of its length-$(n-1)$-prefix.

Using Equation 4.1, the gradient of l with respect to a coordinate value β_j evaluated at a given parameter vector β is:

$$\frac{\partial l}{\partial \beta_j}(\beta) = \sum_{i=1}^{N} x_{ij} \cdot \left(y_i - \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}}\right) \qquad (4.5)$$

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

The following theorem, inspired by work on boosting [KM04], gives a convenient way of computing an upper bound on the gradient value for any super sequence $s_{j'} \supseteq s_j$.

Theorem 4.2.1 *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*

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

Proof *We split the analysis into two subproblems, the first concerning the "positive" class ($y = 1$), and the second concerning the "negative" class ($y = 0$). First we prove the bound for the positive class:*

$$\frac{\partial l}{\partial\beta_{j'}}(\beta) = \sum_{i=1}^{N} x_{ij'}\cdot\left(y_i-\frac{e^{\beta^T\cdot x_i}}{1+e^{\beta^T\cdot x_i}}\right) \tag{4.6}$$

$$= \sum_{\{i|s_{j'}\in x_i\}} x_{ij'}\cdot\left(y_i-\frac{e^{\beta^T\cdot x_i}}{1+e^{\beta^T\cdot x_i}}\right) \tag{4.7}$$

$$\leq \sum_{\{i|y_i=1,s_{j'}\in x_i\}} x_{ij'}\cdot\left(1-\frac{e^{\beta^T\cdot x_i}}{1+e^{\beta^T\cdot x_i}}\right) \tag{4.8}$$

$$\leq \sum_{\{i|y_i=1,s_j\in x_i\}} x_{ij}\cdot\left(1-\frac{e^{\beta^T\cdot x_i}}{1+e^{\beta^T\cdot x_i}}\right). \tag{4.9}$$

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

$$\frac{\partial l}{\partial\beta_{j'}}(\beta) \geq \sum_{\{i|y_i=0,s_j\in x_i\}} x_{ij}\cdot\left(-\frac{e^{\beta^T\cdot x_i}}{1+e^{\beta^T\cdot x_i}}\right). \tag{4.10}$$

Thus we have:

$$\begin{aligned}\sum_{\{i|y_i=0,s_j\in x_i\}} x_{ij}\cdot\left(-\frac{e^{\beta^T\cdot x_i}}{1+e^{\beta^T\cdot x_i}}\right) &\leq \frac{\partial l}{\partial\beta_{j'}}(\beta) \\ \leq \sum_{\{i|y_i=1,s_j\in x_i\}} x_{ij}\cdot\left(1-\frac{e^{\beta^T\cdot x_i}}{1+e^{\beta^T\cdot x_i}}\right)\end{aligned} \tag{4.11}$$

The absolute value of the gradient of $l(\beta)$ at coordinate j' corresponding to n-gram sequence $s_{j'}$

is thus bounded by $\mu(\beta_j)$:

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

□

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 give pseudo-code for our algorithm in the next subsection.

Once we find the feature with the best gradient value, we adjust the value of the relevant parameter:

$$\beta_j^{(n+1)} = \beta_j^{(n)} + \epsilon \cdot \frac{\partial l}{\partial \beta_j}(\beta^{(n)})$$

and repeat the search for the coordinate with maximum magnitude of the gradient value. This essentially produces one candidate feature per iteration. ϵ is known in the literature as the *step length*, and is usually estimated via line search algorithms [NW06]. Finding a good step length is necessary for guaranteeing convergence of the gradient based algorithm. The best direction to follow for maximizing the log-likelihood function is given by the gradient. The distance to move along this direction is given by the line search algorithm which subsequently generates a limited number of trial step lengths until it finds one that loosely approximates the maximum of the objective function.

The iterations typically start at $\beta^0 = 0$ ([HTF03]). Note that, since each restricted gradient direction is not necessarily 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 β, which is a linear model learned in the space of all n-gram sequences.

4.2.3 Algorithm

A high level overview of our gradient-based search algorithm is shown in Algorithm 1. The algorithm calls several routines. The function **find_best_ngram()**, returns the n-gram with the highest magnitude of the gradient value in a given iteration. Once the best n-gram feature is returned, we do a line search (routine **find_best_step_size()**) for computing the best step size to follow in this direction. Next we need to update the relevant parameter, i.e. the weight of the corresponding best n-gram feature. This is done calling the **update_best_ngram_parameter()** procedure. The best n-gram feature is then added to the SLR model (**add_best_ngram_to_SLR_model()**). The

Algorithm 1 Structured Logistic Regression algorithm.

```
 1: Input: Training set $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$,
    where $x_i$ is a training document, $y_i \in \{0, 1\}$ is a class label
 2: Output: Structured Logistic Regression model.
 3: begin
 4:     $iter = 0$   // optimization iteration
 5:     //for each optimization iteration
        while $iter < num\_iterations$ || $convergence\_test$
 6:         find_best_ngram()
 7:         find_step_size()
 8:         update_best_ngram_parameter()
 9:         add_best_ngram_to_SLR_model()
10:         update_log_likelihood()
11:         convergence_test = check_convergence_test()
12:         $iter = iter + 1$
13:     end
14: end
```

final steps are those of updating the log-likelihood objective function (**update_log_likelihood()**) and checking convergence in order to decide whether the objective function increase falls below a given threshold and therefore we can stop the learning process (**check_convergence_test()**).

In Algorithm 2 we give detailed pseudo-code for the routine of finding the best n-gram feature. The gradient value in iteration n is always computed at the parameter vector estimated in iteration $n-1$, thus the selection of a new feature depends on the set of previously chosen features. The main parameter of SLR is the number of optimization iterations, which directly influences the size of the final model. This can be treated as an explicit input parameter for our algorithm or it can be implicitly decided by a convergence test, e.g. by thresholding the aggregated change in score predictions or by thresholding the improvement in the objective function (check_convergence_test()).

Theorem 4.2.1 proposes an efficient way for pruning the search space while searching for the best n-gram feature. Figure 4.1 shows the search and pruning process graphically. We first start by checking the gradient value of single unigrams. In Figure 4.1, we denote the unigrams by a, b, c, the gradient value by $grad$, the current best gradient value by δ and the upper bound on the gradient value of any n-gram in the space rooted at a given node in the search tree with μ.

For each unigram, we compute the upper bound on the gradient value of any n-gram starting with this prefix. If n-grams in this part of the search space can still improve the current best gradient value, we continue expanding and exploring this part of the space. If the upper bound given in Theorem 4.2.1 is below the current best gradient value, this means we can discard this part of the search space, since no n-gram in this part of the space can improve the current best gradient value. 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 4.2.1 prunes more than 90% of the search space.

Algorithm 2 Find best n-gram feature.

```
1:  **Input:** Training set $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$,
    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:      **function find_best_ngram()**
7:          //for each single unigram
            **foreach** $s' \in \bigcup_{i=1}^{N} \{s | s \in x_i, |s| = 1\}$
8:              **grow_sequence**$(s')$
9:          **end**
10:         **return**best_feature
11:     **end**
12: **end**
```

```
1:  **function grow_sequence**(s)
2:      **if** $\mu(s) \leq \tau$ **then return**   //$\mu(s)$ as in Theorem 1
3:      **if** $abs(gradient(s)) > \tau$ **then**
4:          best_feature = s   //suboptimal solution
            $\tau = abs(gradient(s))$
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**
```

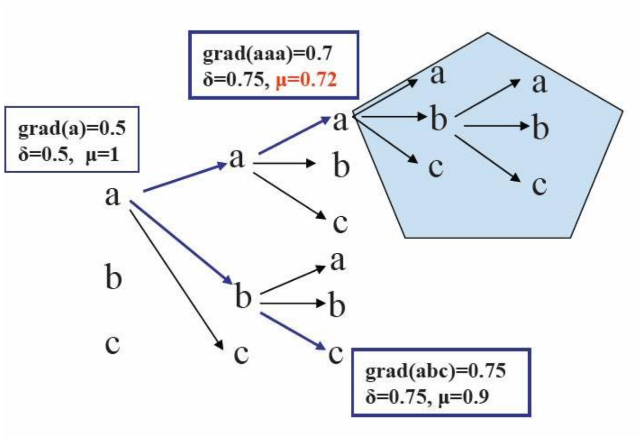

Figure 4.1: Graphical description of the find-best-n-gram-feature algorithm.

4.2.4 Implementation Issues

In this section we discuss several issues related to implementation, in particular we focus on the routines and parameters of our learning algorithm.

Input Representation and Data Structures.

The first aspect of learning in such a large feature space regards data representation. We simply represent the data collection as a vector of strings, where a string is a document, and decide on-the-fly on the type of tokenization required. The user can set an input parameter to specify whether she desires word-level n-grams or character-level n-grams. Furthermore, since we need to keep track of the occurrences of all distinct unigrams and of the features of interest (e.g. best n-grams selected during learning iterations), we decided to keep an inverted index which grows on-demand. In each iteration, we start with an inverted index on all distinct unigrams in the training set, which we subsequently extend to include candidate n-grams which could not be pruned using the bound of Theorem 4.2.1. The inverted index does not grow excessively due to the effectiveness of the pruning bound.

Search Space Traversal and Upper Bounds.

Traversing the search space can be done in a Breadth-First-Search (BFS) fashion or a Depth-First-Search (DFS) fashion. The BFS style traversal first extends all unigrams to bigrams, then all bigrams to trigrams, etc. The DFS traversal first extends a unigram up to its longest sequence which cannot be pruned, and then moves on to extending the next unigram. We have investigated both approaches in our work. For word-level n-grams, longer n-grams are less likely to be better for classification than shorter n-grams; therefore the BFS traversal can prune many potential n-gram candidates very fast. For character-level n-grams, longer sequences are more likely to be beneficial for classification; therefore DFS is more beneficial for the running time of the algorithm, since the pruning bound will faster eliminate large parts of the search space. Depending on the application and the type of n-grams employed, one type of traversal may be preferred over the other or one could use a combination where DFS traversal and BFS traversal steps are alternated.

Line Search Algorithm

We mentioned in the description of our algorithm that we use a line search routine in order to find the best step size in the direction of the selected n-gram feature. Finding a good step size is necessary for steepest ascent style of algorithms in order to guarantee convergence. We use a simple binary line search algorithm in which we start at some initial small value for the step size, and keep on doubling the step size until the log-likelihood function stops increasing. We then narrow down the interval found this way until it becomes a good approximation for the step size. Unless the starting value in the binary line search is small enough, the increase in the log-likelihood may stop right at the start of the step size search. Therefore we need a robust and ideally automated way to estimate such a starting value. We have found out empirically that the starting value can automatically be set based on the magnitude of the gradient value of each individual feature.

With line search, no predetermined step size is used. Here, we sketch the steps of this algorithm in more detail. Let d be the search direction starting from the point β. We evaluate our objective function (log-likelihood l) at a sequence of points $\beta_0 = \beta$ and, recursively, $\beta_{n+1} = \beta_n + 2^n \cdot \epsilon \cdot d$ (doubling the step ϵ in each step). We assume that the direction d and the parameter ϵ are such that $l(\beta_1) > l(\beta_0)$ (e.g., the direction is the gradient and ϵ is very small). The move continues from point to point until $l(\beta_{n+1}) \leq l(\beta_n)$. Now β_{n-1}, β_n, and β_{n+1} are taken as the initial range where the maximum occurs. The range is then narrowed down iteratively until it is so small that one can simply take the middle point as a good approximation of the maximum.

Regularization of the Objective Function

When solving logistic regression we typically use explicit regularization on the log-likelihood function. This plays the role of avoiding degenerated solutions and can also considerably shrink the feature space. For example, if we use an L1 regularizer

$$\max_{\beta} \; l(\beta) - \lambda \cdot \sum_{j=1}^{d} |\beta_j|$$

decreasing the regularizer coefficient λ has the effect of dropping many feature weights to zero. In our case we do not use an explicit regularizer on the log-likelihood function, but rather achieve this effect implicitly by our convergence threshold. The higher the convergence threshold, the fewer learning iterations we will allow, and therefore the final model will contain fewer distinct features. Thus rather than starting with the full space of feature and dropping some to zero using the explicit regularizer, we instead add one feature in each learning iteration. In Section 4.5 we investigate the effect of using an explicit regularizer on the objective function of SLR.

Implementation Modules and Parameters

We have implemented different modules for training and testing the classifier. Open source code for *SLR* is available on-line at: `http://www.mpi-inf.mpg.de/~ifrim/slr`. The training module has several parameters.

./slr_learn [-L max_ngram_size] [-T number_iterations] [-n token_type] [-c convergence_threshold] [-m minsup] [-v verbosity] training_file model_file

The default values and the explanation of the role of each parameter are:

- **L (max_ngram_size) = 0xffff.** This parameter allows restricting the n-grams considered by the learning algorithm to a given length. By default the n-gram size is unrestricted, e.g. at most as long as the largest document in the training set.

- **T (number_iterations) = 50,000.** This parameter sets the number of optimization iterations. By default it is set to 50,000 iterations or it can be indirectly set by the convergence threshold, i.e., once the convergence criterion is met, the iterations terminate.

- **n (token_type)= 0** (word token, default), or 1 (character token). The user can decide what type of n-grams he is interested in, by setting this parameter. A value of 0 triggers the algorithm to consider a word n-gram representation, while for a value of 1 the algorithm considers the input text as a sequence of characters.
- **c (convergence_threshold) = 0.005.** This parameter sets a threshold on the aggregated change in score predictions. If the weight vector does not change much, implicitly the log-likelihood does not increase much and we can decide to stop the learning algorithm.
- **m (min_support) = 1.** With this parameter we can control the minimum support (occurrence) in the input dataset an n-gram should have to be considered for learning. By default we consider all n-grams. A minimum support of k means that an n-gram should occur in at least k documents.
- **v (verbosity)= 1.** This decides the amount of information the algorithm should output, e.g., information on the log-likelihood value, the value of the gradient of the feature selected, etc.
- **training_file.** The training data file name.
- **model_file.** The output file for the learned model.

