Text Categorization for Intellectual Property

Comparing Balanced Winnow with SVM on Different Document Representations

Katrien M.B. Beuls

MSc Speech and Language Processing
University of Edinburgh

2009

Thanks to Bernhard Pflugfelder,
for his help in the experimental setup of this study

Abstract. This study investigates the effect of training different categorization algorithms on various patent document representations. The automation of knowledge and content management in the intellectual property domain has been experiencing a growing interest in the last decade [**Cai and Hofmann, 2004, Fall et al., 2003, Koster et al., 2003, Krier and Zaccà, 2002**], since the first patent classification system was presented in 1999 by Larkey [**Larkey, 1999**]. Typical applications of patent classification systems are: (1) the automatic assignment of a new patent to the group of patent examiners concerned with the topic, (2) the search for prior art in fields similar to the incoming patent application and (3) the reclassification of patent specifications. By means of machine learning techniques, a collection of 1 270 185 patents is used to build a classifier that is able to classify documents with varyingly large feature spaces. The two algorithms that are compared are Balanced Winnow and Support Vector Machines (SVMs). A previous study [**Zhang, 2000**] found that Winnow achieves a similar accuracy to SVM but it is much faster as the execution time for Winnow is linear in the number of terms and the number of classes. This primary finding is verified on a feature space 100 times the size using patent documents instead of news paper articles. Results show that SVM outperforms Winnow considerably on all considered measures. Moreover, SVM is found to be a much more robust classifier than Winnow. The parameter tuning that was carried out for both algorithms confirms this result.

Acknowledgements

This thesis was composed in the offices of the Information Retrieval Facility in Vienna. Without the warm support of my colleagues during the three months I spent there, this document would not have been the same. I am especially indebted to Bernhard Pflugfelder, working for Matrixware, with whom I implemented all the experiments that were carried out and who was always there to answer my questions regarding text categorization and technical details of the machine learning methods used in my research. Moreover, the weekly meetings with my supervisor, Professor Allan Hanbury, were very inspiring as he perfectly could direct my attention to the issues that deserved more attention. Thanks for the professional and warm support.

Just over twelve months ago I finished a MA thesis in Linguistics at the University of Leuven in Belgium. The change from linguistics to computer science required a strong will and a strict discipline from my side but without the support of my Edinburgh friends and especially my fellow MSc students Anna, Angeliki, Phil, Kevin and Erich, it would have been even harder. Thanks for making it such a great year!

Thanks to a scholarship from the Rotary District 1630 and sponsoring of my family I could fully concentrate on my studies without having to worry about the financial aspect of it. This contributed largely to the successful completion of my postgraduate degree. I especially want to thank my parents, Bart and Renée, for believing in me and giving me the opportunity to go abroad and start this degree. It has redirected my attention from theoretical linguistics to an interdisciplinary field that unites knowledge of both linguistics and computer science and has made me more confident in undertaking further academic research. *Bedankt!*

A final special thanks to Alex, for the many visits to Edinburgh and the enthusiasm in his support.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text. This work has not been submitted for any other degree or professional qualification except as specified.

(*Katrien Beuls*)

Contents

CHAPTER 1

Introduction

With large numbers of new patents coming in every day to patent offices all over the world, the automation of some standard every day tasks can save patent officers' valuable time. As IP reviewers have different levels of background, assigning categories to a new patent document often involves multiple categorization steps. Therefore, specific reviewers are responsible to select the actual IP reviewers who are able to decide on the patent categories related to the described technology. An automatic assignment system would thus be an interesting application to speed up the classification process. Especially in small and medium sized offices where there are not enough experts to cover all domains computer-assisted patent categorization can help to find the correct IPC categories of a new patent application. Not only finding the appropriate niche in the patent taxonomy can be a task for a patent classification system, also prior art search relies on an accurate patent classification. Every granted patent must contain a section on prior art. The prior art encloses all information that has publicly appeared in any form before a given date that might be relevant to a patent's claims of originality.

Patents are text documents and therefore fall within the text categorization problem. Text categorization is often defined as the task of assigning a Boolean value to each pair $\langle d_j, c_i \rangle \in D \times C$, where D is a domain of documents and $C = \{c_1, \ldots, c_{|C|}\}$ is a set of pre-defined categories [**Sebastiani, 2002**]. There are however, specific properties of patents that distinguish them from the prototypical documents in text categorization (e.g. news paper articles). Patent documents contain for instance few named entities that are in text categorization key terms to assign a category to a document. Apart from that, the vocabulary of patent documents is in fact much larger than that of newspaper articles, due to the use of technical terminology such as chemical formulas. Depending on the used taxonomy and the level the patent documents are categorised at, there exist patents that can be classified into one or more of 70 000 different categories (e.g. IPC taxonomy, total number of groups). In previous text categorization experiments [**Dagan et al., 1997, Koster et al., 2003**], the number of categories tested is seldom above a hundred. This is because the effectiveness of all algorithm drops as the number of categories increases. The rate at which the accuracy decreases is

strongly dependent on (1) the corpus distribution , (2) the terms that were used for training and (3) the way the categories are defined.

It is exactly those three issues that make patent classification a real challenge. First, there is the unbalance in class distributions that accounts for a whole range of outliers: classes with either very few or very many train documents. This makes it difficult for the learning algorithm to build class models that are robust enough to recognize test documents that belong to one of the smaller classes. Second, the scalability becomes a big issue in patent classification as the target of the search is a very small subset of the huge feature space. Experiments included in this paper show that to build a compatible patent classifier, one has to work with feature space of 1.5 million dimensions. Third, patents are mostly classified into one main class and multiple secondary classifications. This is called multi classification. Moreover, as patent categories are defined within a taxonomy, they can be classified either hierarchically or level-specific.

A number of authors have reported on machine learning methods to classify patent documents. One of the first important patent categorization systems was developed by [**Larkey, 1999**] which is an automatic search and categorization tool for U.S. patent documents according to the USPTO classification system built on Bayesian classifiers combined with k-nearest-neighbor classifiers [**Pflugfelder, 2009**]. The system includes a unique 'phrase help' facility, helping users to find and add phrases and terms related to those in their query.[**Koster et al., 2003**] used the LCS to compare the Rocchio algorithm with Balanced Winnow on mono and multi classification experiments. They detected potential pitfalls in multi classification and presented ways to improve the accuracy. The classification system used currently at the World International Patent Organisation (WIPO, located in Geneva (CH)) is described in [**Fall et al., 2003**] and deploys a proprietary implementation of the Winnow algorithm. The categorization applications used at the European Patent Office (EPO, located in Rijswijk (NL)) are documented in [**Krier and Zaccà, 2002**]. A recent study [**Cai and Hofmann, 2004**] implemented SVM classifiers to learn class models based on either flat or hierarchical classification across the IPC taxonomy. The WIPO-alpha patent collection was used for training, using a title+claim document representation. The number of training documents, however, was below 15 000.

The most prototypical text categorization algorithm is the Winnow algorithm. Belonging to the family of on-line learning algorithms such as the Perceptron [**Rosenblatt, 1958**], Winnow differs from the latter in the way the algorithm learns the linear separation between examples assigned with different categories. Winnow learns multiplicatively rather than additively and is therefore very

fast. As the text categorization problem is linearly separable, Winnow has been used extensively in this domain. Winnow is however not the most robust learner and experiences problems due to the big irrelevant portion of the feature space. This paper contrasts a specific implementation of the Winnow algorithm called Balanced Winnow with an SVM (Support Vector Machine) learner, known to be a more robust algorithm. The computation of the discriminative function of an SVM happens by solving an optimization problem based on a maximum margin separation. The separating function need not to be linear but can be of a higher degree and, therefore, this function actually forms a hyper plane in the feature space. SVMs are frequently applied in many categorization applications based on texts, images, music, etc. In text categorization, multiple publications showed [**Joachims, 2002, Joachims, 1998**] that SVMs outperforms other usual learning methods in terms of accuracy but with the down side of needing far more calculation time. The test collection of previous SVM categorization experiments was never larger than 10 000 documents. This paper includes results with more than 300 000 test documents.

Extensive comparisons between algorithms have been published in the literature [**Hearst, 1998, Sebastiani, 2002, Yang and Liu, 1999**]. Only controlled experiments on the same corpus and taxonomy can be used to draw comparisons between algorithm accuracies. This study uses a corpus of 1 270 185 patent documents supplied by the company Matrixware[1] and sets up categorization experiments using the International Patent Classification (IPC) taxonomy [**WIPO, 2009a**] on the sub class level (639 classes). Next to comparing **two different learning algorithms**, Balanced Winnow and learning with SVMs, the effect of training different patent representations is investigated. **Four representations** were tested: title+abstract, inventors+title+abstract, title+description and inventors+title+abstract+description. The Linguistic Classification System (LCS) developed at the University of Nijmegen was used as a workbench to carry out the experiments. The LCS represents a full text categorization framework with the possibility to adjust various different components for optimal adaption based on the categorization problem.

This study is the first to draw a comparison between the prototypical text classification algorithm Winnow and the more general machine learning classifier SVM, while using such a high dimensional feature space. Chapter 2 describes properties of patent documents in more detail and focuses on the peculiarities of patent classification in the text categorization domain. A general description of the methodology of the experiments is included in Chapter 3, clarifying the general categorization pipeline and the workings of the LCS. The experiments themselves are documented

[1] http://www.matrixware.com/

in Chapter 4 whereas Chapter 5 includes a use case that searches for an optimization of either the precision or the recall using any of both algorithms. Some extensions of the baseline experiments are included in Chapter 6, where the scalability of the SVM learners is tested and the baseline results are compared to the accuracy a random classifier could achieve. Finally, Chapter 7 summarizes the conclusions of this study.

CHAPTER 2

Text Categorization for Patent Documents

1. Features of Patent Documents

One of the main goals of patent classification systems is to organize and structure patent documents in a comprehensive patent document repository based on the technology described in the documents. Patent classification taxonomies have to be defined to offer a suitable structure of those repositories and usually possess a large number of branches and sub branches to properly represent existing technical fields and their key categories. Taxonomies therefore contain a large number of possible patent classes and the categorization of new patent documents requires detailed background knowledge regarding the described technology as well as how this technology is represented in the corresponding taxonomy. Also the integration of new technology categories into an existing taxonomy affects the assignment of patent documents to related classes, as possible revisions of the underlying taxonomy alter existing relations of patent classes and, thus, the assignments of patent classes to documents.

The content of a patent is governed by legal agreements and is therefore semi-structured. An example European patent application document contains the following fields:

- Bibliographic Data
- Abstract
- Description
- Claims
- Legal Status

The bibliographic data contains useful information such as technical details (e.g. the invention title, citations, ...) and a listing of the parties involved (applications, inventors and agents) but also publication and application references, terms of grant, international convention data and priority claims. The abstract describes in general terms the content of the application whereas the description contains more information on the invention. A more thorough documentation of what has been invented can be found in the description, usually accompanied by multiple tables and figures that support the arguments of the applicant. The claims section states the prior art and

the novelty of the patent application and often contains standard expressions. The legal status of a patent document tells you whether the patent is still an application or whether you are dealing with an already granted patent.

According to [**Mase et al., 2005**] the characteristics of patent documents that affect retrieval accuracy are the following:

(1) Various authors: description style, text length and vocabulary differ strongly
(2) Technical fields are diverse (writing styles differ)
(3) Use of vague expressions in titles and claims (prevent direct term matching in IR)

1.1. The International Patent Classification. Patent documents receive specific codes that refer to the class they belong to. The International Patent Classification (IPC), established by the Strasbourg Agreement 1971 and which entered into force on October 7, 1975, provides for a hierarchical system of language independent symbols for the classification of patents and utility models according to the different areas of technology to which they pertain [**WIPO, 2009b**]. In the past, the IPC has been updated every five years and it is currently in the IPC-2009 edition. The details of all editions of the IPC are included below as they were found in the IPC Guide [**WIPO, 2009a**].

The first edition of the Classification was in force from September 1, 1968 to June 30, 1974,
- the second from July 1, 1974 to December 31, 1979,
- the third from January 1, 1980 to December 31, 1984,
- the fourth from January 1, 1985 to December 31, 1989,
- the fifth from January 1, 1990 to December 31, 1994,
- the sixth from January 1, 1995 to December 31, 1999, and
- the seventh from January 1, 2000 to December 31, 2005.

Following the reform of the IPC [**WIPO, 2009a**], the Classification was divided into core and advanced levels. Each edition of the core level is indicated by the year of entry into force of that edition. IPC-2006 was in force from January 1, 2006, to December 31, 2008, and IPC-2009 entered into force on January 1, 2009. Each new version of the advanced level of the IPC is indicated by the year and the month of the entry into force of that version, for example, IPC-2008.01.

Each IPC code is a unique combination of the hierarchical structure codes of the patent identity. The four levels in the patent hierarchy that are used in this paper are Section (8), Class(121), Subclass($\pm$625) and Main Group ($\pm$6000). There is also a fifth level, called Sub Group, which comprises of $\pm$69000 categories.

TABLE 1. Summary of the IPC statistics [**WIPO, 2009c**]

Section	No. of classes		No. of subclasses		No. of main groups		No. of subgroups		Total no. of groups	
	CL	AL	CL	AL	CL	AL	CL	AL	CL	AL
A	16	16	84	84	1095	1106	1367	7349	2462	8455
B	37	37	165	167	1782	1978	2361	14595	4143	16573
C	21	21	91	93	1304	1326	1124	13121	2428	14447
D	9	9	39	39	349	350	179	2611	528	2961
E	8	8	31	31	317	318	454	2900	771	3218
F	18	18	95	97	1040	1041	858	7294	1898	8335
G	14	14	79	79	683	696	1861	6805	2544	7501
H	6	6	49	49	532	537	1898	7172	2430	7709
Total	129	129	633	639	7102	7352	10102	61847	17204	69199

Table 1 gives an overview of the occupation of the different levels in the IPC taxonomy for the version of 2009. CL refers to Core Level and AL to Advanced Level. According to [**WIPO, 2009b**], "the core level is intended for general information purposes, for example, dissemination of information, and for searching smaller, national patent collections". The advanced level is explained as being intended for searching larger, international patent collections. Any industrial property office can choose to use the advanced level for classifying its published patent documents. The differences between CL and AL only become visible at main group level. The corpus used in this study exists of AL classes only.

As can be seen in Table 1, the IPC sections are represented by one of the capital letters A to H, according to: A: Human necessities; B: Performing operations, transporting; C: Chemistry, metallurgy; D: Textiles, paper; E: Fixed constructions; F: Mechanical engineering, lighting, heating, weapons, blasting; G: Physics; H: Electricity. Each section is subdivided into classes, whose symbols consist of the section symbol followed by a two-digit number, such as C01. In turn, each class is divided into several subclasses, whose symbols consist of the class symbol followed by a capital letter, for example, C01B. IPC subclasses are in turn divided into main groups, and then into a hierarchy of subgroups [**Fall et al., 2003**]. Table 2 below shows a portion of the IPC specification at the start of Section C.

The IPC exists in two authentic versions, French and English, which are published in printed form by WIPO and on-line (http://www.wipo.int/classifications). National industrial property offices also publish the complete IPC taxonomy in other languages: Spanish, German, Hungarian,

TABLE 2. Sample portion of the IPC taxonomy at the start of Section C

Category	Symbol	Title
Section	C	CHEMISTRY; METALLURGY
Class	C01	INORGANIC CHEMISTRY
Subclass	C01B	NON-METALLIC ELEMENTS; COMPOUNDS THEREOF
Main group (and references for this main group)	C01B3/00	Hydrogen; Gaseous mixtures containing hydrogen; Separation of hydrogen from mixtures containing it (separation of gases by physical means B01D); Purification of hydrogen (production of water-gas or synthesis gas from solid carbonaceous material C10J; purifying or modifying the chemical compositions of combustible gases containing carbon monoxide C10K)

Czech, Polish, Russian, Japanese, Korean and Chinese. According to [**Fall et al., 2003**] most updates are made at group and subgroup level.

The advantages of a hand-built document taxonomy are multiple. "It reduces complexity and provides an overview of the knowledge contained within the information set. Hierarchical taxonomies capture information about relationships and are easy to browse. The IPC is however a complex classification system. Because of its broad scope, non-expert human classifiers have difficulty using the IPC for manually attributing IPC codes" [**WIPO, 2009d**]. Automation can be very useful for streamlining categorization and enhancing productivity.

2. The Text Categorization Problem

The text categorization (i.e. document classification) problem is formulated by [**Koster, 2009**] as "assigning class(es) to unseen documents given a set of classes (also called categories), each exemplified by a number of documents". Each document in the collection can thus be classified into multiple, one or no category at all. By means of Machine Learning techniques classifiers are learned from the labeled data in order to perform category assignments automatically. This is supervised learning problem. Each category is seen as a separate binary classification problem that decides whether a document belongs to the category or not, depending on certain thresholds.

In order to fully understand the choice of certain metrics used in text categorization, one has to consider its properties first. [**Joachims, 2002**] includes the following characteristics:

- **High-Dimensional Feature Space**: As every word in a document is seen as a possible feature to train the classifiers, the index of a collection can become extensive due to the richness of natural language. According to Heaps' law (1978) the relation between the size of a document and the number of distinct words can be described by a constant:

$$V = k.s^{\beta}$$

  where k and β depend on the data and s is sufficiently large. Values for β lie typically between 0.4 and 0.6 and for k between 10 and 100. Although one would expect that the distribution would level off as the size of the vocabulary increases, new words never cease to occur as unknown names, spelling mistakes, symbols, numbers etc. are bound to be infinite.
- **Sparse Document Vectors**: Due to the large feature space each document in the collection only contains a small fraction of all the features and most entries will be zero.
- **Heterogeneous Use of Terms**: In text classification we have to take more into account than just the co-occurrence of terms across multiple documents to decide their class membership. Also semantic relations are important to determine whether two documents belong to the same class or not: the use of synonyms and linguistic phrases can add considerable improvements to text categorization. Moreover when document one contains A and B; document two B and C and document three C and D document one and three can still be related although they do not cover any similar terms.
- **High Level of Redundancy**: In order to classify a document there are often more cues in a document then one needs and therefore terms have to be selected to make documents distinctive enough to classify (cf. infra)
- **Frequency Distribution of Words**: Zipf's law describes the fact that the occurrence frequencies of words behave in a very stable way. It states that there is a small number of words that occurs very frequently whereas most words occur infrequently. It has been found that the frequency distribution of individual documents follow Zipf's law approximately. The tail of the distribution sometimes drops off too early.

The above characteristics are also valid for patent classification. Due to the domain-specific vocabulary, a dimensionality reduction is indispensable to make the learning algorithms workable. The limits of efficient learning are for Balanced Winnow situated at 50 - 60 000 terms while SVM

training becomes complicated when more than 20 000 terms are involved in training (p.c. Bernhard Pflugfelder).

There are, however, some additional peculiarities specific to the classification of intellectual property documents [**Krier and Zaccà, 2002**]. First, the documents are relatively large in size. A full text patent has on average 5000 words (30-50 KB). Second, patent documents at the WIPO may be written in two official languages, English and French; at the EPO this rises even to three through the addition of Spanish. Thirdly, patent applications are composed in a well-structured, controlled and correct language. On the other hand, however, there can be intentional use of non-standard terminology, vague terms, and *legalistic* language. There may be stylistic differences between claims and the description of a patent. Fourth, a patent document contains non-linguistic material that could contain important pre-classification information: tables, mathematical and chemical formulas, citations of patents and literature, technical drawings. And fifth, the bibliographic data associated with a patent can be also important for the pre-classification: applicant name, inventors' names, priority and filing dates, etc.

[**Fall et al., 2003**] add two more difficulties to this list: IPC references and placement rules. Many categories in the IPC taxonomy contain references and notes, which are included to guide the classification procedure. Two main types of references are: limitations of scope (point at related categories where some patents should preferable be classified) and guidance references (list related categories where similar patents are classified). According to [**Fall et al., 2003**], Patent classification is governed by placement rules. This means that "in certain parts of the IPC, a last-place rule governs the classification of documents relating to two categories at the same hierarchical level [...] This rule indicates that the second of two categories should always be selected if two are found to concord with the subject of the patent application" [**Fall et al., 2003**].

Another problem already mentioned in the introduction is the fact that we are dealing with multi classification. Since every patent document is usually not only labeled with a single IPC category but rather multiple categories (including one main IPC category), every document can be an example for various class profiles during the training [**Pflugfelder, 2009**].

All the above issues have to be taken into consideration in the design of a patent categorization architecture. Chapter 3 presents this process and introduces the LCS, which is the classification framework used in the experiments.

CHAPTER 3

Classification Methods

This chapter describes the general framework of the study, including an explanation of the Linguistic Classification System (LCS) and the detailed documentation of the categorization pipeline. The latter will be described on a theoretical level, focusing on preferable choices for each step in the architecture. The exact methods used in the experiments are documented in Section 2.2 whereas detailed parameter settings are included in Appendix A, so the experiments can be repeated in further research.

1. Linguistic Classification System

The Linguistic Classification System is developed by the University of Nijmegen in the framework of the DORO and PEKING Esprit Projects. It is a system for the automatic classification of full-text electronic documents. The possibility to adjust various different components for optimal adaption to the categorization problem makes the framework interesting for research activities. The LCS manages a collection of class profiles. The adjective "Linguistic" points at the fact that one could implement phrase selection instead of single word selection in the preprocessing stage. Research into phrase extraction in document preparation is still in an experimental stage [**Koster, 2009**] and experiments with phrases lie not within the scope of this research paper.

Running the LCS consists of three phases described in the manual delivered with its installation [**Koster, 2009**]:

(1) **analysis phase**

After the documents included in the examples list (documents with labels separated by whitespaces) are checked for their presence in the files directory, each document is split into terms by an external analyser component. This preprocessing phase also performs document profiling to collect statistical data from to documents that is stored in the data repository. Finally, term selection is carried out. The corpus is now ready to be trained efficiently and generalization accuracy should be improved.

(2) **training phase**

A classifier is trained for each class and optimal thresholds are computed. The class profiles are then again stored in the data repository where they can be consulted during the testing phase.

(3) **testing phase or selection phase**

In testing the score of "each document for each class" [**Koster, 2009**] is computed (full ranking). Depending on the parameter settings, each test document is assigned to zero or more classes. Performance (use of time and space) and Accuracy (precision—recall—F-value) are reported.

With the delivery of a Java API that makes LCS methods easy to implement, the experimental setup for this study was developed in Eclipse and executed with the use of shell scripts. The LCS offers one single method to execute all steps of the categorization pipeline without any further intervention. As a significant number of parameters can be adjusted, the experimenter gets control over almost all steps of the categorization process. All adjusted parameters have to be stored in a separate resource file which will be assigned to the main execution function of the LCS. The LCS system is fully implemented in C++ where a library contains the entire learning functionality and driver implements I/O operations [**Pflugfelder, 2009**]. One disadvantage of the system is its monolithic architecture. It remains for instance impossible to turn off specific parts of the classification process and include for instance a personal term selector.

2. Classification Architecture

In implementing the LCS a whole range of decisions have to made according to the design of the experiments. The sections below discuss for each of the main issues the methods that have been chosen in the preparation of the experiments.

2.1. Preprocessing. The two most frequent options to represent text that has to serve in a automatic classification task are word and phrase-based representations. It has been shown that sub word level units as well as units that transcend phrases are either not discriminative enough or too discriminative for the goals of text classification [**Joachims, 2002**]. All words/phrases in a document are treated as a Bag of Words (BoW). This means that order and (therefore) relationships are neglected and multiple occurrences of words are added as values to the vector entries the BoW creates. The success stories behind search engines such as Google justify this compromise between expressiveness and model complexity inherent to the BoW technique [**Croft et al., 2009**].

2.2. Feature selection and Term Weighting. As each document will be registered as a vector by the training algorithm, the original document is transformed into selected, weighted and normalized vector entries.

The feature selection combats "overfitting" and can be carried out in two - sometimes complementary - ways: as a feature "subset" selection process or as a feature construction algorithm. In the former case, a subset of original features is selected whereas the latter scenario implies new features that are introduced by combining original features. An effective method to select a subset of the original feature space works by either removing infrequent or highly frequent features or remove irrelevant features. When frequency is an issue stop word elimination and Document Frequency (DF) thresholding [**Yang and Pedersen, 1997**] are two methods that are often used. DF thresholding works by setting an arbitrary threshold for the number of times words can occur in the corpus in terms of document frequency, i.e. how many documents contain the word. Mostly this threshold is kept low (DF 2 or 3). The use of a stop list (i.e. a list of very frequent function words) should always be treated with care as it is language and often also domain specific and one might throw away useful information that is necessary to distinguish between documents.

A good overview of feature selection metrics that are commonly used can be found in [**Forman, 2003**]. The metrics that are of interest to the goals of this study are discussed below. The first two are available for use in the LCS, the third one is not a standard measure but could be implemented in future research and is therefore briefly discussed.

- **Simple Chi-Square**[1]

$$t(tp,(tp+fp)P_{pos})+t(fn,(fn+tn)P_{pos})+t(fp,(tp+fp)P_{neg})+t(tn,(fn+tn)P_{neg})$$

where $t(count, expect) = (count - expect)^2/expect$

This statistic estimates the independence of two variables by measuring the divergence of the expected word occurrence and the actual scenario if independent feature occurrence is assumed. A term will be well discriminative if it is highly dependent on one particular class. For every class, term selection is done on a binary basis. The optimal number of selected terms is obtained by a likelihood maximization using the defined learning algorithm (cf.

[1] tp = true positives; fp = false positives; tn = true negatives; fn = false negatives

$$pos = tp + fn; neg = tn + fp; P_{pos} = pos/all; P_{neg} = neg/all$$

infra: Winnow or SVM) as a verification (p.c. Bernhard Pflugfelder). [**Forman, 2003**] points at the fact that it is a statistical test and does not work well for very small expected counts, which is often the case for word occurrence features.

- **Information Gain**

$$e(pos, neg) - [P_{word} * e(tp, fp) + P_{\cancel{word}} * e(fn, tn)]$$

As the name suggests, this metric measures the amount of information particular bits contribute to a document. More concretely it measures the decrease in entropy when a feature is given vs. absent [**Yang and Pedersen, 1997**]. If the entropy decreases after taking out a feature, this indicates the feature is discriminative for the document.

- **Bi-Normal Separation** [**Forman, 2003**]

$$F^{-}1(tpr) - F^{-}1(fpr)$$

($F^{-}1$ is the z-score)

The BNS measures the distance between the thresholds of the positive and the negative classes. If the feature is more prevalent in the positive class, then its threshold is further from the tail of the distribution than that of the negative class. [**Forman, 2003**] pointed at the superiority of BNS over other feature selection methods for SVM learning.

Apart from feature selection, feature weighting is also an important part of constructing a document vector ready to be used by a classifier. In this phase, every remaining feature is given a weight that indicates its importance for the document. In the LCS four different "strength" options are provided:

- **Bool**: 1 if the document contains the term, 0 otherwise
- **Freq**: Term frequency of a term in a document
- **Sqrt**: The square root of Freq
- **LTC**: (1+ ln Freq) * ln $\frac{|C|}{DF}$ where $|C|$ is the size of the collection and DF the document frequency of the term in question in the training set

Important here is the normalization by the length of the document as longer documents are biased towards higher term frequencies than shorter documents. An important formula in this field is the TF.IDF term weighting metric, in the LCS translated into the LTC metric.

2.3. Learning task. When the final document vectors are stored in the desired format, the corpus is split into a training corpus and a test corpus. Divisions of 80%-20% are encountered frequently and therefore used in this study. The training corpus is used to learn the labels from the document vectors so we can afterwards predict the labels of the test corpus and measure the divergence between the predicted and the actual labels. In the past, a whole range of Machine Learning algorithms have been used in text classification tasks. [**Sebastiani, 2002**] gives an excellent overview of the machine learning approach to text categorization and the inductive construction of text classifiers.