The module for building a final model creates a trie on the features of the model for the purpose of faster testing. The input parameter is the model file. The output parameters are a binary file storing the trie model, and optionally a file storing the distinct features of the model.

```
./slr_mkmodel -i model_file -o binary_trie_model -O distinct_rules_file
```

The testing module takes the (trie) model and the test data as inputs. Additionally an optional parameter specifying whether a given classification threshold should be used for prediction, rather than fixing the threshold at 0, e.g. a sample is classified as positive if its predicted score is larger than a certain threshold.

```
./slr_classify [-t classif_threshold] [-v verbosity_level] test_file binary_trie_model
```

4.3 Experimental Results

In this section we compare our learning technique to state-of-the-art logistic regression and support vector machines.

For logistic regression we use the open source implementation by Genkin et al. [GLM06] which we denote by BBR. This is a recent implementation of regularized (quasi-Newton) 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.

For support vector machines we used the latest open source SVM^{perf} solver by Joachims et al. [Joa06] which is especially tuned for linear problems. Additionally we show results with the latest tool for large scale linear classification with SVM, LIBLINEAR [HCL+08, LWK08, Lin08, FCH+08].

4.3.1 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, i.e., all the unigrams, bigrams and trigrams.

For BBR and SVM, we first use a state-of-the-art pattern mining tool [AKA$^+$02, Kud03, Zak02] in order to produce all the actually occurring 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-averaged and macro-averaged F1 measure [YL99], 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 our competitors.

4.3.2 Test Collections

We study three different applications[1] 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 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 [IW06]. 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 [HTT03]. 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**).

[1]All datasets are available at:
`http://www.mpi-inf.mpg.de/∼ifrim/data`

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.

4.3.3 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) parameters for each method. Parameter tuning is always carried out on the training set. The classifier sees the test set only in the final stage where the model learned on the full training using the best parameters from cross-validation and is tested on the held-out test set.

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 (i.e., the aggegration of $\beta^{new} \cdot x_i - \beta^{old} \cdot x_i$ for all documents x_i), 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 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.

BBR

Setting the regularization parameter. For Bayesian logistic regression [GLM06] the regularization parameter is the most important tuning aspect (i.e., the prior variance for parameter values). This is initially set to the value recommended in [GLM06], 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.

SVM^{perf}

Setting the soft-margin C parameter. For SVM^{perf} [Joa06] the soft-margin C parameter is the most important tuning aspect. [Joa06] 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.

LIBLINEAR (SVM)

Setting the cost C parameter. For LIBLINEAR [HCL$^+$08, LWK08, Lin08, FCH$^+$08] the hyper-parameter is the regularization or cost parameter C. We use the recommended default values [HCL$^+$08] for the fixed parameter experiments, and then we tune the C parameter to study the influence on classification accuracy. The default parameters were chosen by the authors of LIBLINEAR such that the best trade-off between running time and accuracy is achieved: $C = 1$, $\epsilon = 0.01$ and $s = 1$. This sets the objective function to L2-loss Support Vector Machines in the dual formulation which was empirically observed to behave best across several datasets. For tuning the C parameter we follow [HCL$^+$08, LWK08] and run cross-validation with C from the set $\{0.125, 0.25, 1, 4, 16\}$ and choose the parameter C which gives the best cross-validated classification performance. LIBLINEAR was designed to exploit the sparsity in the data for improving the scalability and the running time of linear SVM.

4.3.4 Results

All experiments were run on a Red Hat Linux release 3.0 machine with 8GB memory and 2.3GHz AMD Opteron CPU. We show micro/macro-averaged F1 [YL99] as global measure of quality for each dataset and classifier.

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 measure and the influence of the n-gram length on classification quality. Next, we discuss the running time of each of the methods compared.

In terms of micro/macro-averaged F1 (Table 4.4, Table 4.7), we observe that our method is as good as the state-of-the-art regarding generalization ability. For the case of default parameters, SLR is 3 to 5% better than either BBR, LIBLINEAR or SVM. For both word n-grams and character n-grams the macro-averaged F1 increases with increasing n-gram length. For word n-grams SLR's macro-averaged F1 goes from 72.95% for $n = 1$ to 73.28% for n unrestricted. For character n-grams, the macro-averaged F1 increases from 56.09% for $n = 1$ to 74.0% for n unrestricted.

In the case of tuned parameters, for word n-grams SLR achieves 74.04% macro-averaged F1, compared to BBR 73.94%, LIBLINEAR 71.32% and SVM^{perf} 71.24%. Increasing the n-gram length does not seem to improve results for the word representation with tuned parameters. For $n = 1$ the SLR macro-averaged F1 is 74.04%, while for n unrestricted it is 73.91%. With character

Table 4.4: IMDB training running times. Micro/Macro-averaged F1 for varying n.

	word n-grams				char n-grams			
max n-gram length	n=1	n=3	n=5	n unrestricted	n=1	n=3	n=5	n unrestricted
# overall features	63,623	838,620	1,922,942	N/A	135	41,433	704,440	N/A
# iterations SVM^{perf}	500	1,130	1,600	N/A	490	1,721	2,912	N/A
# iterations LIBLINEAR	960	3,500	6,500	N/A	370	6,500	17,500	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	4,167	N/A	97	3,045	4,061	N/A
# features in final model SLR	120	130	135	135	47	440	420	420
Running Time SVM^{perf}	0.3 min	6 min	15 min	N/A	0.2 min	3.2 min	50 min	N/A
Running Time LIBLINEAR	0.1 min	2.5 min	7.5 min	N/A	0.1 min	3.3 min	54 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^{perf}	1.3 min	8.5 min	18.5 min	N/A	1.2 min	6.2 min	55 min	N/A
Total Time LIBLINEAR	1.1 min	5 min	11 min	N/A	1.1 min	6.3 min	59 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
Total 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**
microavgF1 SVM^{perf}	69.08%	71.11%	70.71%	N/A	57.13%	65.27%	69.48%	N/A
microavgF1 LIBLINEAR	68.47%	70.89%	70.52%	N/A	56.78%	64.25%	69.42%	N/A
microavgF1 BBR	67.54%	67.97%	68.21%	N/A	56.32%	65.21%	67.10%	N/A
microavgF1 SLR	**72.90%**	**72.89%**	**72.64%**	**73.17%**	**55.99%**	**72.86%**	**73.84%**	**73.94%**
macroavgF1 SVM^{perf}	69.08%	71.13%	70.74%	N/A	57.16%	65.27%	69.48%	N/A
macroavgF1 LIBLINEAR	68.50%	70.90%	70.55%	N/A	56.79%	64.26%	69.43%	N/A
macroavgF1 BBR	67.56%	68.02%	68.24%	N/A	56.43%	65.21%	67.08%	N/A
macroavgF1 SLR	**72.95%**	**72.94%**	**72.72%**	**73.28%**	**56.09%**	**72.90%**	**73.89%**	**74.00%**

n-grams, SLR achieves the best macro-averaged F1 (74.88%), compared to BBR 74.72%, SVM^{perf} 70.43% and LIBLINEAR 69.46%. Increasing the n-gram length increases the classification quality of SLR with character n-grams. For $n = 1$ the SLR macro-averaged F1 is 58.43%, while for n unrestricted, the macro-averaged F1 increases to 74.88%. We also note that character n-grams provide better classification quality than word n-grams (best macro-averaged F1 for word n-grams of 74.04% compared to character n-grams of 74.88%).

Looking at memory consumption for the four methods, their requirements are as follows. For word 5-grams BBR uses 310 MByte of memory, SVM^{perf} 160 MByte and LIBLINEAR 50 MByte. For character 5-grams, BBR requires 340 MByte memory, SVM^{perf} 200 MByte and LIBLINEAR 122 MByte. SLR with unrestricted n, uses 40 MByte for word n-grams and 75 MByte for character n-grams.

Regarding running time, SLR is one order of magnitude faster than its competitors. In Table 4.4 we report the training running time for all methods. The times reported are average running times over cross-validation splits averaged across topics. We observe that our method is much faster than either BBR, SVM^{perf} or LIBLINEAR. In particular, LIBLINEAR was designed to take advantage of sparsity in the data, and we can observe that for word n-grams it is faster than SVM^{perf} but not faster than SLR. Furthermore, when we look at character n-grams, since the sparsity condition does not hold, i.e. most character n-grams occur in many documents, LIBLINEAR suffers from this symptom of dense data, and takes about the same time as SVM^{perf} and sometimes even a bit longer. Nevertheless, for both word and character n-grams SLR is much faster than its competitors.

For word n-grams, SLR's 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^{perf} shows a similar trend, its running time increases from 0.3 minutes for unigrams to 15 minutes for 5-grams. LIBLINEAR is slightly faster than SVM^{perf} for word n-grams, with running time of 0.1 minutes for $n = 1$ and of 7.5 minutes for 5-grams. For character-level n-grams SLR's running time stays almost constant at 1.7 minutes for 3-grams and 2.4 minutes for unrestricted n. BBR, SVM^{perf} and LIBLINEAR go from 2.4, 3.2 and 3.3 minutes respectively for character 3-grams, to 8.4, 50 and 54 minutes respectively for 5-grams.

Thus, SLR is an order of magnitude faster than the state-of-the-art methods SVM^{perf}, LIBLINEAR and BBR. This difference in running time is due to the way our technique deals with large feature spaces, by its pruning 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 4,167 distinct features in the final model. For character n-grams, our model runs for 620 iterations and selects 420 distinct features, while BBR runs for 370 iterations for 5-grams and selects 4,061 features.

In Table 4.8 we show the top 5 word-level and character-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 character n-grams in Table 4.8. 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, 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).

AMAZON dataset

Similarly, we run 5-fold cross validation on the AMAZON dataset and report the micro/macro-averaged F1 results. We then discuss the training running times for all methods on this dataset.

In terms of classification quality, for the fixed parameters setting (Table 4.4) SLR is better by 2-3% than either BBR, SVM^{perf} or LIBLINEAR. In the tuned setting (Table 4.7), SLR is comparable to the other techniques. For word n-grams, SLR has macro-averaged F1 of 83.16%, compared to BBR with 83.44%, SVM^{perf} 80.68% and LIBLINEAR 79.71%. For character-level n-grams, SLR has macro-averaged F1 of 83.01%, compared to BBR with 82.97%, SVM^{perf} 80.52% and LIBLINEAR 80.90%.

Again, SLR is much faster than BBR and SVM, with running time of 0.2 minutes for unrestricted word n-grams, and 1.5 minutes for unrestricted character n-grams. BBR takes 5.4 minutes for learning a word-level 5-gram model and 9.6 minutes for learning a 5-gram character model. SVM takes somewhat longer with 24 minutes average runtime for word-level 5-gram model and 55

Table 4.5: AMAZON training running times. Micro/Macro-averaged F1 for varying n.

	word n-grams				char n-grams			
max n-gram length	n=1	n=3	n=5	n unrestricted	n=1	n=3	n=5	n unrestricted
# overall features	55,764	1,036,693	2,454,205	N/A	95	50,422	700,993	N/A
# iterations SVM^{perf}	479	1089	1,722	N/A	592	1,578	2,597	N/A
# iterations LIBLINEAR	1,250	5,400	9,500	N/A	450	8,700	20,000	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^{perf}	0.2 min	6 min	24 min	N/A	0.5 min	2.5 min	55 min	N/A
Running Time LIBLINEAR	0.1 min	3.3 min	11.5 min	N/A	0.05 min	3.5 min	45 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^{perf}	0.3 min	7 min	28 min	N/A	1.5 min	7.5 min	63 min	N/A
Total Time LIBLINEAR	0.2 min	4.3 min	15.5 min	N/A	1.05 min	8.5 min	53 min	
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
Total 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**
microavgF1 SVM^{perf}	81.75%	80.42%	78.05%	N/A	39.01%	79.09%	81.88%	N/A
microavgF1 LIBLINEAR	81.11%	79.75%	78.04%	N/A	40.88%	78.57%	82.35%	N/A
microavgF1 BBR	79.08%	79.23%	80.13%	N/A	42.70%	78.37%	80.45%	N/A
microavgF1 SLR	**82.13%**	**81.80%**	**82.15%**	**81.89%**	**43.73%**	**82.75%**	**83.90%**	**84.49%**
macroavgF1 SVM^{perf}	80.24%	78.40%	75.31%	N/A	30.93%	77.37%	80.52%	N/A
macroavgF1 LIBLINEAR	79.44%	77.45%	75.14%		37.91%	76.77%	80.81%	N/A
macroavgF1 BBR	77.49%	77.50%	78.58%	N/A	39.82%	76.75%	78.92%	N/A
macroavgF1 SLR	**80.64%**	**80.30%**	**80.65%**	**80.40%**	**40.92%**	**81.31%**	**82.52%**	**83.20%**

minutes for character-level 5-gram model.

In Table 4.8 we show the 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 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*.

CHINESE dataset

This dataset has a separate set for training and for testing, thus we train all models on the training set and report micro/macro-averaged F1 on the test set. For tuning the hyperparameters, we run cross-validation on the training set. The best set of parameters is then user for learning the model on the full training set. The test set is never seen by the classifiers during leaning. The training

Table 4.6: 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^{perf}	146	116	148	N/A
# iterations LIBLINEAR	35	9	8	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^{perf}	0.1 min	1.5 min	5 min	N/A
Running Time LIBLINEAR	0.01 min	0.08 min	0.22 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^{perf}	1.1 min	6.5 min	14 min	N/A
Total Time LIBLINEAR	1.01 min	5.08 min	9.22 min	N/A
Total Time BBR	1.5 min	6.9 min	20 min	N/A
Total Time SLR	**0.25 min**	**0.5 min**	**0.5 min**	**0.5 min**
microavgF1 SVM^{perf}	72.82%	80.93%	78.96%	N/A
microavgF1 LIBLINEAR	73.20%	80.89%	77.91%	N/A
microavgF1 BBR	73.18%	76.65%	76.85%	N/A
microavgF1 SLR	**77.39%**	**78.87%**	**78.87%**	**78.87%**
macroavgF1 SVM^{perf}	73.29%	80.66%	78.60%	N/A
macroavgF1 LIBLINEAR	73.81%	80.61%	78.26%	N/A
macroavgF1 BBR	73.74%	77.43%	77.52%	N/A
macroavgF1 SLR	**77.69%**	**78.54%**	**78.54%**	**78.54%**

set contains 4,961 distinct characters, and about 6,000,000 character n-grams up to size 5.

Table 4.7 shows results on the prediction quality of all methods. SLR achieves 79.75% macro-averaged F1 which is similar to the 80.66% achieved by SVM^{perf}, 80.61% by LIBLINEAR and 79.63% by BBR.

Table 4.6 shows training run-times for this dataset. SLR takes 0.5 minutes for learning a model in the space of character-level n-grams of unrestricted length. BBR needs 11 minutes, while SVM^{perf} needs 5 minutes and LIBLINEAR needs 0.22 minutes for learning 5-gram models. The running times for BBR, SVM^{perf} and LIBLINEAR do not include the time required for pattern generation (9 extra minutes), since all these methods require generating the n-gram feature space explicitly.

Tables 4.4 to 4.7 show the 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 character n-grams, we observe that all methods benefit from using higher order n-gram features, also in the tuned setting. Overall, character 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 parameters 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 efficiently learn accurate models, thus avoiding the need for additional pre-processing, such as feature selection or word-segmentation.

Table 4.7: All collections. Tuned parameters. Micro/Macro-averaged F1 for varying n.

Collection		word n-grams				char n-grams			
	max n-gram length	n=1	n=3	n=5	n unrestricted	n=1	n=3	n=5	n unrestricted
IMDB	microavgF1 SVM^{perf}	69.21%	71.11%	71.22%	N/A	57.47%	65.65%	70.41%	N/A
	microavgF1 LIBLINEAR	69.07%	70.95%	71.27%	N/A	57.79%	65.01%	69.46%	N/A
	microavgF1 BBR	73.70%	73.58%	73.92%	N/A	56.97%	73.54%	74.71%	N/A
	microavgF1 SLR	**74.02%**	**73.74%**	**73.70%**	**73.86%**	**58.44%**	**73.82%**	**74.44%**	**74.87%**
	macroavgF1 SVM^{perf}	69.21%	71.13%	71.24%	N/A	57.47%	65.66%	70.43%	N/A
	macroavgF1 LIBLINEAR	69.09%	70.98%	71.32%	N/A	57.79%	65.01%	69.46%	N/A
	macroavgF1 BBR	73.76%	73.63%	73.94%	N/A	57.07%	73.56%	74.72%	N/A
	macroavgF1 SLR	**74.04%**	**73.77%**	**73.72%**	**73.91%**	**58.43%**	**73.89%**	**74.47%**	**74.88%**
AMAZON	microavgF1 SVM^{perf}	82.09%	80.60%	78.49%	N/A	39.32%	79.14%	81.88%	N/A
	microavgF1 LIBLINEAR	81.30%	80.11%	78.06%	N/A	40.57%	78.36%	82.41%	N/A
	microavgF1 BBR	84.58%	84.14%	83.99%	N/A	43.62%	82.98%	84.21%	N/A
	microavgF1 SLR	**84.22%**	**83.64%**	**83.75%**	**83.57%**	**45.54%**	**83.05%**	**84.21%**	**84.27%**
	macroavgF1 SVM^{perf}	80.68%	78.54%	75.79%	N/A	34.91%	77.47%	80.52%	N/A
	macroavgF1 LIBLINEAR	79.71%	77.73%	75.08%	N/A	37.58%	76.51%	80.90%	N/A
	macroavgF1 BBR	83.44%	82.95%	82.81%	N/A	40.62%	81.77%	82.97%	N/A
	macroavgF1 SLR	**83.16%**	**82.43%**	**82.55%**	**82.30%**	**42.61%**	**81.78%**	**82.94%**	**83.01%**
CHINESE	microavgF1 SVM^{perf}	N/A	N/A	N/A	N/A	72.82%	80.93%	78.96%	N/A
	microavgF1 LIBLINEAR	N/A	N/A	N/A	N/A	74.36%	80.89%	78.03%	N/A
	microavgF1 BBR	N/A	N/A	N/A	N/A	78.36%	78.88%	78.58%	N/A
	microavgF1 SLR	N/A	N/A	N/A	N/A	**76.18%**	**78.71%**	**78.30%**	**79.17%**
	macroavgF1 SVM^{perf}	N/A	N/A	N/A	N/A	73.29%	80.66%	78.60%	N/A
	macroavgF1 LIBLINEAR	N/A	N/A	N/A	N/A	74.77%	80.61%	77.68%	N/A
	macroavgF1 BBR	N/A	N/A	N/A	N/A	78.17%	79.63%	79.36%	N/A
	macroavgF1 SLR	N/A	N/A	N/A	N/A	**76.09%**	**79.35%**	**79.06%**	**79.75%**

Table 4.8: Top 5 features IMDB and AMAZON. Max n-gram length n=5.

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

4.4 Other Applications: Spam Filtering

Spam filtering is a mission-critical text classification application, which can directly affect our day to day lives. Having a legitimate e-mail end up in the spam folder, or receiving hundreds of spam e-mails every day, can be highly damaging to both private individuals and large industries. In this section we analyze the effect of using our SLR algorithm for learning variable-length n-gram models for spam filtering. In particular, character n-gram models have the potential of improving state-of-the-art classification prediction, since they can potentially capture many ways of altering the original text typically used by spammers to increase the chances that spam gets through to the user. For these experiments we compile a corpus of *spam* and legitimate e-mails (referred to as *ham*) from several large spam benchmarks: TREC 2005 and TREC 2006 Public Spam Corpora [spab, BHS06, BFC$^+$06] and the Guenter spam trap [BHS06, spaa]. The pooled corpus amounts to 201,324 e-mails. From the large pool we randomly select 100,000 e-mails for training and 100,000 e-mails for testing. Table 4.9 gives exact details on the number of spam/ham in the training and test sets. We evaluate the classification quality by the F1 measure [Cha03, MRS08] and the Area Under the Roc Curve (AUC) [MRS08, Faw04]. The AUC measure is particularly used in spam filtering benchmarks. The ROC (Receiver Operating Characteristics) curve is obtained by plotting the fraction of true positives (also called *sensitivity*) versus the fraction of false positives (also called *1 - specificity*) for a binary classifier system as its discrimination threshold is varied [MRS08]. A common aggregate measure is to report the area under the ROC curve (AUC), which is the ROC analog of Mean Average Precision [MRS08].

As a basis for comparison, we give the F1 and the AUC measures reported in [BHS06] obtained on a set of randomly selected 100,000 training and 100,000 test emails from the same benchmarks in relation to the SpamTREC 2006 challenge: F1: 92.78%, AUC: 98.78%.

Typical approaches to spam filtering spend considerable time in pre-processing the input data and in doing feature selection. We keep the original text as it is (no pre-processing, no feature selection), and rely on the learning algorithm to be able to select discriminative n-grams for the final model.

This experimental study aims at:

1. Analyzing the scalability of SLR on a large real world corpus, and the trade-offs between training run-time and prediction quality.

2. Understanding the effect of using unrestricted-length word/character n-grams on a real world application of vital importance.

Table 4.9: **Spam corpus.** Details on the training/test per collection.

Collection	Training size		Test size	
Spam corpus	100,000		100,000	
	Spam	Ham	Spam	Ham
	74,354	25,646	74,228	25,772

Table 4.10: SPAM training running times. F1 and AUC measures for varying n.

	word n-grams				char n-grams			
max n-gram length	n=1	n=3	n=5	n unrestricted	n=1	n=3	n=5	n unrestricted
# overall features	2,198,919	20,627,820	51,714,667	N/A	185	635,118	25,138,274	N/A
convergence_threshold SLR1	2.5e-04	2.5e-04	2.5e-04	2.5e-04	2.5e-03	2.5e-03	2.5e-03	2.5e-03
convergence_threshold SLR2	5e-06	5e-06	5e-06	5e-06	5e-04	5e-04	5e-04	5e-04
# iterations SLR1	1,005	1,001	1,013	1,156	499	555	549	561
# features in SLR1 model	470	493	510	570	82	418	424	425
# iterations SLR2	3,962	5,803	4,074	4,086	1,606	2,739	3,040	2,570
# features in SLR2 model	1,308	1,735	1,395	1,417	106	1,496	1,627	1,473
Running Time LIBLINEAR	1 min	14 min	60 min	N/A	1 min	12 min	138 min	N/A
Running Time BBR	120 min	615 min	N/A	N/A	14 min	1,055 min	N/A	N/A
Running Time SLR1	57 min	79 min	84 min	97 min	19 min	70 min	99 min	150 min
Running Time SLR2	211 min	405 min	292 min	319 min	57 min	303 min	513 min	688 min
Time for generating patterns	16 min	45 min	75 min	N/A	25 min	90 min	150 min	N/A
Total Time LIBLINEAR	**17 min**	**59 min**	**135 min**	**N/A**	**26 min**	**102 min**	**288 min**	**N/A**
Total Time BBR	**136 min**	**660 min**	**N/A**	**N/A**	**39 min**	**1,145 min**	**N/A**	**N/A**
Total Time SLR1	**57 min**	**79 min**	**84 min**	**97 min**	**19 min**	**70 min**	**99 min**	**150 min**
Total Time SLR2	**211 min**	**405 min**	**292 min**	**319 min**	**57 min**	**303 min**	**513 min**	**688 min**
F1 LIBLINEAR	99.43%	99.39%	99.32%	N/A	93.44%	99.33%	99.44%	N/A
F1 BBR	99.40%	99.44%	N/A	N/A	93.50%	99.40%	N/A	N/A
F1 SLR1	98.84%	98.94%	98.92%	99.03%	93.55%	98.81%	98.89%	98.88%
F1 SLR2	99.28%	99.35%	99.29%	99.32%	93.63%	99.25%	99.35%	99.30%
AUC LIBLINEAR	**98.74%**	**98.62%**	**98.50%**	**N/A**	**86.22%**	**98.60%**	**98.83%**	**N/A**
AUC BBR	**99.86%**	**99.87%**	**N/A**	**N/A**	**95.27%**	**99.86%**	**N/A**	**N/A**
AUC SLR1	**99.71%**	**99.74%**	**99.74%**	**99.76%**	**95.09%**	**99.71%**	**99.75%**	**99.74%**
AUC SLR2	**99.81%**	**99.83%**	**99.82%**	**99.82%**	**95.32%**	**99.84%**	**99.87%**	**99.86%**

3. Further analyzing the influence of parameters on the overall performance of the SLR algorithm.

We show experiments with SLR, BBR and LIBLINEAR. Since in the previous experiments the results of SVM^{perf} and LIBLINEAR were very similar in terms of both accuracy and running time, we show results only for LIBLINEAR. The hyperparameters of BBR and LIBLINEAR were tuned using cross-validation on the training set. We report running time, F1 and AUC results for the tuned values of their parameters. For SLR we vary the convergence threshold which affects the number of optimization iterations executed by our algorithm. The value of the convergence threshold can affect the accuracy and the running time of the model. We show results for two values of the threshold (we coin the models SLR1 and SLR2), which give a feeling of the trade-off between running time and F1/AUC values. In Table 4.10 we give details on the models and the F1 and AUC values obtained on the test set. We denote by N/A the entries for which the respective model ran out-of-memory.

First we note that the training corpus has a size of 320MBytes on disk and contains about 2 million distinct word unigrams. Therefore, explicitly generating the n-gram feature space as required by both BBR and LIBLINEAR is an obvious disadvantage. The files corresponding to the explicit feature spaces for $n = 3$ and $n = 5$ are large, occupying around 4 and respectively 8 GByte disk space. Correspondingly, the amount of memory required by BBR and LIBLINEAR is very demanding. BBR requires about 6.5 GByte for $n = 3$ and more than 8 GByte memory (which is more than the resources of the machine used for these experiments), for $n = 5$. Therefore, for BBR we only show experiments for $n = 1$ and $n = 3$. LIBLINEAR requires about 4 GByte memory for $n = 3$ and around 6.5 GByte memory for $n = 5$. SLR takes 1.5 GByte memory for *unrestricted* word n-grams and 2.2 GByte for *unrestricted* character n-grams.

Regarding the AUC measure, SLR delivers results comparable to those of BBR, i.e. 99.75% for SLR1 and 99.87% for SLR2 versus 99.87% for BBR. LIBLINEAR has the smallest AUC with 98.83%. Regarding running time, SLR1 is a good compromise between LIBLINEAR and BBR. LIBLINEAR is quite fast, with 135 minutes for word 5-grams, and 288 minutes for character 5-grams. SLR1 takes 97 minutes for *unrestricted* word n-grams, and 150 minutes for *unrestricted* character n-grams. BBR needs 136 minutes for word unigrams and 660 minutes for word 3-grams. For character n-grams, BBR takes 39 minutes for 1-grams and 1,145 minutes for 3-grams. By lowering the convergence threshold for SLR (denoted in Table 4.10 by SLR2) we can achieve higher F1/AUC values at some cost with respect to running time.

We note that the classification performance with respect to both F1 and AUC values is higher for all methods than that reported in [BHS06], for both word n-grams and character n-grams.