Currently the LCS implements two classification algorithms: Rocchio's algorithm and Balanced Winnow. Any external algorithms can be coupled to the LCS in the training phase. An example of such an algorithm that is used in the experiments documented in this paper is the SVM_{Light} developed by Thorsten Joachims (http://svmlight.joachims.org/). The performance of the SVM algorithm is compared with the results of LCS training with Winnow. A short description of each of the algorithms is included below in order to support the interpretation of the results included in Chapter 4.

2.3.1. *Winnow.* On-line methods are widely used in text categorization. Based on linear classifiers where on-line methods are used as an alternative to batch methods (e.g. LDA)[2] they compute the categorization value incrementally. This means a classifier is built soon after seeing the very first document and the algorithm works by performing multiple iterations over all training documents. In linear classifiers the categorization value consists of calculating the similarity between the document vector and the class vector for instance by computing the dot product or the cosine similarity. Important to note is the fact that these algorithms are heuristical, i.e. calling them twice on the same data set does not lead to exactly the same result.

There are two common implementations of on-line algorithms: weights are either updated additively or multiplicatively. The weights represent the importance of each term for each class. Score functions of linear classifiers have the form:

$$SC_j(d_j) = \sum_{t_k \in d_i} s_{ik} * w_{jk}$$

This function indicates how well a document belongs to a class C_j by taking the dot product of the strengths of the terms in the document and the weight the terms have in the collection.

[2]In batch methods a classifier is built by analyzing the training set all at once. More specifically in LDA learning the stochastic dependence between terms relies on the covariance matrices of various categories.

The algorithm that learns multiplicatively is called Perceptron [**Rosenblatt, 1958**]. It is initialized by setting all the weights to the same positive value. When a training instance comes in, it is examined and the classifier built so far tries to classify the incoming vector. The result of this process is examined and when the prediction is wrong the weights are modified additively. Because of this feedback on-line methods are also called mistake-driven learning methods.

Winnow is a multiplicative weight updating algorithm that just as the Perceptron tries to find the best linear separator between relevant and irrelevant documents for a class. The implementation of the algorithm reported in this paper is Balanced Winnow [**Littlestone, 1988, Grove et al., 2001**]. The classifier consists in this case of weight pairs (positive and negative weights) that are used to calculate the class membership score of a document. The positive weights indicate evidence for class membership whereas negative weights provide negative evidence. The overall weight of a feature is thus the difference between the positive and negative weights:

$$w_{jk} = w_{jk}^{+} - w_{jk}^{-}$$

where both parts are initialized as follows:

$$w_{jk}^{+} = \frac{2}{d} \qquad w_{jk}^{-} = \frac{1}{d} \qquad with \qquad d = \frac{ST_k}{|T_j|}$$

This means that for an average document belong to class C_j, the score function SC_j would yield 1. ST_k is the sum of all term strengths occurring in a class. T_j is the collection of all terms belong to the class.

Again weights are only updated when a mistake is made. If a mistake is made on a positive example, the positive part of the weight is promoted ($w_{i_j}^{+} \leftarrow \alpha * w_{i_j}^{+}$) while the negative part of the weight is demoted ($w_{i_j}^{-} \leftarrow \beta * w_{i_j}^{+}$). When a mistake occurs on a negative example the positive part of the weight is demoted ($*\beta$) and the negative part is promoted ($*\alpha$). Apart from promotion and demotion parameters α and β on-line algorithms also have a threshold θ that forms the decision criterion for class membership. In Balanced Winnow the *thick threshold* heuristic is applied. This means that in training, rather than forcing the score of relevant documents above 1 and irrelevant documents below 1(θ), we have two thresholds: $\theta^{+} > 1.0$ and $\theta^{-} < 1.0$. The result is judged incorrect either if the score of a document is below θ^{+} although it belongs to the class or if the document does not belong to the class although its score is above θ^{-}.

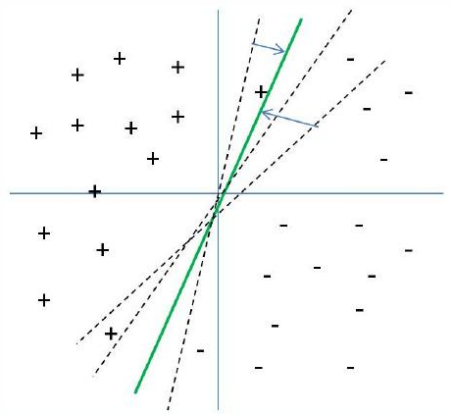


FIGURE 1. The Winnow Algorithm at work

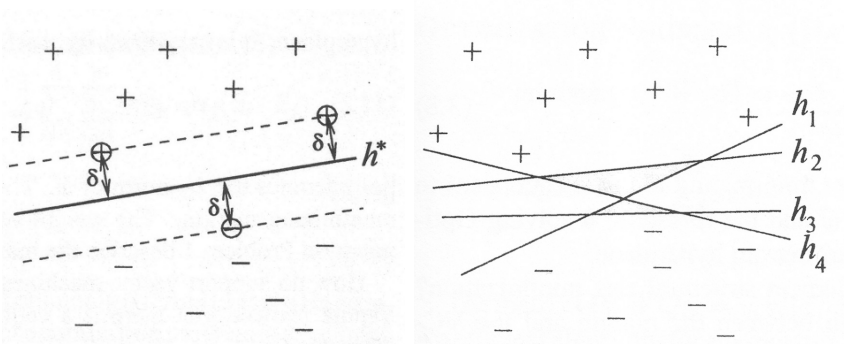
There are versions of Balanced Winnow where only positive or negative weights are used in learning. However, as [**Dagan et al., 1997**] point at, such versions have "the need to update the all the weights" rather than only the active features. When working with a sparse high dimensional space updating only the active features has a big computational advantage.

2.3.2. *SVM.* In the text categorization process the training data can be separated by at least one hyperplane h'. This presupposed a weight vector $\mathbf{w}^T$ and a threshold b^T, so that all the positive examples are on one side, while the negative examples can be located on the other. This is equivalent to requiring $t_i(\mathbf{w}^T * \mathbf{x_n} + b^T) > 0$ for each training example (x_n,t_n). In practice, there can often be several hyperplanes that separate the data but as Support Vector Machines (SVMs) are based on the Structural Risk Minimization principle[3] [**Vapnik, 1999**] only the hyperplane that maximizes the margin δ separating positive and negative examples is selected. This is because the principle searches for the lowest true error on a hypothesis h. This means that by controlling the VC-dimension of H, SVMs can minimize the upper bound that connects the true error with the error on training set and complexity of H. The small set of training examples that determines the best surface are called the *support vectors*. They have a distance of exactly δ. Figure 2 illustrates the binary classification process with SVMs. The support vectors are marked with circles.

Finding the hyperplane with maximized margin comes down to minimizing $\mathbf{w}$ and b:

$$\min V(\mathbf{w}, b) = \frac{1}{2}\mathbf{w} * \mathbf{w}^T$$

[3]More information on how SVMs implement structural risk minimization can be found in [**Joachims, 2002**] on page 38.



(a) Many hyperplanes separate the training examples without error

(b) Support Vector Machines find the maximum margin δ

FIGURE 2. Binary Classification [**Joachims, 2002**]

$$\text{subject to: } \forall_{i=1}^{n} : t_i(\mathbf{w} * \phi(x_i) + b) \geq 1$$

The constraints make sure that all training examples lie on the correct side of the hyperplane [**Joachims, 2002**]. If the equation equals to one the constraints are said to be active, which means the data point in question is closest to the surface. At least two of those are found when the margin is maximized. To solve the optimization problem Lagrange Multipliers are introduced (cf. [**Bishop, 2006**]). The binary training vectors x_i are mapped into a higher dimensional space by the function ϕ. SVMs are kernel methods and can operate with different kernel functions. The simplest kernel function that can be used is a linear kernel where an identity mapping of the kernel function takes place [**Bishop, 2006**]:

$$\phi(x) = x \Rightarrow k(x, x') = x^T x'$$

A linear kernel seems most appropriate for text categorization problems as we are working with a high-dimensional input space and data that is almost always linearly separable. Moreover, there are only few irrelevant features and document vectors are sparse. Other kernels that are used frequently are radial-based functions or polynomial kernels.

One problem with the implementation of SVMs as described above is that training fails when the training examples are not linearly separable. Even though this is almost never the case in text categorization, flawless training can result in over generalization of the data and therefore affect the testing accuracy. This approach is called *soft-margin* SVM [**Cortes and Vapnik, 1995**]. When

training with soft margins, an upper bound on training errors is included in the optimization function where this bound is minimized simultaneously with the length of the weight vector.

$$\min V(\mathbf{w}, b, \xi) = \frac{1}{2}\mathbf{w}\mathbf{w}^T + C\,\Sigma_{i=1}^{n}\xi_i$$

$$\text{subject to.} \qquad \forall_{i=1}^{n} : t_i(\mathbf{w}^T * \phi(x_i) + b) \geq 1 - \xi_i$$

$$\forall_{i=1}^{n} : \xi_i > 0$$

The ξ_i are called slack variables. They will be greater than 1 if the training example lies on the "wrong" side of the hyperplane. Therefore $\Sigma_{i=1}^{n}\ \xi_i$ is an upper bound on the number of training errors [**Joachims, 2002**]. The parameter C allows trade-off between training error vs. model complexity. Small values of C increase the number of training errors, large values tend to behave like hard-margin SVMs. The best value of C depends on the data and must be found empirically.

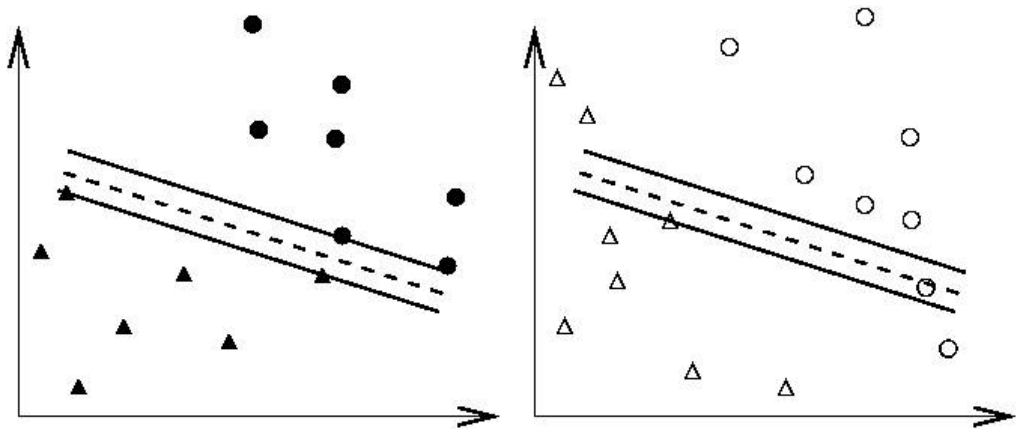
In text categorization, the number of negative examples is always much larger than the number of positive examples. This means that when the classifier predicts everything to be negative it still achieves a high accuracy. This is why in practice, errors on positive examples should be penalized stronger than errors on negative examples. The cost of false positives (C_{-+}) vs. false negatives (C_{+-}) can be incorporated into the SVM by optimizing the following function [**Morik et al., 1999**]:

$$\min V(\mathbf{w}, b, \xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C_{-+}\Sigma_{i:t_i=1}\xi_i + C_{+-}\Sigma_{j:t_j=-1}\xi_j$$

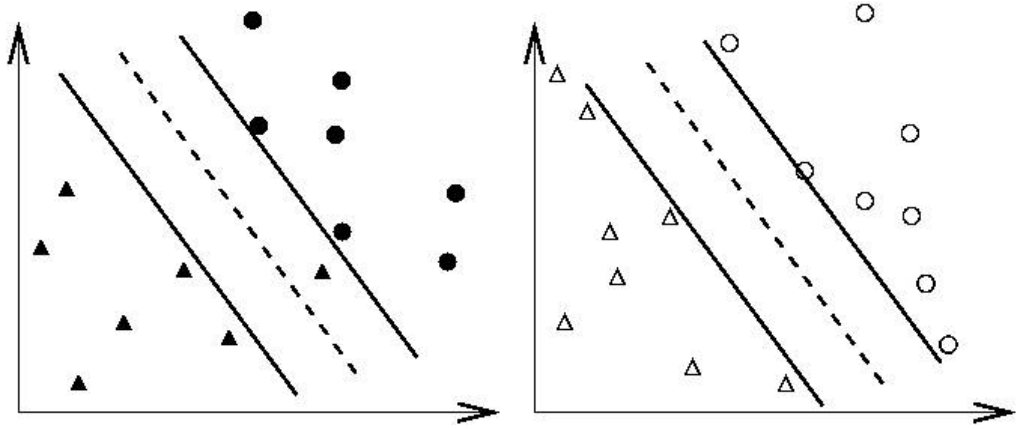
$$\text{subject to} \qquad \forall^k : t_k[\mathbf{w}^T * \phi(x_k) + b] \geq 1 - \xi_k$$

SVMs are universal learners. Independent of the dimensionality of the feature space SVMs can find all the decision surfaces that separate the positive from the negative training examples. The complexity is measured only on the margin with which the SVM separates the data and thus not on the complete feature space. This approach prevents overfitting.

Implementations of the SVM algorithm apply either hard or soft margins in the separation of the training documents. A hard margin distinguishes between positive and negative documents in a strict way whereas the use of a soft margin allows some errors in the classification depending on the value of a slack variable ξ. The soft margin makes the trained model more flexible for variations between the test and train corpora.



(a) Training data and an overfitting classier (b) Applying an overfitting classier on testing data



(c) Training data and a better classier (d) Applying a better classier on testing data

FIGURE 3. An overfitting classier and a better classier [**wei Hsu et al., 2003**]

2.3.3. *Winnow vs. SVM.* The following table summarizes the main characteristics of both algorithms so the results of the experiments can be interpreted with the different nature of the learning algorithms in mind.

Both algorithms work with binary classifiers. This means that we have as many trained models as we have classes and each model is trained on a one-against-the-rest basis. Because of this limitation, the number of negative examples will be very high in each classifier.