For SLR2 with word n-grams, the F1/AUC values increase with increasing n-gram size, with a peak at maximum $n = 3$ with F1: 99.35% and AUC: 99.83%. The character n-gram models further improve these results. For increasing n-gram size the classification quality increases, with a peak at $n = 5$ with F1: 99.35% and AUC: 99.87%. For both BBR and SLR the classification quality with unrestricted n-gram size is closer to the peak than when using unigrams alone. This shows the importance of using n-grams for spam filtering.

In conclusion, from these experiments we observe that LIBLINEAR is quite fast, but does not deliver satisfactory AUC results which are considered more important that the F1 value by the spam filtering community. Additionally LIBLINEAR requires a considerable amount of memory. BBR is slower than our method, but provides good AUC results (99.87%). Nevertheless, similar to LIBLINEAR it has considerable memory demands. SLR provides AUC results comparable to BBR (99.74% for SLR1 and 99.87% for SLR2), at a much lower running time and memory costs.

4.5 Using Explicit Regularization on the Objective Function

In this section we investigate the effect of using an explicit regularizer on the logistic regression objective function and we give new theoretical bounds for the SLR algorithm in this setting. We first focus on the L1 regularizer and then discuss the use of other regularizers such as the L2 regularizer or an n-gram-length-aware regularizer. As we mentioned in the previous sections, the L1 regularized objective takes the form:

$$\tilde{l_1} = \max_{\beta} \ l(\beta) - \lambda \cdot \sum_{j=1}^{d} |\beta_j|$$

We show next how introducing an explicit regularizer affects the pruning bound of SLR. The new objective function becomes:

$$\tilde{l_1} = \max_{\beta} \ \sum_{i=1}^{N} [y_i \cdot \beta^T \cdot x_i - log(1 + e^{\beta^T \cdot x_i})] - \lambda \cdot \sum_{j=1}^{d} |\beta_j| \tag{4.12}$$

Using Equation 4.12, the gradient of $\tilde{l_1}$ with respect to a coordinate β_j evaluated at a given parameter vector β is:

$$\frac{\partial \tilde{l_1}}{\partial \beta_j}(\beta) = \sum_{i=1}^{N} x_{ij} \cdot \left(y_i - \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right) - \lambda \cdot sign(\beta_j) \tag{4.13}$$

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, (i.e., s_j is a prefix of $s_{j'}$). We write $s_j \in x_i$ to denote $x_{ij} \neq 0$.

The following theorem gives a convenient way of computing an upper bound on the gradient value for any super sequence $s_{j'} \supseteq s_j$.

Theorem 4.5.1 *For any $s_{j'} \supseteq s_j$ and $y \in \{0, 1\}$, the absolute value of the gradient of $\tilde{l_1}(\beta)$ with respect to $\beta_{j'}$ is bounded by $\mu(\beta_j)$, where*

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

Proof *Similar to the proof of Theorem 4.2.1 we split the analysis into two subproblems, the first concerning the "positive" class ($y = 1$), and the second concerning the "negative" class ($y = 0$). First we prove the bound for the positive class:*

$$\frac{\partial \tilde{l_1}}{\partial \beta_{j'}}(\beta) = \sum_{i=1}^{N} x_{ij'} \cdot \left(y_i - \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right) - \lambda \cdot sign(\beta_{j'}) \tag{4.14}$$

$$= \sum_{\{i|s_{j'} \in x_i\}} x_{ij'} \cdot \left(y_i - \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right) - \lambda \cdot sign(\beta_{j'}) \tag{4.15}$$

$$\leq \sum_{\{i|y_i=1, s_{j'} \in x_i\}} x_{ij'} \cdot \left(1 - \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right) - \lambda \cdot sign(\beta_{j'}) \tag{4.16}$$

$$\leq \sum_{\{i|y_i=1, s_j \in x_i\}} x_{ij} \cdot \left(1 - \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right) + \lambda. \tag{4.17}$$

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 $s_{j'} \supseteq s_j$. Additionally, since the $sign(\beta'_j) = \pm 1$, we can give a simple bound on the gradient value at the coordinate $s_{j'}$.

Similarly , we can show for the negative class that

$$\frac{\partial \tilde{l_1}}{\partial \beta_{j'}}(\beta) \geq \sum_{\{i|y_i=0, s_j \in x_i\}} x_{ij} \cdot \left(- \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right) - \lambda. \tag{4.18}$$

Thus we have:

$$\sum_{\{i|y_i=0,s_j\in x_i\}} x_{ij}\cdot\left(-\frac{e^{\beta^T\cdot x_i}}{1+e^{\beta^T\cdot x_i}}\right)-\lambda\le\frac{\partial\tilde{l_1}}{\partial\beta_{j'}}(\beta) \tag{4.19}$$
$$\le\sum_{\{i|y_i=1,s_j\in x_i\}} x_{ij}\cdot\left(1-\frac{e^{\beta^T\cdot x_i}}{1+e^{\beta^T\cdot x_i}}\right)+\lambda$$

The absolute value of the gradient of $\tilde{l_1}(\beta)$ at coordinate j' corresponding to n-gram sequence $s_{j'}$ is thus bounded by $\mu(\beta_j)$:

$$\left|\frac{\partial\tilde{l_1}}{\partial\beta_{j'}}(\beta)\right| \le max\{\sum_{\{i|y_i=1,s_j\in x_i\}} x_{ij}\cdot\left(1-\frac{e^{\beta^T\cdot x_i}}{1+e^{\beta^T\cdot x_i}}\right)+\lambda,$$
$$\sum_{\{i|y_i=0,s_j\in x_i\}} x_{ij}\cdot\left(\frac{e^{\beta^T\cdot x_i}}{1+e^{\beta^T\cdot x_i}}\right)+\lambda\}.$$

□

We note from the previous theorem that in order to use L1 regularization with the SLR algorithm we only need to adjust the pruning bound slightly, and pay a small computation cost to retrieve the sign of the current β_j coefficient (needed for computing the current gradient value). This is not the case when using the L2 regularizer. Let $\tilde{l_2}(\beta)$ be the L2 regularized logistic objective:

$$\tilde{l_2}=\max_{\beta}\ \sum_{i=1}^{N}[y_i\cdot\beta^T\cdot x_i-log(1+e^{\beta^T\cdot x_i})]-\frac{\lambda}{2}\cdot\sum_{j=1}^{d}{\beta_j}^2 \tag{4.20}$$

We follow the same steps as for the proof using the L1 regularizer. Using Equation 4.20, the gradient of l with respect to a coordinate value β_j evaluated at a given parameter vector β is:

$$\frac{\partial\tilde{l_2}}{\partial\beta_j}(\beta)=\sum_{i=1}^{N}x_{ij}\cdot\left(y_i-\frac{e^{\beta^T\cdot x_i}}{1+e^{\beta^T\cdot x_i}}\right)-\lambda\cdot\beta_j \tag{4.21}$$

We note that the presence of the coefficient β_j in the gradient computation makes it harder to write down a clean bound.

$$\frac{\partial\tilde{l_2}}{\partial\beta_{j'}}(\beta) = \sum_{i=1}^{N}x_{ij'}\cdot\left(y_i-\frac{e^{\beta^T\cdot x_i}}{1+e^{\beta^T\cdot x_i}}\right)-\lambda\cdot\beta_{j'} \tag{4.22}$$
$$= \sum_{\{i|s_{j'}\in x_i\}}x_{ij'}\cdot\left(y_i-\frac{e^{\beta^T\cdot x_i}}{1+e^{\beta^T\cdot x_i}}\right)-\lambda\cdot\beta_{j'} \tag{4.23}$$
$$\le \sum_{\{i|y_i=1,s_{j'}\in x_i\}}x_{ij'}\cdot\left(1-\frac{e^{\beta^T\cdot x_i}}{1+e^{\beta^T\cdot x_i}}\right)-\lambda\cdot\beta_{j'} \tag{4.24}$$
$$\le \sum_{\{i|y_i=1,s_j\in x_i\}}x_{ij}\cdot\left(1-\frac{e^{\beta^T\cdot x_i}}{1+e^{\beta^T\cdot x_i}}\right)-\lambda\cdot\beta_{j'}. \tag{4.25}$$

Additionally, since $\beta'_j \in \Re$ we do not have the same nice monotonicity property when trying to bound the gradient value of sequence $s_{j'}$ based on its prefix s_j, as for the L1 regularizer case. In order to guarantee that we find the best n-gram sequence in a given iteration, we need to explicitly compute the quantity in Equation 4.25 for all the non-zero n-grams (i.e. previously selected features) which have s_j as a prefix. This implies that at each point in the search space, when checking the pruning bound we need to explicitly compute the quantity in Equation 4.25 for all non-zero features starting with this prefix. This additional computation can result in making the algorithm less efficient (by increasing the training time).

Furthermore, using an n-gram-length-aware regularizer suffers from the same need for additional computation as the L2 regularizer. For example, even when using an additional length penalty in the L1 regularizer we run into the same problem of having to explicitly compute the bounding value for all non-zero features.

$$\tilde{l_*} = \max_{\beta} \sum_{i=1}^{N}[y_i \cdot \beta^T \cdot x_i - log(1 + e^{\beta^T \cdot x_i})] - \lambda \cdot \sum_{j=1}^{d} |\beta_j| \cdot n_j \tag{4.26}$$

$$\begin{aligned}
\frac{\partial \tilde{l_*}}{\partial \beta_{j'}}(\beta) &= \sum_{i=1}^{N} x_{ij'} \cdot \left(y_i - \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right) - \lambda \cdot sign(\beta_{j'}) \cdot n_{j'} && (4.27)\\
&= \sum_{\{i|s_{j'} \in x_i\}} x_{ij'} \cdot \left(y_i - \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right) - \lambda \cdot sign(\beta_{j'}) \cdot n_{j'} && (4.28)\\
&\leq \sum_{\{i|y_i=1, s_{j'} \in x_i\}} x_{ij'} \cdot \left(1 - \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right) - \lambda \cdot sign(\beta_{j'}) \cdot n_{j'} && (4.29)\\
&\leq \sum_{\{i|y_i=1, s_j \in x_i\}} x_{ij} \cdot \left(1 - \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right) + \lambda \cdot n_{j'}. && (4.30)
\end{aligned}$$

As we note from Equation 4.30 we need to explicitly compute the bounding value for all non-zero features starting with prefix s_j. This again results in additional computation during the training process which can make the learning algorithm inefficient.

Using the L1 regularizer is widely regarded in the literature [GLM06] as a good choice for controlling the selection of features for the final model. Many learning researchers [GLM06] actually advocate the use of L1 regularization over L2 regularization, since the L1 regularization has the advantage of shrinking many feature coefficients to zero, therefore providing feature selection and learning in a single algorithm. Since training SLR using an L1 regularizer comes at small additional computation time we focus next on empirically analyzing the effect of using L1 regularization with SLR.

4.5.1 Empirical Results using SLR with L1 Regularization

One problem concerning L1 regularization is that the regularized objective function is not differentiable at $\beta_j = 0$. Furthermore, if β_j switches its sign (i.e., we selected the feature as positive and now it is selected as negative) the increase in the objective function is no longer guaranteed.

Table 4.11: IMDB training running times. Micro/Macro-averaged F1 for unrestricted n.

	word n-grams			char n-grams		
L1 regularizer	time	microavgF1	macroavgF1	time	microavgF1	macroavgF1
$\lambda = 0$	*0.35 min*	*73.17*	*73.28*	*2.40 min*	*73.94*	*74.00*
$\lambda = 0.01$	1.25 min	72.62	72.67	7.56 min	74.07	74.09
$\lambda = 0.02$	1.27 min	72.73	72.78	7.55 min	73.69	73.73
$\lambda = 0.04$	1.27 min	73.02	73.04	7.36 min	73.93	73.96
$\lambda = 0.08$	1.15 min	72.62	72.68	7.26 min	74.01	74.03
$\lambda = 0.1$	1.20 min	72.92	73.01	7.53 min	73.57	73.61
$\lambda = 0.125$	1.30 min	72.78	72.84	7.38 min	73.89	73.91
$\lambda = 0.25$	1.25 min	**73.36**	**73.40**	7.69 min	74.12	74.19
$\lambda = 0.5$	1.25 min	72.97	73.04	6.95 min	73.45	73.47
$\lambda = 1$	1.26 min	72.88	72.96	7.08 min	74.12	74.14
$\lambda = 5$	1.10 min	71.89	72.00	5.88 min	**74.20**	**74.24**

Table 4.12: AMAZON training running times. Micro/Macro-averaged F1 for unrestricted n.

	word n-grams			char n-grams		
L1 regularizer	time	microavgF1	macroavgF1	time	microavgF1	macroavgF1
$\lambda = 0$	*0.20 min*	*81.89*	*80.40*	*1.50 min*	*84.49*	*83.20*
$\lambda = 0.01$	0.70 min	79.49	77.63	2.57 min	83.98	82.75
$\lambda = 0.02$	0.65 min	79.46	77.65	3.09 min	84.16	82.95
$\lambda = 0.04$	0.68 min	79.79	78.05	2.53 min	84.01	82.68
$\lambda = 0.08$	0.73 min	**79.85**	**78.19**	2.36 min	**84.42**	**83.16**
$\lambda = 0.1$	0.71 min	79.63	77.97	2.56 min	84.09	82.82
$\lambda = 0.125$	0.76 min	79.54	77.90	2.59 min	84.21	82.84
$\lambda = 0.25$	0.67 min	79.45	77.65	2.27 min	83.94	82.73
$\lambda = 0.5$	0.74 min	79.26	77.47	2.77 min	84.21	82.92
$\lambda = 1$	0.69 min	79.14	77.30	2.66 min	84.02	82.72
$\lambda = 5$	0.58 min	77.94	76.03	2.43 min	84.19	82.76

In order to tackle these problems we follow the steps in [GLM06], i.e., whenever the coefficient β_j changes its sign (crosses zero), we set it to $\beta_j = 0$.

Furthermore, some additional running time goes into retrieving the sign of the coefficient for the coordinate currently investigated during gradient and bounds computation. Since we keep all the selected features in a binary search tree, we need logarithmic time in the size of the selected feature set for retrieving such information. Nevertheless, this search has to be done repeatedly for each coordinate, in each optimization iteration, therefore we should expect the regularized SLR algorithm to be somewhat slower than its original (not explicitly regularized) counterpart.

Table 4.11 shows results for the L1 regularized SLR on the IMDB dataset. We fix the convergence threshold to 0.005, and show running time and micro/macro-averaged F1 results for varying values of the regularizer value, denoted by λ. We denote by $\lambda = 0$ the original SLR without explicit regularization (presented in Section 4.2). We observe that varying the value of the regularizer can affect the running time and the micro/macro-averaged F1 values to some degree. The variation in running time is nevertheless very little affected by the regularizer value λ. This happens because of the following effect: increasing the regularizer value leads to less optimization iterations, but in each iteration the time required for finding the best n-gram feature increases with increasing λ. We can observe the letter effect from the bound given in Theorem 4.5.1. The larger the λ, the looser the bound, and therefore the more time is needed to search for the best n-gram in a given iteration.

Table 4.13: CHINESE training running times. Micro/Macro-averaged F1 for unrestricted n.

	char n-grams		
L1 regularizer	time	microavgF1	macroavgF1
$\lambda = 0$	*0.50 min*	*78.87*	*78.54*
$\lambda = 0.01$	1.60 min	77.32	77.42
$\lambda = 0.02$	1.68 min	77.57	77.70
$\lambda = 0.04$	1.67 min	77.63	77.86
$\lambda = 0.08$	1.63 min	78.04	78.07
$\lambda = 0.1$	1.65 min	78.57	78.63
$\lambda = 0.125$	1.61 min	78.07	77.98
$\lambda = 0.25$	1.67 min	78.19	78.27
$\lambda = 0.5$	1.41 min	**78.70**	**78.59**
$\lambda = 1$	1.56 min	78.35	78.15
$\lambda = 5$	1.85 min	77.47	77.15

The variation in micro/macro-averaged F1 with increasing λ is about 1-2%. The largest micro-averaged F1 (73.36%) value for word-ngrams is achieved with $\lambda = 0.25$. For character n-grams the highest value of the micro-averaged F1 was achieved at $\lambda = 5$ (74.20%).

We note that using L1 regularization can slightly improve the SLR results as compared to the original SLR algorithm. At the same time when comparing running times, because we have to pay some small additional computation in the regularized case, the regularized SLR is slightly slower than the original SLR. For example, the running time for word n-grams goes up from 0.35 min to 1.30 min, and for character n-grams from 2.4 min to 7.55 min. Using regularization seems to have a higher impact for character n-grams than for word n-grams. Decreasing the convergence threshold to less than 0.005 resulted in increased running times, without notable accuracy benefits.

In Table 4.12 we show results on the AMAZON dataset. Similar to IMDB, using regularization has a more notable effect on the character n-gram models than on the word n-gram models. For word n-grams the micro/macro-averaged F1 results are actually lower than for the original (un-regularized) SLR, for all the regularizer values tried. For character n-grams, the micro-averaged F1 results are comparable to those of original SLR (84.42% versus 84.49%). Regarding running time, the regularized version requires 2.36 minutes as compared to 1.5 minutes for the un-regularized version.

Table 4.13 summarizes the results on the CHINESE dataset. The classification results using regularization are not better than those achieved by original SLR (micro-averaged F1 of 78.70% versus 78.87%). Running-time wise, the regularized version is also slower than the un-regularized one, with 1.41 minutes versus 0.5 minutes. This preliminary experiments suggest that using regularization with SLR does not improve the results of the original SLR algorithm. This may happen because of the way our algorithm works, i.e. the feature selection strategy in our algorithm (always choose the feature with best gradient value) can play an implicit regularization effect on the final model selected. Furthermore, using regularization comes with additional computational costs which results in increased running time. In the future, we plan to further investigate using other types of regularization with SLR.

4.6 Speeding Up SLR by Using Second Order Information

In the previous sections we presented a first order algorithm for solving logistic regression in the large space of all n-gram features. There are several optimization approaches in the literature which use second order information for faster convergence. These are called Newton algorithms [NW06], which are typically quadratic, or quasi-Newton methods which are linear in the feature space size. In this section we give an upper bound on the magnitude of the weight of any n-gram sequence based on its prefix, for the case of using second order information.

Let

$$l(\beta) = \sum_{i=1}^{N} [y_i \beta^\top \mathbf{x}_i - log(1 + e^{\beta^\top \mathbf{x}_i})]$$

be the log-likelihood function for logistic regression. We are interested in maximizing $l(\beta)$. We will achieve this by repeatedly optimizing in one dimension, while keeping all the other dimensions fixed, e.g. the update for a single dimension j using second order information is:

$$\beta^{(n+1)} = \beta^{(n)} + \frac{\frac{\partial l}{\partial \beta_j}(\beta^{(n)})}{\frac{\partial^2 l}{\partial {\beta_j}^2}(\beta^{(n)})}$$

The gradient of l with respect to a coordinate β_j at some point β is:

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

From this it follows that the second derivative of l with respect to a coordinate β_j at some point β is:

$$\begin{aligned}
\frac{\partial^2 l}{\partial {\beta_j}^2}(\beta) &= \left[\sum_{i=1}^{N} x_{ij} \left(y_i - \frac{e^{\beta^\top x_i}}{1 + e^{\beta^\top x_i}} \right) \right]_{\partial \beta_j} && (4.31)\\
&= - \left[\sum_{i=1}^{N} \mathbf{x}_{ij} \frac{e^{\beta^\top x_i}}{1 + e^{\beta^\top x_i}} \right]_{\partial \beta_j} && (4.32)\\
&= - \left[\sum_{i=1}^{N} {x_{ij}}^2 \cdot \frac{e^{\beta^\top x_i}}{1 + e^{\beta^\top x_i}} - \sum_{i=1}^{N} {x_{ij}}^2 \cdot \frac{e^{2\beta^\top x_i}}{\left(1 + e^{\beta^\top x_i}\right)^2} \right] && (4.33)\\
&= \sum_{i=1}^{N} {x_{ij}}^2 \cdot \frac{e^{\beta^\top x_i}}{\left(1 + e^{\beta^\top x_i}\right)} \cdot \left(\frac{e^{\beta^\top x_i}}{1 + e^{\beta^\top x_i}} - 1 \right) \le 0 && (4.34)\\
& && (4.35)
\end{aligned}$$

The Newton step projected at a coordinate $\beta_{j'}$ corresponding to a super-sequence $s_{j'}$ of s_j is thus:

$$\frac{\frac{\partial l}{\partial \beta_{j'}}(\beta)}{\frac{\partial^2 l}{\partial {\beta_{j'}}^2}(\beta)} = \frac{\sum_{i=1}^{N} x_{ij} \left(y_i - \frac{e^{\beta^\top x_i}}{1+e^{\beta^\top x_i}} \right)}{\sum_{i=1}^{N} {x_{ij}}^2 \cdot \frac{e^{\beta^\top x_i}}{\left(1+e^{\beta^\top x_i}\right)} \cdot \left(\frac{e^{\beta^\top x_i}}{1+e^{\beta^\top x_i}} - 1 \right)} \qquad (4.36)$$

We prove a bound on the absolute magnitude of the Newton update $\left|\frac{\frac{\partial l}{\partial \beta_{j'}}(\beta)}{\frac{\partial^2 l}{\partial \beta_{j'}{}^2}(\beta)}\right|$ corresponding to a super-sequence $s_{j'}$ which only uses information about the subsequence s_j of $s_{j'}$. This will help us prune the search space while looking for the best coordinate w.r.t. the absolute magnitude of the Newton update.

We have:

$$\left|\frac{\frac{\partial l}{\partial \beta_{j'}}(\beta)}{\frac{\partial^2 l}{\partial \beta_{j'}{}^2}(\beta)}\right| = \left|\frac{\sum_{i=1}^N x_{ij'}\left(y_i - \frac{e^{\beta^\top x_i}}{1+e^{\beta^\top x_i}}\right)}{\sum_{i=1}^N x_{ij'}{}^2 \cdot \frac{e^{\beta^\top x_i}}{\left(1+e^{\beta^\top x_i}\right)} \cdot \left(\frac{e^{\beta^\top x_i}}{1+e^{\beta^\top x_i}} - 1\right)}\right| \tag{4.37}$$

$$= \frac{\left|\sum_{i=1}^N x_{ij'}\left(y_i - \frac{e^{\beta^\top x_i}}{1+e^{\beta^\top x_i}}\right)\right|}{\sum_{i=1}^N x_{ij'}{}^2 \cdot \frac{e^{\beta^\top x_i}}{\left(1+e^{\beta^\top x_i}\right)} \cdot \left(1 - \frac{e^{\beta^\top x_i}}{1+e^{\beta^\top x_i}}\right)} \tag{4.38}$$

We cannot directly apply the trick used for proving a bound on the first derivative of l because the numerator and the denominator behave similarly, i.e. even if the numerator is bounded by the same expression in which we replace the subsequence occurrences, so does the denominator, thus sort of balancing out the effect. Rather than trying to bound the numerator and the denominator separately, we will prove a bound directly on the ratio. Since we work with a binary setting ($x_{ij'} \in \{0,1\}$), we observe that $x_{ij'} = x_{ij'}^2$.

Let $u = \sum_{i=1}^N x_{ij'}\left(y_i - \frac{e^{\beta^\top x_i}}{1+e^{\beta^\top x_i}}\right) = \sum_{\{i|x_{ij'}=1\}}\left(y_i - \frac{e^{\beta^\top x_i}}{1+e^{\beta^\top x_i}}\right)$ and

$$v = \sum_{i=1}^N x_{ij'}{}^2 \frac{e^{\beta^\top x_i}}{\left(1+e^{\beta^\top x_i}\right)} \cdot \left(1 - \frac{e^{\beta^\top x_i}}{1+e^{\beta^\top x_i}}\right) = \sum_{\{i|x_{ij'}=1\}} \frac{e^{\beta^\top x_i}}{\left(1+e^{\beta^\top x_i}\right)} \cdot \left(1 - \frac{e^{\beta^\top x_i}}{1+e^{\beta^\top x_i}}\right).$$

Let $u_+ = \sum_{\{i|x_{ij'}=1,y_i=1\}}\left(1 - \frac{e^{\beta^\top x_i}}{1+e^{\beta^\top x_i}}\right)$, $u_- = \sum_{\{i|x_{ij'}=1,y_i=0\}}\left(-\frac{e^{\beta^\top x_i}}{1+e^{\beta^\top x_i}}\right)$. Let $z_i = \frac{e^{\beta^\top x_i}}{1+e^{\beta^\top x_i}}$.

Then:

$$u = \sum_{\{i|x_{ij'}=1\}} (y_i - z_i);\; u_+ = \sum_{\{i|x_{ij'}=1,y_i=1\}} (1-z_i);\; u_- = \sum_{\{i|x_{ij'}=1,y_i=0\}} -z_i;\; v = \sum_{\{i|x_{ij'}=1\}} z_i \cdot (1-z_i).$$

We can directly see that the following inequality holds:

$$\frac{u_-}{v} \leq \frac{u}{v} \leq \frac{u_+}{v}.$$

We would like to find bounds f and g that depend only on the occurrences of the subsequence s_j (rather than the super-sequence $s_{j'}$) such that:

$$f \leq \frac{u_-}{v} \leq \frac{u}{v} \leq \frac{u_+}{v} \leq g.$$

Next, we prove such bounds exist:

$$\begin{aligned}
\frac{u_+}{v} &= \frac{\sum_{\{i|x_{ij'}=1,y_i=1\}}(1-z_i)}{\sum_{\{i|x_{ij'}=1\}} z_i \cdot (1-z_i)} && (4.39)\\
&\le \frac{\sum_{\{i|x_{ij'}=1,y_i=1\}}(1-z_i)}{\sum_{\{i|x_{ij'}=1,y_i=1\}} z_i \cdot (1-z_i)} && (4.40)\\
&= \frac{\sum_{\{i|x_{ij'}=1,y_i=1\}} \frac{1}{z_i} \cdot z_i \cdot (1-z_i)}{\sum_{\{i|x_{ij'}=1,y_i=1\}} z_i \cdot (1-z_i)} && (4.41)\\
&\le \frac{\sqrt{\sum_{\{i|x_{ij'}=1,y_i=1\}} \frac{1}{{z_i}^2}} \cdot \sqrt{\sum_{\{i|x_{ij'}=1,y_i=1\}} {z_i}^2 \cdot (1-z_i)^2}}{\sum_{\{i|x_{ij'}=1,y_i=1\}} z_i \cdot (1-z_i)} && (4.42)\\
&\le \sqrt{\sum_{\{i|x_{ij'}=1,y_i=1\}} \frac{1}{{z_i}^2}} \cdot \frac{\sqrt{\sum_{\{i|x_{ij'}=1,y_i=1\}} {z_i}^2 \cdot (1-z_i)^2}}{\sum_{\{i|x_{ij'}=1,y_i=1\}} z_i \cdot (1-z_i)} && (4.43)\\
&\le \sqrt{\sum_{\{i|x_{ij'}=1,y_i=1\}} \frac{1}{{z_i}^2}} && (4.44)\\
&\le \sqrt{\sum_{\{i|\mathbf{x_{ij}=1},y_i=1\}} \frac{1}{{z_i}^2}} && (4.45)
\end{aligned}$$

Equation 4.42 uses the Cauchy-Schwarz inequality and the fact that $\frac{1}{z_i} \ge 0$ and $z_i \cdot (1-z_i) \ge 0$.

Equation 4.43 uses the fact that $\sum_i {\alpha_i}^2 \le \left(\sum_i \alpha_i\right)^2$, for $\alpha_i \ge 0$, thus $\sqrt{\sum_i {\alpha_i}^2} \le \sum_i \alpha_i$.