Winnow	SVM
Fix threshold;	Directly optimize the Perceptron mistake bound [4]
go iteratively through data to update **w**	
Multiplicative on-line learning	Convert mistake bounds to generalization bounds
Mistake-driven	Batch algorithm
Many irrelevant features [**Zhang, 2000**]	Few irrelevant features
Supervised learning	Supervised learning [**Joachims, 1998**]
	Generalizations depend on feature space
Generative training	Discriminative training

2.4. Measures of quality. Each document in the test set is relevant to one or more classes. During testing the classification system assigns zero or more classes to each test document. Taking into account the class thresholds, the document is then selected for any class for which it crosses the threshold. The documents tested for each class fall into four categories:

	relevant	non-relevant
selected	True Positive (TP)	False Positive (FP)
not selected	False Negative (FN)	True Negative (TN)

The quality of the classification is often calculated by the following measures:

- The **precision** is the fraction of relevant documents out of the selected set: $P = \frac{TP}{TP+FP}$
- The **recall** measures how many relevant documents were found overall: $R = \frac{TP}{TP+FN}$
- **F measure** is a combination of P and R, depending on the β - parameter: $F_1 = \frac{2}{1/R+1/P}$

To calculate the quality of the complete classification process two averages over all classes can be calculated:

- The **macro average** is the harmonic mean of a specific measure over all classes
- The **micro average** (**lumped average** [**Koster, 2009**]) is obtained by summing the quantities TP etc. over all classes and then computing the above measures over this aggregated class:

$$P_{micro} = \frac{\Sigma_{classes} RS}{\Sigma_{classes} RS + NRS}$$

CHAPTER 4

Experiments

The following experiments explore what the effects are of training several document representations and different learners on a corpus of patent documents. The evaluation is done with the LCS, as introduced in Chapter 3.

1. Description of the corpus

1.1. General Statistics. The complete corpus contains 1 270 185 patent documents that are split up into two sub collections: EP (563 248) and WO (706 937).

TABLE 1. Statistics after quality check

	EP collection	WO collection	Overall
Available documents	563713	730279	1293992
Different doc numbers	563271	711701	1274972
Valid patent documents	563248	706937	1272868
Invalid patent documents	465	20659	21122
no title	0	4	4
no abstract	342	19296	16953
no description	4	0	4
no inventor name(s)	98	1973	2071
no applicant name(s)	0	5	5
otherwize	21	2064	2085
Number of classes	121 (93.8% of CL)	121 (93.8% of CL)	121 (93.8% of CL)
Number of sub classes	618 (97.2% of CL)	621 (98.1% of CL)	624 (98.6% of CL)
Number of main groups	5846 (82.7% of CL)	6028 (84.9% of CL)	6261 (88.17% of CL)

The patents were provided in XML format under a patent-specific DTD called Alexandria which was created by Matrixware (http://www.matrixware.com). The collection contains all EP and WO patent documents in the period 01/01/1985 - 31/12/2006. The experiments were set up to explore

the effect of selecting different parts of a patent document as the input of the LCS. Therefore, segmentation processes were carried out using the Java API XOM™, an XML object model that makes the extraction of tag- and attribute-specific text more flexible. After XML stripping is done, the LCS takes care of tokenization (e.g. separating punctuation marks from words), parsing of sentences from documents and the removal of spurious characters.

Documents can have multiple categories assigned to them. The Alexandria collection is built so that the first category in the list is the main category the patent belongs to. On average a patent document has 1.920 categories. There is one document with 20 assigned categories: WO/001997/02/93/19/WO-1997029319-A2 (A23G A23L A23P A24D A45D A61K A61Q B05B B05C B65D C07K C12C C12G C12N C12Q D06Q F21K F21S F21V F41B). The main category for this document is A23G: COCOA; CHOCOLATE; CONFECTIONERY; ICE-CREAM. As the patent document is a WIPO patent and carries the kind code A2 we know that this is a patent application published without a search report. This means that the patent office was paid to carry out a prior art search. Codes of existing patents that are similar to the patent application have to be included.

1.2. Train and test sets. Train and test sets were separately assembled in terms of IPC sub class and IPC main group level. Since a couple of sub classes and main groups are represented by a very low number of examples (or even a single example), only those sub classes and main groups were considered in the train/test set generation which were represented by 5 examples at minimum. Consequently, every sub class and main group is linked to 4 patent documents in terms of the training set and one patent document in the test set respectively. The tables below present statistics for the sub class train and test sets.

TABLE 2. Train set

IPC Sub Class Level	EP-based Collection	WO-based Collection	Combined Collection
Number of patents	450682	565491	1016173
diff. sub classes	618	621	624
min examples per sub class	4	4	4
max examples per sub class	26858	75032	97564
avg. examples per sub class	1402.25	1745.53	3126.09

1.3. Variations in corpus size. Additionally, four smaller subsets of the full corpus are built including 100 000, 200 000, 400 000 and 500 000 patent documents respectively. In order to select

TABLE 3. Test set

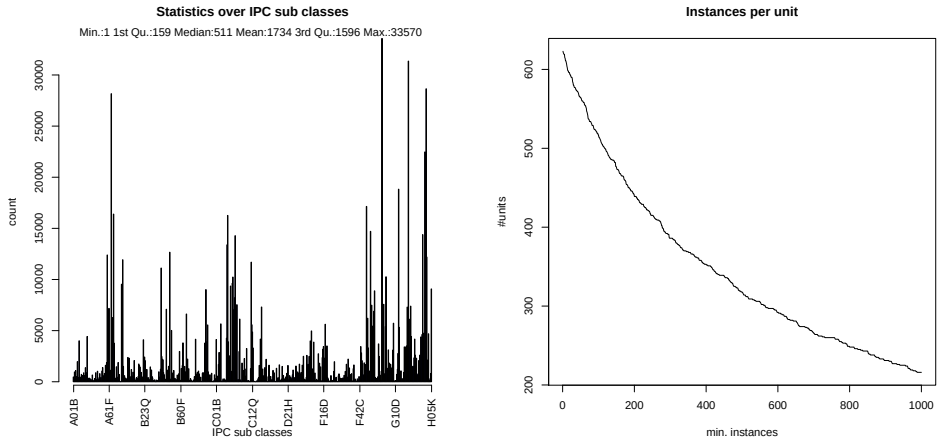
IPC Sub Class Level	EP-based Collection	WO-based Collection	Combined Collection
Number of patents	112563	141442	254005
diff. sub classes	618	621	624
min examples per sub class	2	1	1
max examples per sub class	6715	18758	24391
avg.examples per sub class	351.08	436.86	782.51

appropriate subsets an application was written that sorts the classes in increasing size and picks N classes out of each slice of the M-folded list. The application is described in the following steps:

- Collect all categories from the large train/test collection and the number of examples for each category.
- Group the categories according to the number examples representing each category
- For every group a minimum number of categories must be selected and a overall number of patent documents must be reached.

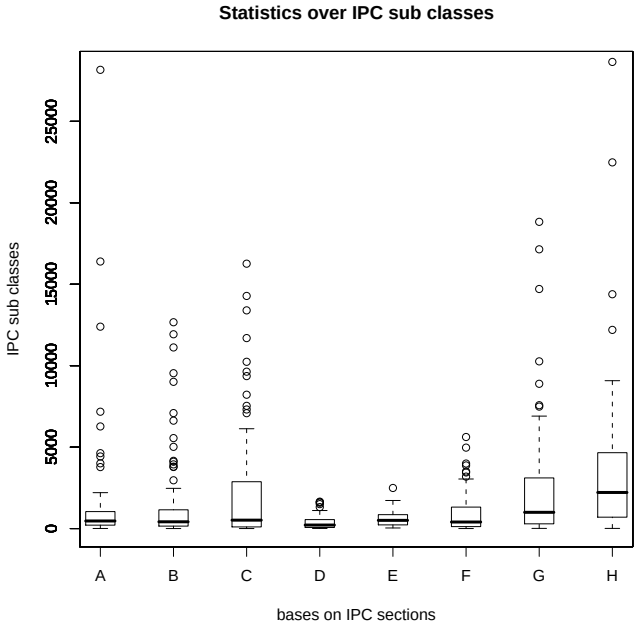
Eventually, this application maintains the unbalance in terms of available examples for every categories which can also be observed in the large collection, even though the number of included categories decreases. This issue must be considered as it significantly effects the quality of categorization independently from the choice of learning algorithm. The natural unbalance of the corpus is illustrated in Figure 1.

The sub figures show the distribution of examples over the different categories on sub class level. In (a) basic statistics are depicted such as the minimum and the maximum number of examples in one class (1 vs. 33570), the median (511) and the mean (1734). The fact that the median is smaller than the mean indicates that the distribution has a heavy tail, i.e. there are large outliers that pull the mean up. This is also illustrated by Figure (b) where the x-axis displays the number of examples per class and the y-axis the number of classes. The curve decreases rapidly in an almost linear fashion at the beginning and has a long tail towards the end with the classes that have more than 1000 examples (not visible in (b)). The box plot in (c) shows that D is the section that contains the categories that are smallest in size. In Section D patents in the field of TEXTILES and PAPER are collected. The section with the fewest sub classes is E (30) (FIXED CONSTRUCTIONS). Section H, containing patents on inventions in ELECTRICITY, is the third smallest section if the number of sub classes (48) is taken into account but has the widest box plot, indicating that the unbalance in the size of the categories is maximized in this section. The section with most subclasses (166, i.e.



(a) Distribution of examples

(b) Feature - class membership



(c) Class distributions

FIGURE 1. Sub class statistics

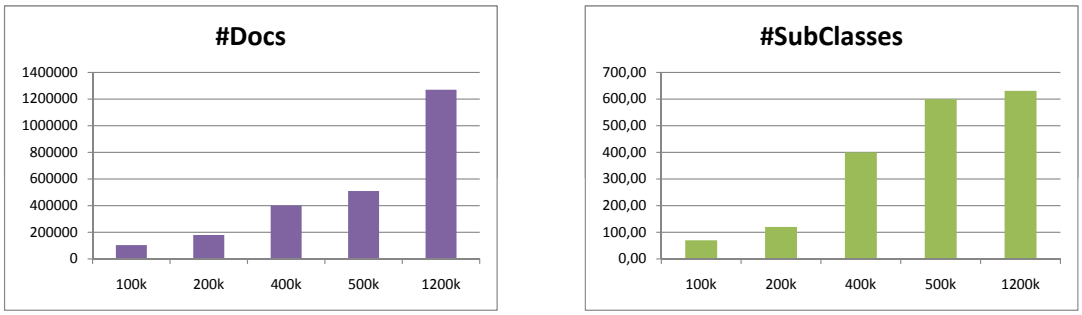
26%) is B: PERFORMING OPERATIONS; TRANSPORTING. The two biggest outliers (outside the scope of (c)) can be found in section A (HUMAN NECESSITIES): A61K with 94004 examples (PREPARATIONS FOR MEDICAL, DENTAL, OR TOILET PURPOSES) and A61P with 49363 examples (THERAPEUTIC ACTIVITY OF CHEMICAL COMPOUNDS OR MEDICINAL PREPARATIONS). The top ten biggest sub classes are included in Appendix B.

The following sample collection is used in the experiments. The percentages between brackets indicate the amount of all the IPC categories are trained in each subset. In the complete corpus, 94% of all classes is used and 98% of all subclasses. One can conclude that the corpus is highly representative for the IPC taxonomy and can be relied on to build a classification system.

TABLE 4. Training sets

Dataset	Num. of docs	Num. classes	Num. subclasses
100k	103666	52(40%)	70(11%)
200k	179250	70(54%)	120(19%)
400k	400750	109(84%)	400(63%)
500k	509560	120(93%)	600(94%)
1200k	1270185	121(94%)	631(98%)

Figure 2 depicts the five training corpora and their distribution of documents and features for the tabs document representation. There is a visible jump between 200k and 400k: although the number of documents is doubled, the number of classes added to the training is more than tripled (120 → 400). On the other hand, the proportion of added documents between 500k and 1200k is much bigger than the corresponding difference in class numbers (600 → 631).



(a) Docs

(b) Subclasses

FIGURE 2. Distribution of documents and subclasses across the different corpora

1.4. Document representation. The four different kinds of document representation that were selected for the experiments are summarised in Table 5.

TABLE 5. Document representations

Doc representation	**Name**
tabs	Title + Abstract
`tdesc`	Title + Description
`itabs`	Inventor(s) + Abstract
`itabs`desc	Inventor(s) + Abstract + Description

2. Test Environment

2.1. Test Server. All the experiments were run on the LDC(Large Data Collider), a server provided by the IRF. The parallel computing abilities used in the LCS were specifically adapted for use on the LDC.

This server runs under SUSE Linux Enterprise Server 10 and has the following properties:

- 40 * Dual Core Itanium 2 (1.4 GHz)
- 320 GB Memory

2.2. Parameter Settings. For each experiment, a number of parameters have to be set that define the way the categorization is carried out. As described in Section 2, a categorization process exists of multiple phases that have to be tried out chronologically. As the LCS does not have phrase extraction installed in its standard version yet, there was no choice as to how the textual data was analyzed and the standard BoW method was therefore used. Any semantic interpretation whatsoever is thus excluded from the experiments. The standard parameter settings used in the baseline experiments are briefly explained below. Please note that the default values indicated below do not correspond to the default LCS parameter settings. The parameters used here were selected conform with/to the objectives of the baseline experiments.

The settings are presented according to the categorization pipeline:

- The patent documents are stored in the `FileDirectory` after they were transformed from the Alexandria XML format into ASCII text documents in an application executed outside the LCS. Although ASCII encoding does not always suffice for the encoding of patent documents (as they often contain special characters), the LCS does not yet support other encodings. Each patent is stored in a single file onto the file system [1].

[1] As this can cause I/O problems, it could be considered to generate one Master Label File (cf. Speech Recognition) that contains the complete patent corpus according to the document representation.