Similarly, we can prove:

$$\begin{aligned}
\frac{u_-}{v} &= \frac{\sum_{\{i|x_{ij'}=1,y_i=0\}}(-z_i)}{\sum_{\{i|x_{ij'}=1\}} z_i \cdot (1-z_i)} && (4.46)\\
&\geq \frac{\sum_{\{i|x_{ij'}=1,y_i=0\}}(-z_i)}{\sum_{\{i|x_{ij'}=1,y_i=0\}} z_i \cdot (1-z_i)} && (4.47)\\
&= -\frac{\sum_{\{i|x_{ij'}=1,y_i=0\}} \frac{1}{1-z_i} \cdot z_i \cdot (1-z_i)}{\sum_{\{i|x_{ij'}=1,y_i=1\}} z_i \cdot (1-z_i)} && (4.48)\\
&\geq -\frac{\sqrt{\sum_{\{i|x_{ij'}=1,y_i=0\}} \frac{1}{(1-z_i)^2}} \cdot \sqrt{\sum_{\{i|x_{ij'}=1,y_i=0\}} {z_i}^2 \cdot (1-z_i)^2}}{\sum_{\{i|x_{ij'}=1,y_i=0\}} z_i \cdot (1-z_i)} && (4.49)\\
&\geq -\sqrt{\sum_{\{i|x_{ij'}=1,y_i=0\}} \frac{1}{(1-z_i)^2}} \cdot \frac{\sqrt{\sum_{\{i|x_{ij'}=1,y_i=0\}} {z_i}^2 \cdot (1-z_i)^2}}{\sum_{\{i|x_{ij'}=1,y_i=0\}} z_i \cdot (1-z_i)} && (4.50)\\
&\geq -\sqrt{\sum_{\{i|x_{ij'}=1,y_i=0\}} \frac{1}{(1-z_i)^2}} && (4.51)\\
&\geq -\sqrt{\sum_{\{i|\mathbf{x_{ij}}=\mathbf{1},y_i=0\}} \frac{1}{(1-z_i)^2}} && (4.52)
\end{aligned}$$

Thus we have:

$$-\sqrt{\sum_{\{i|\mathbf{x_{ij}}=\mathbf{1},y_i=0\}} \frac{1}{(1-z_i)^2}} \leq \frac{u}{v} \leq \sqrt{\sum_{\{i|\mathbf{x_{ij}}=\mathbf{1},y_i=1\}} \frac{1}{{z_i}^2}}$$

or in a more compact form:

$$\left|\frac{\frac{\partial l}{\partial \beta_{j'}}(\beta)}{\frac{\partial^2 l}{\partial {\beta_{j'}}^2}(\beta)}\right| = \left|\frac{u}{v}\right| \leq max\left\{\sqrt{\sum_{\{i|\mathbf{x_{ij}}=\mathbf{1},y_i=1\}} \frac{1}{{z_i}^2}}, \sqrt{\sum_{\{i|\mathbf{x_{ij}}=\mathbf{1},y_i=0\}} \frac{1}{(1-z_i)^2}}\right\}$$

We can thus proceed with the Newton update :

$$\beta^{new} = \beta^{old} + \frac{\frac{\partial l}{\partial \beta_j}(\beta)}{\frac{\partial^2 l}{\partial {\beta_j}^2}(\beta)}$$

always going in the direction of the coordinate with largest magnitude for the newton update.

4.6.1 Empirical Results using SLR with Second Order Information

Our preliminary experiments using second order information did not provide encouraging results. Using the Newton update for selecting features in each optimization iteration, lead to the selection of many noisy features, therefore resulting in low classification quality. We think this is due to the fact that the information captured in the second order derivative and therefore in the Newton update (the ratio of the gradient value and the second derivative) led to the selection of very rare features, therefore resulting in high precision, but very low recall. For the future we plan to further investigate the effect of using second order information with the SLR algorithm.

4.7 Conclusion

In this chapter we presented 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 gave theoretical bounds which quantify the “goodness” of the gradient for each n-gram candidate given its length-(n-1) prefix. We show that by using the proposed bounds, we can efficiently work with variable-length n-gram features both at the word-level and the character-level.

We presented experiments that compare our *SLR* method against the state-of-the-art classifiers *BBR* (a logistic regression method) [GLM06], SVM^{perf} [Joa06] and the latest implementation of sparse SVM, LIBLINEAR (SVM) [HCL$^+$08, LWK08, Lin08, FCH$^+$08]. We showed that while *SLR* achieves similar F1/AUC results to those of the state-of-the-art methods, it is typically more than one order of magnitude faster than its opponents.

We applied the above models to spam filtering, a text classification application of critical importance. This allowed us to study all models on a large real-world dataset. We observed that our model delivers classification results comparable to the best results of the other techniques, but at a much lower memory and running time cost.

With the method presented in this chapter we studied 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 arbitrarily sized, rather than restricted to a hypothesized “good” size. This work opens interesting research directions. Given the flexibility of our model for learning variable-length n-gram patterns, this model could be applied to supervised information extraction in order to learn patterns indicative of binary relations such as *Saarbruecken locatedIn Germany*. Additionally, our technique could be applied to other domains such as gene sequence classification, where mining variable-length sequences is of particular importance.

The same theoretical results presented in this chapter apply directly to trees or graph representations (for example for XML documents), rather than sequences, with only some implementation modifications (e.g., the process of enumeration of candidate features and search space traversal is a bit more sophisticated for trees and graphs). This is true because the simple monotonicity property needed by our proofs holds also in the case of more complex structures such as trees and graphs.

Gradient projection approaches similar to those used in SLR could be applied to other learning problems such as least squares regression. Furthermore, our current SLR technique could be treated as a black-box feature selection algorithm and used as a starting point for other classifiers or even more sophisticated quasi-Newton learning methods.

Chapter 5

Conclusion

5.1 Summary

In this thesis we studied the problem of learning text classifiers when training data is sparse using statistical learning techniques. We focused on two different approaches.

The first approach, coined the **Inductive/Transductive Latent Model (ILM/TLM)**, is a new generative model for inductive and transductive learning which can take advantage of other sources of knowledge besides the small labeled training set. ILM/TLM uses background ontologies, available unlabeled data and encyclopedia, in order to gather additional information about feature distributions, relationships between word-and-concepts and phrases that express certain concepts. We designed different building blocks for the generative model, which can be turned on or off depending on how much we trust certain parameter configurations. For example we can turn on a Dirichlet prior on the model parameters, if we are fairly confident that the background knowledge resources are rich enough to provide robust information on the model structure and parameters. We showed experiments comparing ILM/TLM to state-of-the-art classifiers. For small training data, our model outperforms its competitors. Additionally ILM/TLM can take advantage of re-training techniques, in order to boost its performance.

The second approach, coined **Structured Logistic Regression (SLR)**, is an efficient algorithm for learning logistic regression with variable-length n-grams. Although the dimensionality of the feature space poses efficiency challenges, we showed that by transforming simple optimization schemes into search-and-pruning strategies we can deliver an efficient and accurate learning algorithm. Experimental results on various datasets demonstrated that using variable-length n-grams as features, is beneficial for classification accuracy. At the same time, SLR is one order of magnitude faster than state-of-the-art classifiers. SLR benefits from the fact that it does not need to represent the feature space explicitly, but it rather implicitly searches for the best n-gram features in the huge feature space. We investigated the application of SLR to a fundamental text classification application: spam filtering. We demonstrated the scalability of our technique on a large real-world dataset of spam e-mails. We additionally showed that treating text as a sequence of bytes and learning with features at the level of byte-subsequences is beneficial for spam filtering. We analyzed various extensions of SLR and suggested potential applications to other domains such as biological sequence classification.

5.2 Future Research Directions

We have seen that the ILM/TLM model is particularly promising for small training sets. This suggests using this approach for automatically producing larger training sets. One can start with a few labeled samples and use TLM with potentially unlabeled data from related domains in order to create a larger labeled training set. Furthermore, TLM can be used in re-training or active learning schemes, where the classifier can be boosted by learning from its own predictions or from labels provided by a human user.

SLR was showcased for sequences in this thesis, but an interesting research direction is its usage with trees and graphs structures. The same theoretical framework directly applies to these complex structures. Additionally, during the search for a candidate feature, one can envision a more flexible matching mechanism, so that features that are within a certain distance from the original feature are considered during the learning steps. For example, instead of producing sequences of consecutive tokens $x\ y\ z$, we could consider sequences with gaps such as $x\ \%\ z$, where % stands for any token. This could particularly benefit applications in information extraction from natural language text or the mining of biological data. Certain parameter combinations could also be avoided by considering prior distributions on the logistic regression parameters. These priors could be learned from available data from related domains. Future work could consider combining the strengths of the two learning paradigms into hybrid models that could benefit from the advantages of both schemes.

With the techniques presented in this thesis we aim at advancing the technologies for automatically and efficiently building large training sets, thereby reducing the need for spending human computation on this task.

List of Figures

List of Algorithms

List of Tables

Bibliography

[AKA$^+$02] K. Abe, S. Kawasoe, T. Asai, H. Arimura, and S. Arikawa. Optimized substructure discovery for semi-structured data. In *Proceedings of the Conference on Principles and Practice of Knowledge Discovery in Databases.* Springer–Verlag, 2002.

[BBL06] M.-F. Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. In *ICML '06: Proceedings of the 23rd International Conference on Machine Learning*, pages 65–72, New York, NY, USA, 2006. ACM.

[BBS07] S. Bickel, M. Brückner, and T. Scheffer. Discriminative learning for differing training and test distributions. In *ICML '07: Proceedings of the 24th International Conference on Machine Learning*, pages 81–88, New York, NY, USA, 2007. ACM.

[BC01] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proc. 18th International Conference on Machine Learning*, pages 19–26. Morgan Kaufmann, San Francisco, CA, 2001.

[BCM05] R. Basili, M. Cammisa, and A. Moschitti. A semantic kernel to exploit linguistic knowledge. In Stefania Bandini and Sara Manzoni, editors, *AI*IA 2005: Advances in Artificial Intelligence — Proceedings of the 9th Congress of the Italian Association for Artificial Intelligence, September 21-32, 2005, Milan, Italy*, volume 3673 of *Lecture Notes in Computer Science*, pages 290–302. Springer, Berlin–Heidelberg, Germany, 2005.

[BCW90] T. C. Bell, J. G. Cleary, and I. H. Witten. *Text Compression.* Prentice-Hall, 1990.

[Ben99] K. P. Bennett. Combining support vector and mathematical programming methods for classification. *Advances in kernel methods: support vector learning*, pages 307–326, 1999.

[BFC$^+$06] A. Bratko, B. Filipič, G. V. Cormack, T. R. Lynam, and B. Zupan. Spam filtering using statistical data compression models. *Journal of Machine Learning Research*, 7:2673–2698, 2006.

[BGJT04] D.M. Blei, T.L. Griffiths, M.I. Jordan, and J.B. Tenenbaum. Hierarchical Topic Models and the Nested Chinese Restaurant Process. *Advances in Neural Information Processing Systems 16*, 2004.

[BH04a] S. Bloehdorn and A. Hotho. Boosting for text classification with semantic features. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Mining for and from the Semantic Web Workshop*, pages 70–87, 2004.

[BH04b] S. Bloehdorn and A. Hotho. Text classification by boosting weak learners based on terms and concepts. In *International Conference on Data Mining*, pages 331–334, 2004.

[BHS06] M. Brückner, P. Haider, and T. Scheffer. Highly scalable discriminative spam filtering. In *Proceedings of the 15th Text REtrieval Conference (TREC 2006)*, Gaithersburg, USA, 2006.

[BLRR04] A. Blum, J. Lafferty, M. R. Rwebangira, and R. Reddy. Semi-supervised learning using randomized mincuts. In *ICML '04: Proceedings of the twenty-first International Conference on Machine Learning*, page 13, New York, NY, USA, 2004. ACM.

[BM98] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT' 98: Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, New York, NY, USA, 1998. ACM.

[BM07] S. Bloehdorn and A. Moschitti. Exploiting structure and semantics for expressive text kernels. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on Information and Knowledge Management*, pages 861–864, New York, NY, USA, 2007. ACM.

[BMP00] R. Basili, A. Moschitti, and M. T. Pazienza. Language-sensitive text classification. In *Proceeding of RIAO-00, 6th International Conference "Recherche d'Information Assistee par Ordinateur"*, pages 331–343, Paris, FR, 2000.

[BNJ03] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[Bou02] O. Bousquet. Transductive learning: Motivation, models, algorithms. University of New Mexico, Albuquerque, USA, January 2002.

[BS07] S. Bickel and T. Scheffer. Dirichlet-enhanced spam filtering based on biased samples. In *Advances in Neural Information Processing Systems 19*, pages 161–168. MIT Press, 2007.

[BT04] G. Bouchard and B. Triggs. The tradeoff between generative and discriminative classifiers. In *IASC International Symposium on Computational Statistics (COMPSTAT)*, pages 721–728, Prague, August 2004.

[Car97] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[CBM06] M. Cammisa, R. Basili, and A. Moschitti. A semantic kernel to classify texts with very few training examples. *International Journal of Computing and Informatics*, 2006.

[CC95] V. Castelli and T. M. Cover. On the exponential value of labeled samples. *Pattern Recongnition Letters*, 16(1):105–111, 1995.

[CCC03] F. G. Cozman, I. Cohen, and M. C. Cirelo. Semi-supervised learning of mixture models. In *ICML-03, 20th International Conference on Machine Learning*, pages 99–106, 2003.

[CCZ06] O. Chapelle, M. Chi, and A. Zien. A continuation method for semi-supervised svms. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 185–192, New York, NY, USA, 2006. ACM.