- The only preprocessing steps done by default in the LCS are de-capitalization and removal of special characters like braces. Numbers are not replaced by a NUM token. As they are mostly not significant to the class profile a document belongs to, they will not be selected.
- A global, class-independent term selection is applied by selection only terms which occur in more than three documents. An additionally noise reduction is also done based on the Uncertainty heuristic. This heuristic defines the implementation of an additional noise selection that is based on the uncertainty of the class average of the relative term frequency [**Koster, 2009**], modeled by a Gaussian distribution. A term is defined to be *noisy* within a class if its quality factor is below a certain threshold. A *stiff* term is a term that occurs too regularly in the train set (e.g. "the" in English documents) and has a quality factor that is higher than the upper threshold.
- The local term selection (class-specific) is done by calculating the Simple Chi Square heuristic to estimate the independence of a specific term among each of the classes (cf. 2.2).
- Term strengths (i.e. term weights) are computed by the so-called LTC algorithm, a TF.IDF estimator that is used inside the LCS.
- A term profile is created by a proprietary indexer that extracts the terms. Class profiles containing the term distributions (per class) are created. These are important in the term selection step (p.c. Bernhard Pflugfelder).
- Training is done using 10 CPUs of the LDC based on a train/test split of 80/20. The class distribution in train and test set is kept proportionally similar in order to provide realistic performance measurements. The train-test split algorithm was favoured against a cross validation algorithm because of the sufficiently large number of patent documents available for both training and testing guarantee robust categorization results.
- For both text categorization algorithms, Balanced Winnow and Support Vector Machines, the learning parameters were chosen based on comparable experiments published in the literature or on common default parameters. Chapter 5 explores the influence of learning parameters and reports parameter tuning experiments that were carried out to optimize the baseline performance.
- The Winnow experiments are conducted with the following parameter settings:

TABLE 6. Winnow parameter settings

Promotion(α)	Demotion(β)	Iterations	Θ^+	Θ^-
1.1	0.9	5	1.1	0.9

These settings are Winnow-specific and chosen based on an evaluation carried out within the domain of patent categorization [**Koster et al., 2003**]. The fact that five training iterations are used means that each example was trained five times.

- Experiments with SVM_{Light} had the following default learning parameters:

TABLE 7. SVM parameter settings

Soft margin (C)	Kernel function	Cost (J)
1.5	linear	1.0

A basic evaluation of C using a small subset (±50000 docs) of the corpus revealed $C = 1.5$ as a good choice in terms of F measure. Figure 3 shows the results of this small parameter tuning. The F measure reaches its maximum at 1.5. In this graph, the recall is still increasing whereas the precision decreases slowly. In any case, this fast parameter tuning does not necessarily yield the best results as the data set is so much smaller than the real corpus.

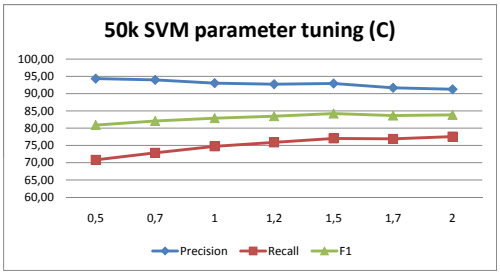


FIGURE 3. Tuning C on a small scale subset

SVM is known to be efficient in working with many irrelevant features. As opposed to Winnow, SVM_{Light} uses its own term selection on top of the LCS term selection.

- In order to discover the underlying multi-labeling categorization (cf. IPC taxonomy), two fundamental meta-learning approaches were applied. The first series of experiments use

a strategy where an ensemble of a binary classifier is selected where every classifier is trained on a single class against all the others. The final predictions are then determined by a specific aggregation of all partial predictions (p.c. Bernhard Pflugfelder). The second meta-learning approach implements an hierarchical classifier into the experiments. Every inner node of the IPC taxonomy is defined by the classifier as it decides on every level the class the document most probably belongs to, without the possibility of a back propagation of error.

- The last step in the classification process is testing. The two most important parameters to set here indicate the lower and upper classification bounds of each document. The maximum number of classifications (default 4) can be chosen to produce a mono or multi classification depending whether it gets a value that is greater than 1. As it has to be possible to use this system for commercial goals, the classifier has to suggest at least one class for a given test document.
- The results are stored on the file system including the plain list of predictions for every test document as well as all model parameters that can be used to reproduce test results.

3. Results

The parameter settings specified above were used for all the experiments reported in this section. The results included in this section are the consequences of changing the input data directory in the parameter file (size of corpus + document representation), the learning algorithm (Balanced Winnow or SVM_{Light}), the training method (Flat or Hierarchical) and the parts file (Multi or Mono). The latter can either be a file that contains multiple classes for each document or just one single main class and therefore make the training a mono-classification process, whereas testing remains multi-classification. Performance is measured according to the MICRO precision, recall and F1 of the results. The baseline experiments are carried out with the following settings:

Algorithm	**Corpus Size**	**Doc Representation**	**Train Method**	**Catalog**
Winnow	100k	Title + Abstract	Flat	Multi

3.1. Corpus Size. As the full corpus consists of 1293993 documents shared by the collections EP and WO, an application was written to create sub samples that could be compared against each other. In order to select appropriate subgroups of the complete corpus the application sorted the classes in increasing size and picked N classes out of each slice of the M-folded list. Experiments

showed that the fewer documents (and therefore categories) were used, the better the performance. With fewer classes to train, fewer mistakes can be made and consistency in classification tends to be higher. When using 100k, the feature space is more than ten times smaller than when training with the full corpus (±1 200 000 docs) and only 70 instead of 631 classes are trained.

Figure 4 shows that there is a similarity between 100k and 200k, whereas there is a sharp downfall to 400k followed by a relatively flat tail. The change from 500k to 1200k documents is compared to the difference between 200k and 400k relatively stable. This is due to the small difference in classes that were used to train both corpora (600 vs. 631). The big challenge lies thus in dealing with a high number of classes rather than with an elevated number of documents.

Depending on the size of the corpus, either the precision or the recall takes the overhand. Whereas the recall is highest when the number of classes stays small (77.87% recall vs. 72.75% precision at 100k (70 classes)), in the steepest part of the curve, when adding 280 classes to the 120 we had, the recall decreased much more than the precision and the latter became the best scoring measure (59.20% vs. 57.77% at 400k). In the tail of the curve, the recall keeps on decreasing more rapidly than the precision. When the full corpus is trained, the precision strands on 56.43% (−2.77% from 400k) while the recall reaches only 52.81% (−4.96% from 400k).

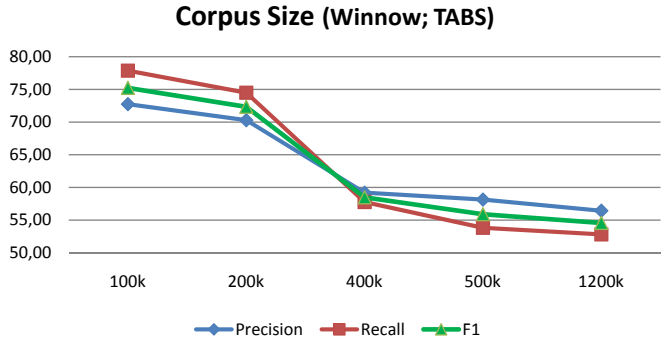
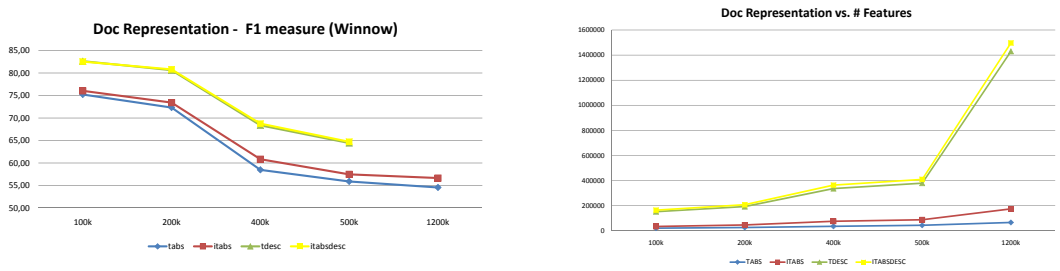


FIGURE 4. Effect of the corpus size on classification performance

3.2. Document representation. Four document representations were given as an input to the classifier: tabs, `itabs`, `tdesc` and `itabs`desc. Statistics of the collections can be found in Section 4. Figure 5 summarizes the results of the experiments with a baseline training. It becomes

clear that training with larger data representations, i.e. more features to define a document and to build thresholds for class assignment, leads to a better performance. The relative difference between training with the least extracted fields (`tabs`) and the most extracted (`itabsdesc`) lies at 7.31% in terms of F1 measure. The backside of this level of performance is the time cost of training and testing, which increases with the number of added features.

What is more remarkable is the effect of adding the inventor names to the training. Although this can in the most extreme scenario only add two extra words to the document representation (name + last name of the inventor), it is these words that are discriminative in training and classifying unseen documents. The difference in performance between `tabs` and `itabs` is minor but visible. Their means (F1 over all document sizes) are situated respectively at 63.3 and 64.9. For the 100k experiments, the difference in feature number between `tabs` and `itabs` is 7.6 words on average. In percentage of F1 value is the difference less than 1% (0.81%). The biggest difference is situated in the 400k collection where it amounts to 2.36%.



(a) Doc Representation (b) Features

FIGURE 5. Document representation (baseline)

When only using title and abstract, the addition of inventor names seems to have a much bigger impact than when the description is included in the data representation. The difference between `tdesc` and `itabsdesc`, the latter even containing two additional data fields, is only minor. This result points at two different things. First, the role of the features found in the patent's description is important as both representations that contain the description score considerably higher than the ones without this field.

Second, the slight difference in F1 value could point to the fact that perhaps the abstract itself does not contain such discriminative information as one would assume. On the 100k corpus,

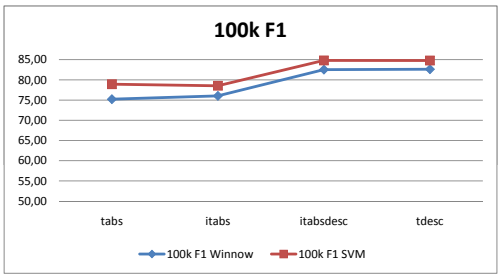
the difference in feature number between `tdesc` and `itabsdesc` is situated at 8.7 words (vs. 7.6 between `tabs` and `itabs`). Although it is generally expected of abstracts to be precise and contain words that are carefully chosen, this does not hold for most patent abstracts. On the contrary, lawyers use inclusive and general language to summarize the patent application. This argument can also be found in the literature [**Koster et al., 2003**]. On the other hand, one could argue that once the description is part of the document representation, the abstract does not really add an extra value to the document as words that occur in the abstract will probably also appear in the description. Figure 5(b) shows that the difference in features that are selected for the training is almost negligible between `tdesc` and `itabsdesc`. The biggest difference in feature number occurs in the full corpus training. `itabsdesc` has significantly more features than `tdesc` at 1200k compared to 400k. Also the difference between `tabs` and `itabs` is significant and even proportionally larger than `itabsdesc`/`tdesc`. Whereas the `tabs` feature number fluctuates between 20767 and 66508 across the different corpora, the `itabs` documents collect more features when the size of the corpus increases: 34466 at 100k and 174923 at 1200k.

Statistical significance tests are included further in this chapter (Section 4).

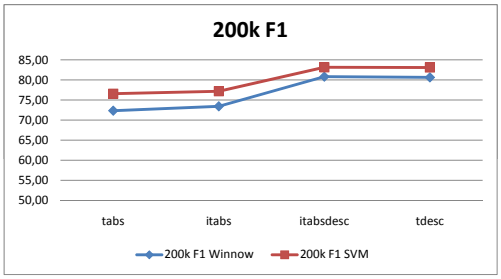
3.3. Learning algorithm.

3.3.1. *Subsamples.* The graphs in Figure 6 show that training with SVM (in red) is always superior to Winnow training (in blue) in terms of F1 measure. Results for all the different collections are included [2] as the differences between the two learners are not exactly the same for each subset. When the 100k collection is considered, F1 measures are on average (over all document representations) 2.66% higher for SVM trained categorisers, whereas this number rises up to 3.71% for the 400k collection. Figure 7(a) illustrates this difference. If one compares the difference between 100k and 400k in terms of classes (70 vs. 400), one could argue that this increase influences the way training reacts in both algorithms. When another 200 classes are added (500k), the difference in F1 score decreases again slightly. It seems thus that Winnow reacts stronger on the increase of classes. This suggests that Winnow tuning should happen on the basis of the class number rather than the document representation (and therefore feature number) as was done in [**Koster et al., 2003**].

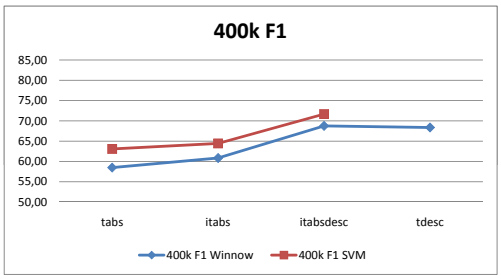
[2]The results of the experiment **SVM 1200k itabsdesc** were not available at the time of handing in this document, due to the extremely long training time of SVM with a corpus of over 1 million documents. They will be added as soon as the training finishes.



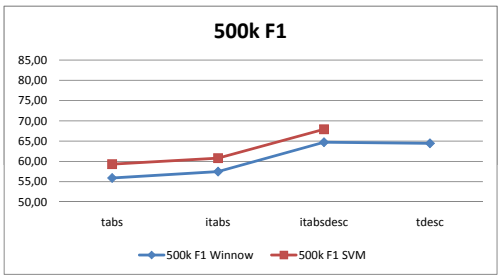
(a)



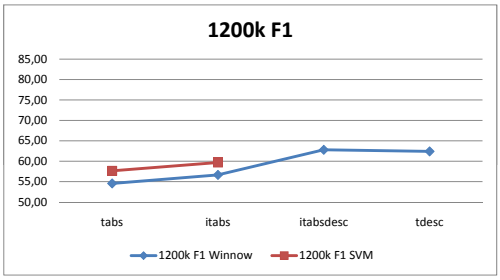
(b)



(c)



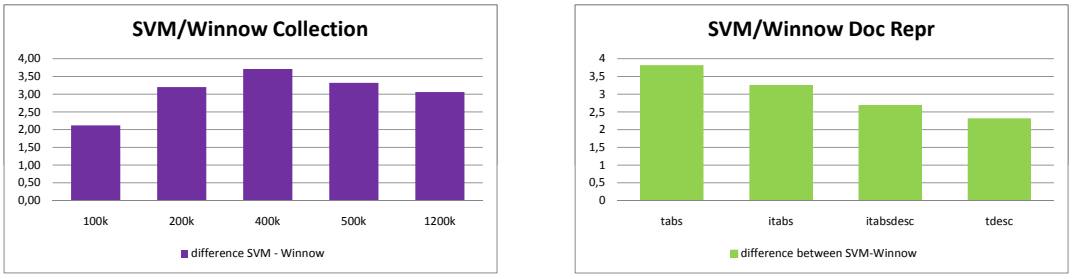
(d)



(e)

FIGURE 6. Comparison of SVM and Winnow in terms of the corpus size (F1)

3.3.2. *Feature space.* Figure 7(b) illustrates the effect of the document representation on the performance of the learner. The gap in F1 measure appears to be biggest in the briefest representation, i.e. `tabs`, and lowest in `itabsdesc` and `tdesc` (−1.5%). This suggests that Winnow can work effectively with a large feature space. If we look at the 100k collection, the difference between `tabs` and `itabsdesc` mounts to 7.31% when Winnow is used while SVM training only yields a difference of 5.88%. An increase of features, in contrast with an increase of classes, seems to have a smaller impact on the Winnow algorithm than on SVM training. With a bigger feature space, the SVM optimisation problem becomes more complex. The question arises whether reducing the feature size of document representations such as `itabsdesc` to a fixed number of terms (e.g. 600) would increase the SVM performance to the same amount as training did for Winnow with the default `itabsdesc` representation.



(a) Collection size

(b) Document representation

FIGURE 7. The relative difference between SVM and Winnow training

One more thing that can be shown by Figure 7(b) is that the margins between `tabs` and `itabsdesc` across the different collections is always bigger for Winnow. Only for the 500k collection, the difference between the margins of SVM and Winnow training is insignificant. As parameter tuning is concerned, one can say that while tuning the 100k `tabs` collection can already predict something about bigger collections in the case of SVM, this is not the case for a Winnow learner. SVM is thus a more robust learner and not that sensitive to fluctuations of the corpus as Winnow.

3.3.3. *200k → 400k.* Figure 8 shows the different reactions of the biggest classes increase (70 classes in 200k, 400 in 400k) that occurs in the subsets. Both in the dimension of the learning algorithms as in the document representations differences appear. One similarity that can be found is the fact that the gap in F1 measure between training with 200k and training with 400k narrows

when more features are used. An increase in classes introduces finer distinctions between them, which becomes easier with more features available. The size of this narrowing gap, however, is dependent on the learning algorithm that was used.

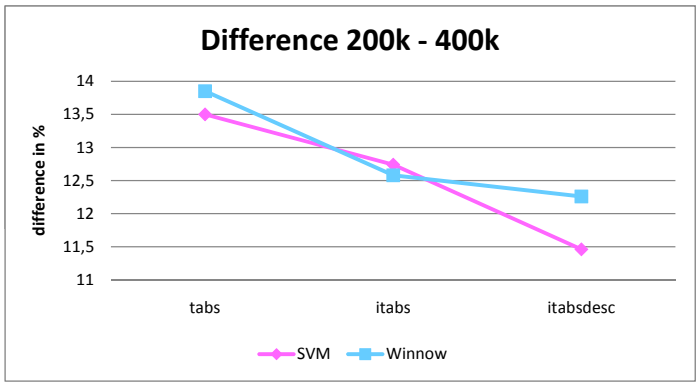


FIGURE 8. Difference in F1 measure in training with the 200k and 400k corpora

When the feature space is smallest, Winnow is slightly more affected by the increase of classes than SVM. Running from small to largest feature space, the SVM curve decreases more steeply than the Winnow curve. Whereas in Winnow training, the addition of inventors to the document representation has a bigger effect on the adaptation to an increase in classes than the addition of the description, SVM reacts more strongly on the explosion of the feature space when `itabsdesc` is used. The difference between 200k and 400k across the learning algorithms is maximized at this point. Again, this shows that SVM is more robust to the enlargement of the feature space than Winnow.

If only the F1 measure is taken into account, it can be concluded that although the SVM algorithm was implemented ad hoc, i.e. without testing the optimal parameters for each corpus, the results are already better than the experiments with the fine-tuned Winnow parameters (cf. [**Koster et al., 2003**]). For Winnow, parameters could also be tuned for each collection separately as the algorithm reacts stronger to changes in the feature space. A tuning for the 100k `tabs` corpus is included in Chapter 5.

TABLE 8. Relative difference in margins in Winnow and SVM training

corpus	algorithm	`tabs`	`itabsdesc`	difference
100k	SVM	78.93	84.81	5.88
	Winnow	75.23	82.54	**7.31**
200k	SVM	76.57	83.14	6.57
	Winnow	72.33	80.81	**8.48**
400k	SVM	63.07	71.68	8.61
	Winnow	58.48	68.75	**10.27**
500k	SVM	59.32	67.92	8.6
	Winnow	55.90	64.72	**8.82**
1200k	Winnow	54.56	62.8	**8.24**

3.3.4. *Precision and recall.* However, if one also considers precision and recall, rather than F1 only, more differences show up. The complete results are included in Tables 9 and 10. Figure 9 depicts the two tendencies that are at work in both algorithms.

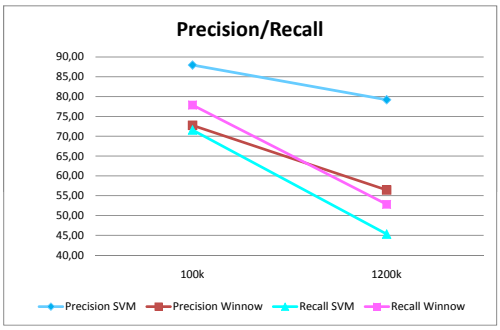


FIGURE 9. Precision/recall for 100k `tabs` and full corpus `tabs` trained with Winnow/SVM

First, both precision curves (dark colours) expose a less steep decrease in accuracy than the recall curves (fluorescent colours). Whereas recall drops with 26.27% for SVM and 25.06% for Winnow training, the difference in SVM precision is only 8.76%. Winnow precision, on the other hand, decreases by double the percentage of SVM precision, namely 16.32%. This rather indicates the extremely high precision of SVM training. Second, both of the Winnow curves (reddish ones) are situated between the SVM curves, indicating the extreme results of the SVM training. This

TABLE 9. Training results with Winnow

Coll.	Classes	Repr.	MICRO			MACRO		
			precision	recall	F1	precision	recall	F1
100k	66	tabs	72.75	77.87	75.23	61.23	64.10	62.38
	67	itabs	73.67	78.57	76.04	62.81	64.55	63.26
	66	itabsdesc	80.86	84.30	82.54	74.61	77.20	75.63
	66	tdesc	80.71	84.67	82.64	76.46	77.87	76.88
200k	115	tabs	70.29	74.49	72.33	56.23	59.31	57.39
	114	itabs	71.84	75.07	73.42	58.37	59.34	58.45
	115	itabsdesc	79.66	81.99	80.81	72.50	71.31	71.41
	116	tdesc	79.48	81.82	80.63	71.28	70.10	70.24
400k	375	tabs	59.20	57.77	58.48	46.93	43.24	44.58
	384	itabs	62.03	59.68	60.84	49.93	45.94	47.47
	380	itabsdesc	70.30	67.27	68.75	64.55	56.67	59.41
	381	tdesc	69.83	66.97	68.37	62.95	56.32	58.72
500k	574	tabs	58.14	53.83	55.90	45.88	40.55	42.61
	571	itabs	60.37	54.85	57.48	48.95	41.90	44.71
	575	itabsdesc	68.22	61.57	64.72	61.95	50.83	55.00
	573	tdesc	67.74	61.46	64.45	62.11	50.68	54.98
1200k	613	tabs	56.43	52.81	54.56	42.86	38.80	40.42
	615	itabs	59.64	53.94	56.65	46.47	40.66	43.03
	66	itabsdesc	66.37	59.59	62.8	59	48.75	52.75
	66	tdesc	65.63	59.49	62.41	58.51	48.61	52.53

raises the question as to whether parameter tuning could increase the recall without lowering the precision too much. A use case is included in Chapter 5.

3.3.5. *Detail MICRO Winnow.* When the MICRO result tables are looked at in greater detail, there are a couple more interesting trends visible. The training results for Winnow show that over the collection sizes, the recall is higher than the precision for 100k whereas the opposite scenario is found for the 1200k corpus. The recall thus decreases much more rapidly than the precision when the number of classes is increased. Moreover, the bandwidth between the smallest feature space and the biggest feature space varies in size between precision and recall. Averaged out over the 5

TABLE 10. Training results with SVM

Coll.	Classes	Repr.	MICRO			MACRO		
			precision	recall	F1	precision	recall	F1
100k	66	tabs	87.97	71.58	78.93	85.92	52.55	62.28
	65	itabs	88.18	70.80	78.54	84.88	51.77	61.99
	66	itabsdesc	90.40	79.86	84.81	92.03	65.33	74.39
	65	tdesc	90.31	79.9	84.79	92.26	65.64	74.65
200k	114	tabs	88.07	67.72	76.57	84.94	47.42	58.16
	114	itabs	88.42	68.48	77.18	84.82	48.48	59.00
	115	itabsdesc	90.61	76.80	83.14	90.66	58.54	68.04
	113	tdesc	90.51	76.79	83.09	86.16	58.63	67.5
400k	380	tabs	82.38	51.09	63.07	75.93	33.66	43.72
	383	itabs	83.18	52.59	64.44	77.11	35.19	45.54
	380	itabsdesc	84.47	62.26	71.68	81.32	45.77	55.70
	381	tdesc	84.31	61.84	71.35	81.53	45.58	55.68
500k	574	tabs	81.01	46.79	59.32	73.91	30.46	40.21
	574	itabs	82.13	48.27	60.81	77.50	31.58	41.79
	573	itabsdesc	83	57.48	67.92	80.14	41.37	51.57
1200k	615	tabs	79.21	45.31	57.64	74.84	28.26	38.02
	618	itabs	80.08	47.57	59.69	76.53	30.23	40.45

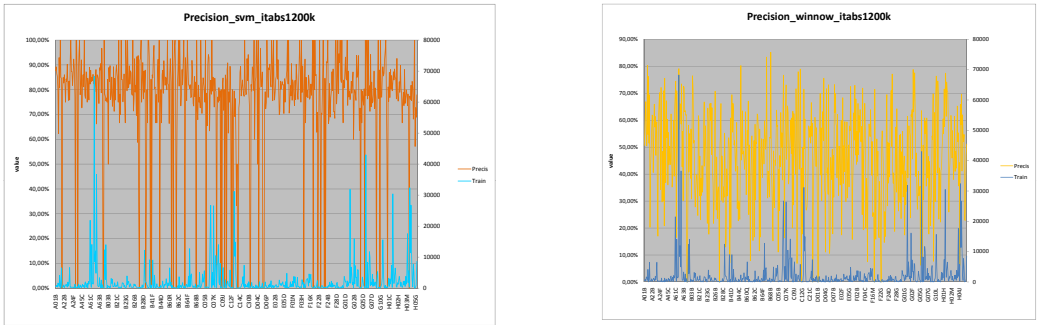
corpus sizes, the relative distance in precision is situated at 9.72% whereas this number is more than 2% lower for recall (7.59%). The addition of features (inventors and description) has thus a greater effect on the precision values than on the recall, which seems to be more stable in this regard.

3.3.6. *Detail MICRO SVM.* The SVM table repeats the finding that the precision is always greater than the recall, regardless of the size of the corpus. The gap between both measures does become bigger as the collection grows. This is due to the parameter settings of the SVM_{Light} learner. Chapter 5 explores better settings to improve the recall. The settings used in the baseline are the ones where F1 is maximized and therefore a good option for an ad hoc system. As opposed to the Winnow training results, the differences in feature space are much larger for recall than for precision: 9.81% vs. 2.26%. Worth noting is the fact that the increase of classes, i.e. corpus

size, is not perceptible in the precision scores that if the difference between `itabsdesc` and `tabs` is considered. SVM precision for 400k is situated at 84.47% for `itabsdesc` and at 82.38% for `tabs`, resulting in a difference of 2.09%. For Winnow, this difference lies at 11.10%, almost exact the same value as the SVM recall `itabsdesc-tabs` result (11.17%). One can conclude that if only precision is taken into account, training with SVM on a big corpus is extremely advantageous as SVM is a very robust learner and can deal with a large feature space without problems.

3.3.7. *MACRO.* Investigating the MACRO results, it becomes clear that also the Macro precision stays up very high in the SVM experiments. As the Macro measures take the harmonic mean over all the classes, this result suggests that SVM can learn examples that occur only in a few classes. On the contrary, training with Winnow works well on classes that have many examples but fails on sparsely populated classes. The Winnow algorithm is thus more effected by the imbalance in the training collection than SVM.

The following figures illustrate that the SVM classifier is able to learn most of the sub classes with a quite similar precision, even though the numbers of examples strongly vary among the sub classes, whereas the Winnow classifier is not that stable. The orange/red lines depict the precision values, while the blue curves represent the corresponding number of examples used in the training.