[CH03] L. Cai and T. Hofmann. Text categorization by boosting automatically extracted concepts. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 182–189, New York, NY, USA, 2003. ACM.

[Cha03] S. Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufman, San Francisco, 2003.

[CJ01] A. Corduneanu and T. Jaakkola. Stable Mixing of Complete and Incomplete Informaation. Technical report, CSAIL, MIT, 2001.

[CM07] M. Ceci and D. Malerba. Classifying web documents in a hierarchy of categories: a comprehensive study. *Journal of Intelligent Information Systems*, 28(1):37–78, 2007.

[CNM06] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *ICML '06: Proceedings of the 23rd International Conference on Machine Learning*, pages 161–168, New York, NY, USA, 2006. ACM.

[CSK07] O. Chapelle, V. Sindhwani, and S. Keerthi. Branch and bound for semi-supervised support vector machines. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, Cambridge, MA, 2007.

[CSWB06] R. Collobert, F. Sinz, J. Weston, and Léon Bottou. Large scale transductive svms. *Journal of Machine Learning Research*, 7:1687–1712, 2006.

[CSZ06] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[CT97] J. G. Cleary and W. J. Teahan. Unbounded length contexts for ppm. *Computer Journal*, 3(40):67–75, 1997.

[CWD03] Y. Chen, G. Wang, and S. Dong. Learning with progressive transductive support vector machine. *Pattern Recongnition Letters*, 24(12):1845–1855, 2003.

[CYHH07] H. Cheng, X. Yan, J. Han, and C. Hsu. Discriminative frequent pattern analysis for effective classification. In *Proceedings of the International Conference on Data Engineering*, pages 716–725, 2007.

[CYM08] M. Chang, W. Yih, and C. Meek. Partitioned logistic regression for spam filtering. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 97–105, New York, NY, USA, 2008. ACM.

[CZ05] O. Chapelle and A. Zien. Semi–supervised classification by low density separation. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, 2005.

[Das04] S. Dasgupta. Analysis of a greedy active learning strategy. In *Neural Information Processing Systems (NIPS)*, pages 337–344, 2004.

[DDF$^+$90] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.

[DH08] S. Dasgupta and D. Hsu. Hierarchical sampling for active learning. In *ICML '08: Proceedings of the 25th International Conference on Machine Learning*, pages 208–215, New York, NY, USA, 2008. ACM.

[DHM08] S. Dasgupta, D. Hsu, and C. Monteleoni. A general agnostic active learning algorithm. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 353–360. MIT Press, Cambridge, MA, 2008.

[DLM$^+$06] A. Dayanik, D. D. Lewis, D. Madigan, V. Menkov, and A. Genkin. Constructing informative prior distributions from domain knowledge in text classification. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 493–500, New York, NY, USA, 2006. ACM.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[Dmo] Open directory project. http://www.dmoz.org/.

[DPHS98] S. T. Dumais, J. Platt, D. Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In Georges Gardarin, James C. French, Niki Pissinou, Kia Makki, and Luc Bouganim, editors, *Proceedings of CIKM-98, 7th ACM International Conference on Information and Knowledge Management*, pages 148–155, Bethesda, US, 1998. ACM Press, New York, US.

[DSSS04] O. Dekel, S. Shalev-Shwartz, and Y. Singer. The power of selective memory: Self-bounded learning of prediction suffix trees. In *Proceedings of Advances in Neural Information Processing Systems*, Vancouver, Canada, 2004.

[DYXY07] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu. Boosting for transfer learning. In *ICML '07: Proceedings of the 24th International Conference on Machine Learning*, pages 193–200, New York, NY, USA, 2007. ACM.

[DYXY08] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu. Self-taught clustering. In *ICML '08: Proceedings of the 25th International Conference on Machine Learning*, pages 200–207, New York, NY, USA, 2008. ACM.

[EHJT04] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2)(4047499), 2004.

[EYG05] R. El-Yaniv and L. Gerzon. Effective transductive learning via objective model selection. *Pattern Recongnition Letters*, 26(13):2104–2115, 2005.

[Faw04] T. Fawcett. Roc graphs: Notes and practical considerations for researchers, 2004.

[FC04] G. Forman and I. Cohen. Learning from little: comparison of classifiers given little training. In *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 161–172, New York, NY, USA, 2004. Springer-Verlag New York, Inc.

[FCH$^+$08] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[FCW00] E. Frank, C. Chui, and I. H. Witten. Text categorization using compression models. In James A. Storer and M. Cohn, editors, *Proceedings of DCC-00, IEEE Data Compression Conference*, pages 200–209, Snowbird, US, 2000. IEEE Computer Society Press, Los Alamitos, US.

[Fel99] C. Fellbaum. *WordNet: An Electronic Lexical Database*. Cambridge: MIT Press, 1999.

[FM01] E. M. Felcher and A. L. McGill. The role of taxonomic and goal-derived product categorization in, within, and across category judgments. *J. Wiley & Sons*, 18(8):865–887, August 2001.

[GLM06] A. Genkin, D. Lewis, , and D. Madigan. Large-scale bayesian logistic regression for text categorization. Technical Report no-06-05-18, DIMACS, 2006.

[GM05] E. Gabrilovich and S. Markovitch. Feature generation for text categorization using world knowledge. In *Proceedings of The Nineteenth International Joint Conference for Artificial Intelligence*, pages 1048–1053, Edinburgh, Scotland, 2005.

[GM06a] E. Gabrilovich and S. Markovitch. Overcoming the brittleness bottleneck using wikipedia: Enhancing text categorization with encyclopedic knowledge. In *AAAI*, 2006.

[GM06b] A.-M. Giuglea and A. Moschitti. Semantic role labeling via framenet, verbnet and propbank. In *ACL*, 2006.

[GM07] E. Gabrilovich and S. Markovitch. Harnessing the expertise of 70,000 human editors: Knowledge-based feature generation for text categorization. *Journal of Machine Learning Research*, 8:2297–2345, 2007.

[Goo01] J. Goodman. A bit of progress in language modeling. In *Technical report*. Microsoft Research, 2001.

[GSD05] A. Gliozzo, C. Strapparava, and I. Dagan. Investigating unsupervised learning for text categorization bootstrapping. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 129–136, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[GSD06] G. Getz, N. Shental, and E. Domany. Semi-supervised learning – a statistical physics approach. *CoRR*, abs/cs/0604011, 2006. informal publication.

[GZ06] A. B. Goldberg and X. Zhu. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *HLT-NAACL 2006 Workshop on Textgraphs: Graph-based Algorithms for Natural Language Processing*, 2006.

[HCL$^+$08] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the Twenty Fifth International Conference on Machine Learning (ICML)*, 2008. Software available at `http://www.csie.ntu.edu.tw/~cjlin/liblinear`.

[HF95] D. Holmes and R. Forsyth. The federalist revisited: New directions in autorship atttribution. *Literary and Linguistic Computing*, 2(10):111–127, 1995.

[Hof01] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Journal of Machine Learning Research*, 42(1-2):177–196, 2001.

[HTF03] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer Series in Statistics, 2003.

[HTT03] Ji He, Ah-Hwee Tan, and Chew-Lim Tan. On machine learning methods for chinese document categorization. *Applied Intelligence*, 18(3):311–322, 2003.

[IBW08] G. Ifrim, G. Bakir, and G. Weikum. Fast logistic regression for text categorization with variable-length n-grams. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 354–362, New York, NY, USA, 2008. ACM.

[Ifr05] G. Ifrim. A bayesian learning approach to concept-based document classification. Master's thesis, Universität des Saarlandes, February 2005.

[ITW05] G. Ifrim, M. Theobald, and G. Weikum. Learning word-to-concept mappings for automatic text classification. In Luc De Raedt and Stefan Wrobel, editors, *Proceedings of the 22nd International Conference on Machine Learning - Learning in Web Search (LWS 2005)*, pages 18–26, Bonn, Germany, 2005.

[IW06] G. Ifrim and G. Weikum. Transductive learning for text classification using explicit knowledge models. In *Proceedings of the Conference on Principles and Practice of Knowledge Discovery in Databases*, Springer Lecture Notes in Artificial Intelligence, pages 223–234, Berlin, Germany, 2006.

[JH99] T. S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems 11*, pages 487–493. MIT Press, 1999.

[JM00] L. S. Jensen and T. Martinez. Improving text classification by using conceptual and contextual features. In *In Proceedings of the Workshop on Text Mining at the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 101–102, 2000.

[Joa98] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE. Published in the “Lecture Notes in Computer Science” series, number 1398.

[Joa99a] T. Joachims. Making large-scale support vector machine learning practical. *Advances in kernel methods: support vector learning*, pages 169–184, 1999.

[Joa99b] T. Joachims. Transductive inference for text classification using support vector machines. In *ICML ’99: Proceedings of the Sixteenth International Conference on Machine Learning*, pages 200–209, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[Joa03] T. Joachims. Transductive learning via spectral graph partitioning. In *International Conference on Machine Learning (ICML)*, pages 290–297, 2003.

[Joa06] T. Joachims. Training linear SVMs in linear time. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–226, New York, NY, 2006. ACM Press.

[JYZH03] R. Jin, R. Yan, J. Zhang, and A. Hauptmann. A faster iterative scaling algorithm for conditional exponential model. In *Proceedings of the International Conference on Machine Learning*, 2003.

[KB06] A. C. König and E. Brill. Reducing the human overhead in text categorization. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 598–603, New York, NY, USA, 2006. ACM.

[KB07] T. Kuroiwa and S. Bhalla. Dynamic personalization for book recommendation system using web services and virtual library enhancements. In *CIT '07: Proceedings of the 7th IEEE International Conference on Computer and Information Technology*, pages 212–217, Washington, DC, USA, 2007. IEEE Computer Society.

[KM03] P. Komarek and A. Moore. Fast robust logistic regression for large sparse datasets with binary outputs. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, New York, NY, 2003.

[KM04] T. Kudo and Y. Matsumoto. A boosting algorithm for classification of semi-structured text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 301–308, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[KNS97] B. Kessler, Geoff Nunberg, and Hinrich Schütze. Automatic detection of text genre. In Philip R. Cohen and Wolfgang Wahlster, editors, *Proceedings of ACL-97, 35th Annual Meeting of the Association for Computational Linguistics*, pages 32–38, Madrid, ES, 1997. Morgan Kaufmann Publishers, San Francisco, US.

[KPA08] A. Kosmopoulos, G. Paliouras, and I. Androutsopoulos. Adaptive spam filtering using only naive bayes text classifiers. In *Proceedings of the Fifth Conference on Email and Anti-Spam (CEAS)*, 2008.

[Kra05] W. Kraaij. Variations on language modeling for information retrieval. *SIGIR Forum*, 39(1):61–61, 2005.

[KSG$^+$03] Y. Khopkar, A. Spink, C. L. Giles, P. Shah, and S. Debnath. Search engine personalization: An exploratory study. *First Monday*, 8(7), 2003.

[KSI$^+$08a] G. Kasneci, F. M. Suchanek, G. Ifrim, Shady Elbassuoni, Maya Ramanath, and G. Weikum. Naga: Harvesting, searching and ranking knowledge. In *ACM International Conference on Management Of Data (SIGMOD/PODS 2008)*. ACM, 2008.

[KSI$^+$08b] G. Kasneci, F. M. Suchanek, G. Ifrim, Maya Ramanath, and G. Weikum. NAGA: Searching and Ranking Knowledge. In *24th International Conference on Data Engineering (ICDE 2008)*. IEEE, 2008.

[KT06] C. Kang and J. Tian. A hybrid generative/discriminative bayesian classifier. In *FLAIRS Conference*, pages 562–567, 2006.

[Kud03] T. Kudo. An implementation of freqt (frequent tree miner), 2003. http://chasen.org/∼taku/software/freqt/.

[LB03] Quan Le and Samy Bengio. Noise Robust Discriminative Models. Technical report, 2003.

[LBM06] J. A. Lasserre, C. M. Bishop, and T. P. Minka. Principled hybrids of generative and discriminative models. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society*

Conference on Computer Vision and Pattern Recognition, pages 87–94, Washington, DC, USA, 2006. IEEE Computer Society.

[Lew] D. D. Lewis. Reuters-21578 dataset. http://www. daviddlewis. com/resources/ testcollections/reuters21578/.

[Lew92] D. D. Lewis. *Representation and Learning in Information Retrieval.* PhD thesis, Department of Computer Science, University of Massachusetts, 1992.

[Lew98] D. D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *ECML '98: Proceedings of the 10th European Conference on Machine Learning*, pages 4–15, London, UK, 1998. Springer-Verlag.

[LEW08] J. Luxenburger, Shady Elbassuoni, and G. Weikum. Task-aware search personalization. In *31st Annual International ACM SIGIR Conference*, pages –, Singapore, 2008. ACM.

[Lin08] C. Lin. Liblinear, 2008. `http://mloss.org/software/view/61/`.

[LLLY04] B. Liu, X. Li, W. S. Lee, and P. S. Yu. Text classification by labeling words. In *AAAI-2004*, 2004.

[LM02] Y.-B. Lee and S. H. Myaeng. Text genre classification with genre-revealing and subject-revealing features. In Micheline Beaulieu, Ricardo Baeza-Yates, Sung Hyon Myaeng, and Kalervo Järvelin, editors, *Proceedings of SIGIR-02, 25th ACM International Conference on Research and Development in Information Retrieval*, pages 145–150, Tampere, FI, 2002. ACM Press, New York, US.

[LSST$^+$01] 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.

[LWK08] C.-J. Lin, R. C. Weng, and S. S. Keerthi. Trust region Newton method for large-scale logistic regression. *Journal of Machine Learning Research*, 9:627–650, 2008. Software available at `http://www.csie.ntu.edu.tw/~cjlin/liblinear`.

[MFP06] G. Mühl, L. Fiege, and P. Pietzuch. *Distributed Event-Based Systems.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[MN98a] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on "Learning for Text Categorization*, 1998.