(a) SVM (b) Winnow

FIGURE 10. precision bandwidth with SVM vs. Winnow according to the number of training examples

The boxplot in Figure 11 illustrates the difference in bandwidth between SVM and Winnow precision results. The SVM plot is much more narrow but has more outliers than the Winnow plot. The detailed results reveal that the precision is 8 times 0.00% for the Winnow learner whereas

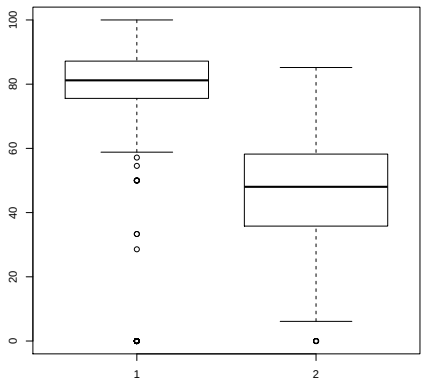


FIGURE 11. Boxplot of SVM(1) and Winnow(2) precision across classes

SVM training yields this value 43 times. At the other end, i.e. precision reaching 100%, we see that SVM achieves this 59 times while Winnow never reaches a precision that is up so high.

A typical example where SVM training yielded 100% is in the case of class A23P. SVM selected three train examples less than Winnow, 413 as opposed to 416. In total, 765 documents belong to class A23P. The reason why SVM precision is maximized is due to the fact that there were only 3 examples selected for class A23P, all of which were relevant. The Winnow results show that only 17 out of 100 selected examples were relevant, therefore yielding a precision score of 17%. What is also visible is that the SVM learner leaves aside more relevant examples as Winnow: 199 vs. 180 Relevant Not Selected.

Over the whole corpus, the RNS count for SVM is higher than the RS count (RNS 306787 > RS 278366), while the opposite result is found for Winnow training (RNS 269705 < RS 315804). The high precision of SVM training is reflected in the low number of false positives (NRS): 69243. Winnow training yields three times as many NRS documents: 213697.

TABLE 11. Detailed results for class A23P

learner	train ex.	precision	recall	RS	RNS	NRS
Winnow	416	17,00%	8,63%	17	180	83
SVM	413	100,00%	1,49%	3	199	0

3.4. Classification method. A single learning algorithm in a multi-label categorization usually produces poorer categorization results compared with a combination of multiple learning algorithms applied on the same categorization problem [**Dietterich, 2000**] (p.c. Bernhard Pflugfelder).

All the experiments described above were produced with ensemble (flat) learning. This section briefly introduces the second meta-learning approach: hierarchical learning. Here, the categoriser first decides on the section level, then on the class level, etc. There are thus less possibilities to be considered at each level. The results in Figure 3.4 show that the performance of the hierarchical learners is in fact worse than the flat learners when the F1 measures are taken into account. This is probably due to the fact that there was no back propagation of error used in the algorithm. Once it has made a wrong decision, it is impossible to jump back to an earlier node in order to readjust the path through the hierarchy.

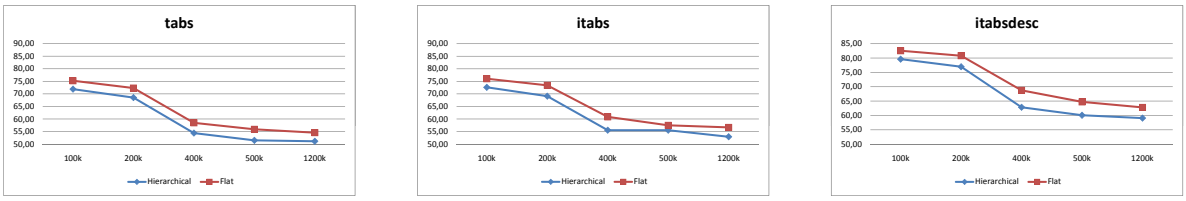


FIGURE 12. Hierarchical vs. Flat Classification

The positive element in this approach is the exponential time savings at both learning and classification time [**Esuli et al., 2008**]. This is due to the "local" policy for selection negative examples [id.]:

> given a category c, its negative training examples are by default identified with the training examples that are negative for c and positive for the categories sibling to c in the hierarchy.

3.5. Time estimation. Although SVM scores generally better than Winnow there is one downside, namely its speed. This is related to the difference in architecture of both algorithms. The Winnow algorithm is in fact extremely simple. It changes its discrimination function by multiplication and is linear in the number of features it can process. Moreover, "the time complexity of the Winnow algorithm does not depend on the number of features, but rather on the average number of non-zero features per data, which is usually quite small" [**Tzhang et al., 2002**]. Although the SVMs used in the experiments all had linear kernels, the algorithm has a squared complexity in the number of features.

The CPU times of the experiments have been measured by the LCS application itself. Not surprisingly, the calculation time increases drastically when the description is added to the document representation.

Figure 13 summarizes the CPU times for `tabs`, `itabs` and `itabsdesc` Winnow and SVM experiments across all five document representations. If the smallest document representations, i.e. models with the smaller feature spaces, are considered, the calculation times of LCS Winnow and SVM are quite similar where both algorithms have needed approximately 24 hours to process an experiment. Contrarily, the `itabsdesc` document representation (large number of terms) causes a strong increase of the calculation time for both the LCS Winnow and SVM . The CPU time of LCS Winnow is, however, still far more limited and, for instance, in case of the sample collection 500k the time proportion is already close to 1:2 compared with SVM.

The absolute time an experiment needs is even higher because the import of the data and the pre-processing of the train and test collection strongly depend on the I/O speed. On the LDC a quite slow I/O rate has been observed due to some other heavy I/O-related processes (e.g. an index process) that were running at this time. The training time can definitely be reduced if a separate machine will be used having also fast hard discs. Also a multi-threading training would be promising. Although the LCS claims to use multi-threading it appears to be insufficiently implemented as no significant time gain was observable compared with single process run.

However, a time frame of 10 day for training a categoriser might be still acceptable if the categoriser need not to be updated in short intervals. For instance the categoriser used at the WIPO has not been retrained over the last years and, therefore, the training time should not be a decisive decision factor.

4. Significance Testing

In order to verify the validity of some of the results presented above, statistical significance tests were implemented by means of the statistical package R (http://www.r-project.org/). The two main issues that needed statistical backup were:

- Document representation: distance of F1 curves
- Learning algorithm: difference Winnow/SVM

4.1. Document Representation. Figure 5(a) depicts the accuracy the Winnow algorithm achieves across the different corpus sizes and document representations. In order to know whether the distance between the different document representation curves is significant, there are a range

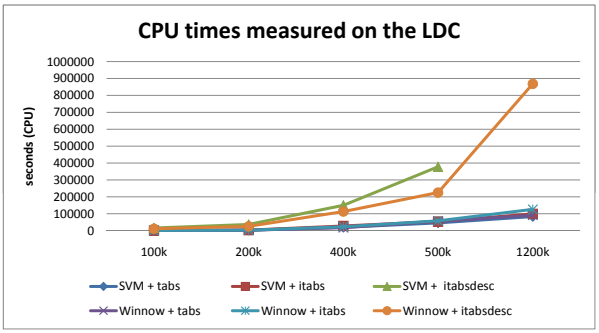


FIGURE 13. CPU time

of parametric and non-parametric statistical tests that can be applied. First of all, it has to be found out whether or not we are dealing normal distributions. Therefore, a Shapiro-Wilk normality test was carried out. As the null hypothesis holds that the population normally distributed and the alpha level was set to 0.05, the alternative hypothesis (i.e. the sample does not come from a normal distribution) can be accepted if $p \leq 0.05$. The results of the normality test are summarised in Table 12 and show that the null hypothesis cannot be rejected for any of the samples. This implies that all document representations are normally distributed across the corpus sizes.

TABLE 12. Shapiro-Wilk normality test (doc representation)

sample	p-value
`tabs`	0.1358
`itabs`	0.1459
`tdesc`	0.2219
`itabsdesc`	0.2343

Because of this normal distribution, a parametric statistical test can be carried out. The student's t-test was used here to test the divergence in means across the different document representations. Figure 14 illustrates the distribution of the means (black horizontal line) and the standard deviations (boxes) in the samples. Note the skewedness of the distributions towards the smaller accuracies.

The null hypothesis of the t-test is that sample A and B come from the same distribution. Two samples become significantly different if the alternative hypothesis can be accepted, i.e. if

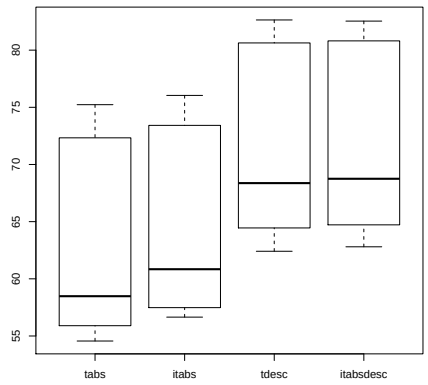


FIGURE 14. Boxplot of the four document representations

TABLE 13. P-values of the t-test

	tabs	itabs	tdesc	itabsdesc
tabs	–	0.7975	0.2013	0.1873
itabs	0.7975	–	0.2781	0.2597
tdesc	0.2013	0.2781	–	0.9704
itabsdesc	0.1873	0.2597	0.9704	–

TABLE 14. P-values of the Wilcoxon rank sum test (doc representation)

	tabs	itabs	tdesc	itabsdesc
tabs	–	0.5476	0.2222	0.2222
itabs	0.5476	–	0.2222	0.2222
tdesc	0.2222	0.2222	–	0.8413
itabsdesc	0.2222	0.2222	0.8413	–

the two means differ significantly. Results are included in Table 13. It becomes clear that none of the p-values is smaller than the 0.05. Therefore the differences in accuracy between the document representations cannot be judged significant. However, there is indeed a difference in means, as can be seen in Figure 14 and from the additional information R provides for the Welch Two Sample t-test. The means for `tabs`, `itabs`, `tdesc` and `itabsdesc` are respectively: 63.3, 64.9, 71.7 and 71.9.

TABLE 15. Shapiro-Wilk normality test (learning algorithm)

	100k	200k	400k	500k	1200k
winnow	0.07448	0.09275	0.2001	0.3223	0.4707
svm	0.04522	0.0616	0.1338	0.3110	–

TABLE 16. P-values of the Wilcoxon rank sum test (learning algorithm)

100k	200k	400k	500k	1200k
0.3429	0.3429	0.3429	0.4	0.3333

However, the acceptance of the null hypothesis can be caused by another factor. The size of the samples is rather small as we only have one F1 measure for each corpus size, resulting in five data points per sample. Now, each of the samples was duplicated so it contained ten data points instead of five. The results showed this time a significant difference between `tabs` and `itabsdesc` (t = -2.1633, df = 17.94, p-value = 0.04426). The difference between `tabs` and `itabs` remains insignificant, although their means differ by 1.6 %. We can thus conclude that the only real significantly better performing document representation is `itabsdesc`, as also the difference between duplicated `tabs` and `tdesc` is insignificant (t = -2.089, df = 17.972, p-value = 0.05121).

P-values of a non-parametric test are included for reference in Table 14. The Wilcoxon rank sum test was used to double check the results of the t-test. Also here tests with the five point sample data set were insignificant. In the duplicated tests, the difference between `tabs` and `itabsdesc` was equal to 0.05, and could therefore not cancel out the null hypothesis (W = 24, p-value = 0.053).

4.2. Learning Algorithm. To test whether there is a significant difference in using a Winnow algorithm or SVMs, the data from Figure 6 was used as 10 data samples, each with 3 to 4 data points. Each sample represents an algorithm trained on one of the five corpus sizes (2*5) evaluated for 3 or 4 document representations. As the samples are so small, the results of the normality test have to be treated with care. A t-test cannot be carried out as the 100k SVM sample does not correspond to a normal distribution.

Results of the Wilcoxon rank sum test in Table 16 show that the difference between Winnow and SVM training was never significant. However, this result should perhaps not be taken too strictly as the sample sizes were so small.

CHAPTER 5

Use Case

The results in Chapter 4 showed that, although training with SVM always yielded a higher accuracy, the experiments were run using ad hoc parameter settings that caused the precision to stay up relatively high while the recall dropped under 50%. Depending on the goal of the experiment, such a low level of recall may not be ideal. Therefore, the first section in this chapter focuses on improving recall in SVM experiments. The second section goes into parameter tuning for Winnow experiments.

1. SVM Tuning

Our previous experiments were conducted with three parameters that affected the workings of the SVM_{light} package: the soft margin parameter (C), the cost parameter (J) and the type of kernel (T). The latter is kept at its default value, i.e. linear. As the type of problem we are trying to solve is linearly separable, using a polynomial or radial-based kernel would not bring an increase in accuracy but rather delay the classification even more.

The parameter tuning was carried out with the smallest collection (100k) to speed up the process. Looking back at Section 3.3, the robustness of SVM training will probably guarantee the validity of the results for bigger sized corpora.

1.1. C tuning. The default setting for the soft margin parameter in $SVM^{light}$is $[avg.x * x]^{-}1$. This parameter setting tries to find the line that best separates the positive and negative training examples. To maximise this difference (x), the value is squared and inverted. This is done for every training example that has a counterexample in its neighbourhood, i.e. only for support vectors. In general, C can be any floating number that is bigger than 0. In order to get a first impression of possible effects of the C parameter, values were tested between 2.5 and 50 with a step size of 2.5.

At first sight, it becomes clear that precision stays higher than recall across the complete interval. Both curves are separated neatly from each other by the green F1 curve that stays quite stable in the course of the tuning (max. 78.8, min. 73.69). While the precision drops more than 10% for C values that ly between 2.5 and 15, the recall rises less than 3%. The results between 15

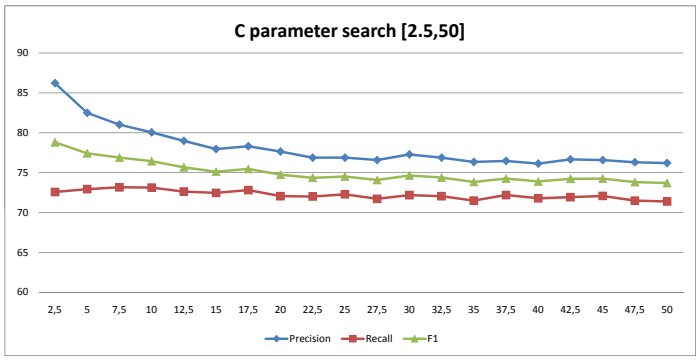


FIGURE 1. Tuning of the C parameter

and 50 do not add much more to the parameter tuning: precision fluctuates in a slightly downward trend from 77.96% to 76.2%; recall finds its maximum at C=17.5 (72.81%) and fluctuates up and down across the interval (71.38% at C=50).