[MN98b] A. McCallum and K. Nigam. Employing em and pool-based active learning for text classification. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 350–358, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[MN99] A. Mccallum and K. Nigam. Text classification by bootstrapping with keywords, em and shrinkage. In *ACL99 - Workshop for Unsupervised Learning in Natural Language Processing*, pages 52–58, 1999.

[Mos03] A. Moschitti. *Natural Language Processing and Text Categorization: a study on the reciprocal beneficial interactions.* PhD thesis, University of Rome Tor Vergata, Rome, Italy., 2003.

[Mos06] A. Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. In *ECML*, pages 318–329, 2006.

[MRS08] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval.* Cambridge University Press, July 2008.

[MS00] C. D. Manning and H. Schuetze. *Foundations of Statistical Natural Language Processing.* MIT Press, Cambridge, 2000.

[NH06] K. Nigam and M. Hurst. Towards a robust metric of polarity. In *Computing Attitude and Affect in Text: Theories and Applications*, 2006.

[Nig01] K. P. Nigam. Using unlabeled data to improve text classification. Technical report, PhD thesis, Carnegie Mellon University., 2001.

[NJ01] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classi ers: A comparison of logistic regression and naive bayes, 2001.

[NJT05] Z.-Y. Niu, D.-H. Ji, and C. L. Tan. Word sense disambiguation using label propagation based semi-supervised learning. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 395–402, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[NMTM00a] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Journal of Machine Learning Research*, 39(2-3):103–134, 2000.

[NMTM00b] K. Nigam, A. K. McCallum, Sebastian Thrun, and Tom M. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2/3):103–134, 2000.

[NW06] J. Nocedal and S. Wright. *Numerical Optimization.* Springer Series in Operation Research and Financial Engineering, 2006.

[PL04] B. Pang and L. Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 271–278, 2004.

[PLV02] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 79–86, 2002.

[PNL05] T. P. Pham, H. T. Ng, and W. S. Lee. Word sense disambiguation with semisupervised learning. In *AAAI-05, The Twentieth National Conference on Artificial Intelligence.*, 2005.

[PS03] F. Peng and D. Schuurmans. Combining naive bayes n-gram and language models for text classification. In F. Sebastiani, editor, *Proceedings of ECIR-03, 25th European Conference on Information Retrieval*, pages 335–350, Pisa, IT, 2003. Springer Verlag.

[PSM07] Ó. Pérez and M. A. Sánchez-Montañés. A new learning strategy for classification problems with different training and test distributions. In *IWANN*, pages 178–185, 2007.

[PSW04] F. Peng, D. Schuurmans, and S. Wang. Augmenting naive bayes text classifier with statistical language models. *Information Retrieval*, 3(7):317–345, 2004.

[RBL$^+$07] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *ICML '07: Proceedings of the 24th International Conference on Machine Learning*, pages 759–766, New York, NY, USA, 2007. ACM.

[RK07] D. M. Roy and L. P. Kaelbling. Efficient bayesian task-level transfer learning. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence, Hyderabad, India*, 2007.

[RMKD05] M. T. Rosenstein, Z. Marx, L. P. Kaelbling, and T. G. Dietterich. To transfer or not to transfer. In *Proceedings of the Neural Information Processing Systems Workshop on Transfer Learning*, Whistler, BC., 2005.

[Ros00] R. Rosenfeld. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88(8):1270–1278, 2000.

[RRN$^+$06] R. B. Rao, R. Rosales, S. Niculescu, S. Krishnan, L. Bogoni, X. S. Zhou, and B. Krishnapuram. Mining medical records for computer aided diagnosis, 2006.

[RSNM03] R. Raina, Y. Shen, A. Y. Ng, and A. Mccallum. Classification with hybrid generative/discriminative models. In *Advances in Neural Information Processing Systems 16*. MIT Press, 2003.

[RSTK03] J. Rennie, L. Shih, J. Teevan, and D. Karger. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of ICML-03, 20th International Conference on Machine Learning*, Washington, DC, 2003. Morgan Kaufmann Publishers, San Francisco, US.

[RZ04] G. Rios and H. Zha. Exploring support vector machines and random forests for spam detection. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004. Available: `http://www.ceas.cc/papers-2004/174.pdf`.

[SACY04] C. Siefkes, F. Assis, S. Chhabra, and W. S. Yerazunis. Combining Winnow and orthogonal sparse bigrams for incremental spam filtering. In Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2004)*, volume 3202 of *Lecture Notes in Artificial Intelligence*, pages 410–421. Springer, 2004. Available: `http://www.siefkes.net/papers/winnow-spam.pdf`.

[Sal89] G. Salton. *Automatic text processing: the transformation, analysis, and retrieval of information by computer.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[Sar08] S. Sarawagi. Information Extraction. *Foundations and Trends in Databases*, 2(1), 2008.

[Seb01] F. Sebastiani. Organizing and using digital libraries by automated text categorization. In *Luciana Bordoni and Giovanni Semeraro (eds.), Proceedings of the AI*IA Workshop on Artificial Intelligence for Cultural Heritage and Digital Libraries*, pages 93–94, 2001.

[Seb02] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[SIW06] F. M. Suchanek, G. Ifrim, and G. Weikum. Combining linguistic and statistical analysis to extract relations from web documents. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 712–717, New York, NY, USA, 2006. ACM.

[SJ01] M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In T. Dietterich et al., editor, *Proceedings of Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001. `http://www.ai.mit.edu/people/szummer/`.

[SK03] S. K. Shevade and S. S. Keerthi. A simple and eficient algorithm for gene selection using sparse logistic regression. *Bioinformatics*, 19:2246–2253, 2003.

[SKC06] V. Sindhwani, S. S. Keerthi, and O. Chapelle. Deterministic annealing for semi-supervised kernel machines. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 841–848, New York, NY, USA, 2006. ACM.

[SKF00] E. Stamatatos, G. Kokkinakis, and N. Fakotakis. Automatic text categorization in terms of genre and author. *Comput. Linguist.*, 26(4):471–495, 2000.

[SKM07] M. Sugiyama, M. Krauledat, and K.-R. Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8:985–1005, 2007.

[SKS07] B. Stein, M. Koppel, and E. Stamatatos. Plagiarism analysis, authorship identification, and near-duplicate detection pan'07. *SIGIR Forum*, 41(2):68–71, 2007.

[SKW07] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW '07: Proceedings of the 16th International Conference on World Wide Web*, pages 697–706, New York, NY, USA, 2007. ACM.

[SKW08] F. Suchanek, G. Kasneci, and G. Weikum. Yago - a large ontology from wikipedia and wordnet. *Elsevier Journal of Web Semantics*, 2008.

[SM99] S. Scott and S. Matwin. Feature engineering for text classification. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 379–388, Bled, SL, 1999. Morgan Kaufmann Publishers, San Francisco, US.

[SM00] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 2000.

[SMA] Smart stopwords list. http://www.lextek.com/manuals/onix/stopwords2.html.

[spaa] Guenter spam trap. http://untroubled.org/spam/.

[spab] Trec 2005 and trec 2006 spam challenge. http://plg.uwaterloo.ca/ gvcormac/treccorpus/about.html.

[SS02] B. Scholkopf and A. J. Smola. *Learning with Kernels.* MIT Press, Cambridge, MA, 2002.

[SS04] S. Staab and R. Studer. *Handbook on Ontologies.* Springer, Berlin, 2004.

[SS07] A. J. Storkey and M. Sugiyama. Mixture regression for covariate shift. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1337–1344. MIT Press, Cambridge, MA, 2007.

[TJBB06] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[TKK00] S. Tong, D. Koller, and P. Kaelbling. Support vector machine active learning with applications to text classification. In *Journal of Machine Learning Research*, pages 999–1006. Morgan Kaufmann, 2000.

[tra] Ebtl: Effective bayesian transfer learning project. http://www.cs.berkeley.edu/ russell/ebtl/.

[TSW03] M. Theobald, R. Schenkel, and G. Weikum. Exploiting structure, annotation, and ontological knowledge for automatic classification of XML data. In V. Christophides and J. Freire, editors, *6th International Workshop on the Web and Databases (WebDB-03)*, pages 1–6, San Diego, USA, 2003. OGI School of Science and Engineering / CSE. Acceptance ratio 1:4.

[Vap98] V. Vapnik. *Statistical learning theory.* Wiley, 1998.

[WBB$^+$03] J. Wiebe, E. Breck, C. Buckley, C. Cardie, P. Davis, B. Fraser, D. Litman, D. Pierce, Ellen Riloff, Theresa Wilson, D. Day, and Mark Maybury. Recognizing and organizing opinions expressed in the world press. In *Proceedings of the AAAI Spring Symposium on New Directions in Question Answering*, 2003.

[WD04] P. Wu and T. G. Dietterich. Improving svm accuracy by training on auxiliary data sources. In *ICML '04: Proceedings of the twenty-first International Conference on Machine Learning*, page 110, New York, NY, USA, 2004. ACM.

[WD08] P. Wang and C. Domeniconi. Building semantic kernels for text classification using wikipedia. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 713–721, New York, NY, USA, 2008. ACM.

[WHW08] F. Wu, R. Hoffmann, and D. S. Weld. Information extraction from wikipedia: moving down the long tail. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 731–739, New York, NY, USA, 2008. ACM.

[WHZ$^+$07] P. Wang, J. Hu, H.-J. Zeng, L. Chen, and Z. Chen. Improving text classification by using encyclopedia knowledge. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 332–341, 2007.

[WHZC08] P. Wang, J. Hu, H.-J. Zeng, and Z. Chen. Using wikipedia knowledge to improve text classification. *Knowledge and Information Systems*, 2008.

[Wik] Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Main_Page.

[WP03] G. M. Weiss and F. Provost. Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 19:315–354, 2003.

[WW08] F. Wu and D. S. Weld. Automatically refining the wikipedia infobox ontology. In *WWW '08: Proceeding of the 17th International Conference on World Wide Web*, pages 635–644, New York, NY, USA, 2008. ACM.

[XJZ$^+$08] Z. Xu, R. Jin, J. Zhu, I. King, and M. Lyu. Efficient convex relaxation for transductive support vector machine. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1641–1648. MIT Press, Cambridge, MA, 2008.

[XS05] L. Xu and D. Schuurmans. Unsupervised and semi-supervised multi-class support vector machines. *AAAI*, 2005.

[YL99] Y. Yang and X. Liu. A re-examination of text categorization methods. In Marti A. Hearst, Fredric Gey, and R. Tong, editors, *Proceedings of SIGIR-99, 22nd ACM*

International Conference on Research and Development in Information Retrieval, pages 42–49, Berkeley, US, 1999. ACM Press, New York, US.

[YNBN03] J. Yi, T. Nasukawa, R. Bunescu, and W. Niblack. Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, 2003.

[YP97] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In Douglas H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

[Zak02] 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.

[Zel02] S. Zelikovitz. *Using background knowledge to improve text classification.* PhD thesis, Rutgers University, New Brunswick, NJ, USA, 2002. Director-Haym Hirsh.

[ZG02] X. Zhu and Z. Ghahramani. Towards semisupervised classification with markov random fields. Technical report, Technical Report CMU-CALD-02-106. Carnegie Mellon University., 2002.

[ZGL03] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *International Conference on Machine Learning*, pages 912–919, 2003.

[ZH00] S. Zelikovitz and Haym Hirsh. Improving short text classification using unlabeled background knowledge to assess document similarity. In Pat Langley, editor, *Proceedings of ICML-00, 17th International Conference on Machine Learning*, pages 1183–1190, Stanford, US, 2000. Morgan Kaufmann Publishers, San Francisco, US.

[ZH01] S. Zelikovitz and Haym Hirsh. Improving text classification with lsi using background knowledge. In *IJCAI01 Workshop Notes on Text Learning: Beyond Supervision*, pages 113–118, 2001.

[ZH02] S. Zelikovitz and Haym Hirsh. Integrating background knowledge into nearest-neighbor text classification. In *ECCBR '02: Proceedings of the 6th European Conference on Advances in Case-Based Reasoning*, pages 1–5, London, UK, 2002. Springer-Verlag.

[Zha04] Y. Zhang. Using bayesian priors to combine classifiers for adaptive filtering. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 345–352, New York, NY, USA, 2004. ACM.

[Zhu05] X. Zhu. *Semi-supervised learning with graphs.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2005. Chair-J. Lafferty and Chair-Ronald Rosenfeld.

[Zhu08] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2008.

[ZJ08] Jeff Zabin and Alex Jefferies. Social media monitoring and analysis: Generating consumer insights from online conversation. Aberdeen Group Benchmark Report, January 2008.

[ZK05] Y. Zhao and G. Karypis. Topic-driven clustering for document datasets. In *Siam Conference on Data Mining*, 2005.

[ZL06] D. Zhang and W. S. Lee. Extracting key-substring-group features for text classification. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 474–483, New York, NY, USA, 2006. ACM.

[ZO00] T. Zhang and F. J. Oles. A probability analysis on the value of unlabeled data for classification problems. In *Proc. 17th International Conference on Machine Learning*, pages 1191–1198, 2000.

[ZO01] T. Zhang and F. Oles. Text categorization based on regularized linear classifiers. *Information Retrieval*, 4(1):5–31, 2001.

[ZV06] Z. Zhang and Balaji Varadarajan. Utility scoring of product reviews. In *Proceedings of the ACM SIGIR Conference on Information and Knowledge Management (CIKM)*, pages 51–57, 2006.

[ZWZ08] B. Zhao, F. Wang, and C. Zhang. Cuts3vm: a fast semi-supervised svm algorithm. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 830–838, New York, NY, USA, 2008. ACM.

[ZZ06] F. Zhu and X. (M.) Zhang. The influence of online consumer reviews on the demand for experience goods: The case of video games. In *International Conference on Information Systems (ICIS)*, 2006.