Most of the changes take place in the beginning of Figure 1.1. Therefore, a more detailed parameter tuning between 0.1 and 2.5 with steps of 0.1 is carried out. The C value used in the baseline experiments is marked in yellow. This is the point where the F value reaches its maximum.

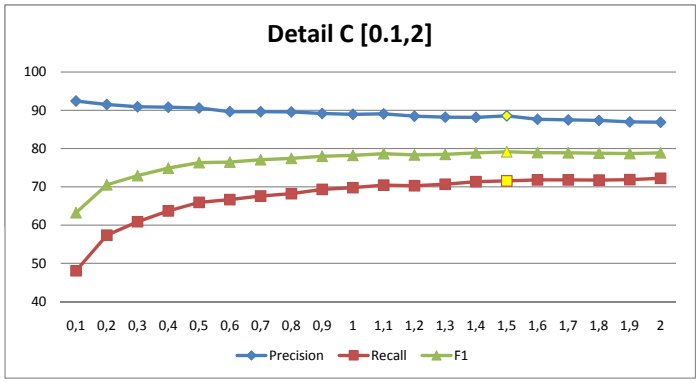


FIGURE 2. Detail Tuning C in the interval [0,2]

TABLE 1. Results of the C parameter tuning

	C=1.5	max
recall	71.56	73.16 (C=7.5)
precision	88.56	92.43 (C=0.1)
F1	79.16	79.16 (C=1.5)

For the smallest values of C, the distance between precision and recall is greatest. The maximum precision (92.43%) and minimum recall (48.06%) values are situated at C=0.1. The recall curve then rises rapidly between 0.1 and 1 up to 69.78% (+ 21.72%). Precision drops only by 3.5%. The highest recall in this interval is situated at C=2 (72.22%).

The results of the C parameter tuning did not reveal results that can be taken as significant proof for a better recall level than the one reached in the baseline experiments. An increase in recall of 1.6% can be reached by using C=7.5. Note that this result holds for the `tabs` 100k corpus. Although SVM is a robust learner and similar improvements will probably be found for larger corpora, this was not explicitly verified here. Table 1 summarises the maximum attainable precision, recall and F1 values and compares them to the baseline.

1.2. J tuning. The second parameter that can be changed in the SVM algorithm used in the experiments is the cost factor. By changing J, the misclassification of a positive example can be punished more or less severely. The cost of misclassifying a positive example is determined as $C - + = J * C$, while the misclassification cost of a negative example remains unaltered [**Joachims, 2002**]. By default J equals 1.0.

Figure 1.2 shows the results of a tuning in the interval [1,10] with step sizes of 1. The highest recall value is found at J=8. Although the goal of this parameter tuning is to get the recall possibly at its highest level, the precision, and therefore F1, value should in the best cases not drop too much after all. Therefore, J=5 seems a reasonable choice. In general, the curve between J=5 and J=10 does not change that much to the accuracy anymore.

Remarkable is the shape of the curves at J=4. The recall increases along the complete curve, only between J=3 and J=4 it decreases. At J=4 precision, recall and F1 are almost equal (precision= 75.44; recall = 76.22, F1 = 75.83).

When the two parameter settings where the recall is highest are combined in an experiment (C=7.5, J=8), the accuracy is lower as when either C or J is set to its default value. precision lies at 77.14, recall at 73.71 and the F1 measure at 75.39. Two large parameter settings do not

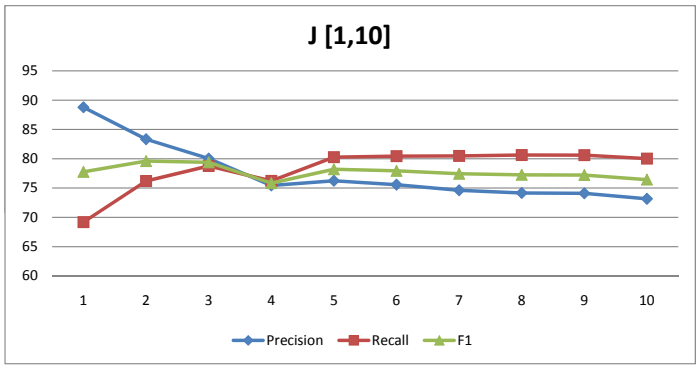


FIGURE 3. Tuning of the J parameter

TABLE 2. Results of the J parameter tuning

	J=1	max
recall	69.19	80.63 (J=8)
precision	88.78	88.78 (J=1)
F1	77.77	79.61 (J=2)

seem to work extremely well. Therefore, other pairs of C and J were tried out. At the point where the recall curve does not change that much anymore (J=5), different settings of the soft margin parameter are experimented with. The results can be analyzed in Figure 4.

The figure shows that the bigger the C values are, the closer precision and recall grow. When J values are taken into account, it becomes clear that the smallest settings are mirrored by the smallest distance between both curves: when J equals 5, the precision and recall curves are the middlemost of all curves when C = 0.5 and 1. At larger values of C, the recall curve overlaps with J=6 and J=8. If recall is important, a large J setting and a small C setting are preferable. The **highest recall** level achieved in this tuning was **83.27%** with settings **C=0.5** and **J=8**. This value can possibly be increased when even bigger values of J are tried in combination with smaller C parameters. precision was highest at J=5 and C=2 (77,25%). This is more than 10% below the **maximal precision** value (**92.43%**) that was achieved with settings **C=0.1** and **J=1**. This

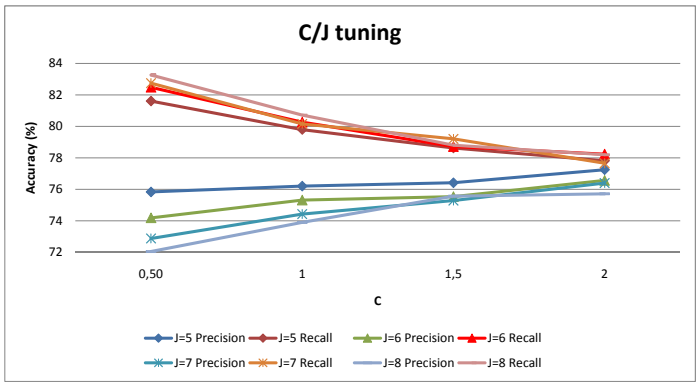


FIGURE 4. Pairwise tuning of C and J

means that no extra cost was incorporated and the soft margin value is very small (i.e. almost a hard margin (C=0)).

As it looks as if the curves are not yet at a point of convergence at the left side of the x-axis (small values of C), two more experiments were conducted that were expected to improve the recall even more (and worsen the precision). Results are summarised in the following table.

TABLE 3. Optimal recall Settings

Settings	precision	recall	F1
J=10; C=0.25	68.54%	**85.21%**	75.97%
J=12; C=0.05	62.46%	**85.55%**	72.20%

1.3. Test on bigger corpus. It was verified how the best precision/recall settings on the 100k would score on the 500k corpus. Is the accuracy on 500k still proportionally higher with tuning as without? To maximize the precision, the values C= 0.1 and J= 1.0 were used. The maximum recall was in the 100k tests achieved with C= 0.25 and J = 10.0. These values were tested on the 500k `itabsdesc` corpus. Results are summarised in the table below.

The experiment that was run with the optimal precision settings yielded an accuracy score of 89.34% on the 500k corpus, which is 6.3% higher than the baseline run on the 500k itabsdesc corpus. On the 100k corpus, the difference between the baseline and the optimal precision settings

TABLE 4. Maximum precision and recall settings on `itabsdesc` 500k

	precision	recall	F1
max P	89.34%	30.97%	46%
max R	62.92%	73.5%	66.14%
baseline	83%	57.48%	67.92%

was situated at 3.6%. This shows that the optimal precision settings hold for a bigger corpus as well. Moreover, they even seem to widen the difference between baseline and tuning even more.

The maximum recall settings yielded an increase in accuracy of 16.02% on top of the baseline. Also here the gap between baseline and tuning is bigger for the 500k `itabsdesc` corpus than it was for the 100k `tabs` (13.63%). A last thing to note is the stability of the F1 value, losing only just over 1% of accuracy after the tuning has taken place.

2. Winnow Tuning

Although there are two good reasons that would make a grid search for the Winnow parameters unnecessary, a tuning for α and β is nevertheless included below. Firstly, the parameter settings used in the baseline experiments were borrowed from [**Koster and Beney, 2007**], who performed various Winnow tests with patent applications on the LCS. The corpora they used were, however, quantitatively from a different kind than the material used in our baseline experiments. They used abstracts (average length of 143 words) and full-text ($\pm$2000 words) patent applications from the European Patent Office but only classified on 16 classes with 1000 documents/class, therefore ignoring the unbalancedness that exists between EPO classes. With only 16 000 documents, it is difficult to estimate the workings of their parameter tuning results on a corpus of 1 250 000 documents.

Secondly, a tuning might not be ideal due to the type learner that Winnow proved to be in the experiments. As the Winnow results were more affected by an increase of classes, i.e. a bigger corpus in our case, parameter generalizations are rather hard to draw. Despite these two arguments, a tuning on the 100k `tabs` corpus was carried out, in order to get an impression whether the *Koster parameters* were valid on a corpus that was almost 10 times bigger than the one used the [**Koster and Beney, 2007**] paper.

The thick threshold was kept at 0.2, i.e. $\theta^+ = 1.1$ and $\theta^- = 0.9$ and only α and β were tuned. In the first tuning experiments it became clear that a constraint had to be built in as the LCS got stuck in calculating thresholds that were enormously big. This constraint was chosen based on the

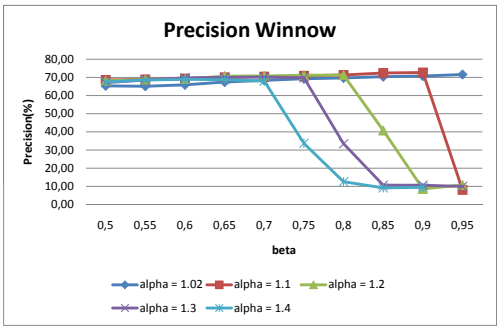


FIGURE 5. Primary results of Winnow tuning

primary results included in Figure 5. The accuracy consistently dropped down at points where α was smaller than $\frac{1}{\beta}$: e.g. α=1.4, β=0.75 ($\frac{1}{\beta} = 1.333\ldots < \alpha$). Therefore the tuning application was written so it did not consider parameter pairs if $\beta < 1.0/\alpha$. As α has to be bigger than 1 and β smaller than 1, following parameter range was tried:

$\alpha \in [1.02, 1.05, 1.1, 1.15, 1.2, 1.25, 1.3, 1.35, 1.4, 1.45, 1.5, 1.55, 1.6, 1.65, 1.7, 1.75, 1.8, 1.85, 1.9, 1.95, 2.0, 3.0]$

$\beta \in [0.1, 0.20.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]$

Both graphs in Figure 6 show that the larger β becomes (x-axis), the higher the accuracy climbs up. Three α parameters reach the top right of the figures: α=1.02, α=1.05 and α=1.1 (the default value). For recall, the highest accuracy (78.73%) is reached with α=1.05 and β=0.9. The default settings (α=1.1 and β=0.9) achieve the second highest recall value: 78%. It is only at the highest β value that the α=1.05 curve takes the overhand. The third highest recall is found for the steepest ascending curve: α=1.02 (76.84%). The form of the recall curves tell us that learning with α=1.05 and β=0.95 could perhaps add some extra points to the accuracy.

Precision on the 100k `tabs` corpus has slightly less distinctive top results as recall: α=1.1: 72.23% α=1.05: 71.43% and α=1.02: 70.93%. The maximum accuracy for precision also lies considerably lower than the maximum level of recall that was achieved. This result contrasts sharply to the SVM tuning, where recall could be optimized to 85% but the maximum precision value was even situated at 92%.

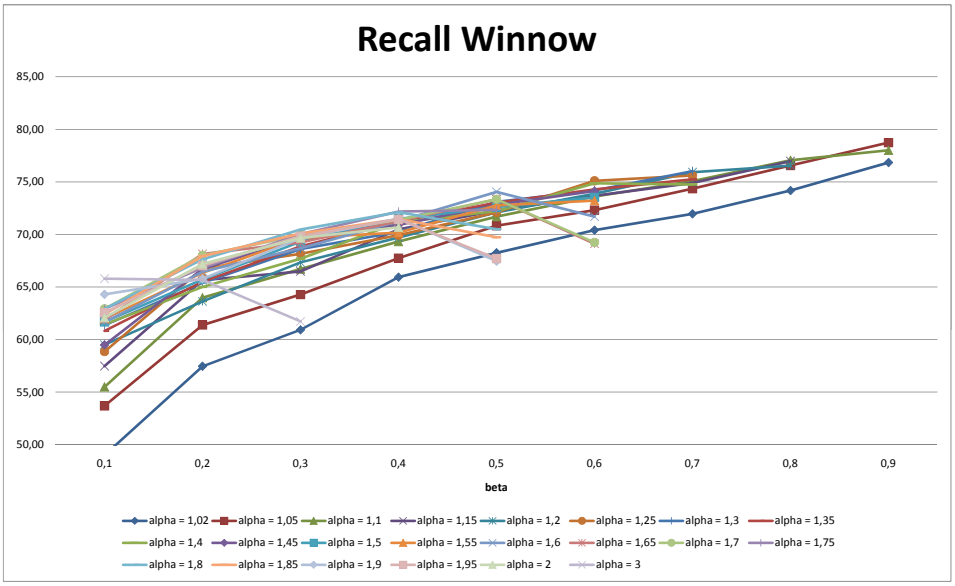
The F1 accuracy (graph not included) was almost the same for α=1.1 and α=1.05 (75.0% vs. 74.9%). The default values achieve thus the highest F1 results. In general, it can be said that the *Koster parameters* seem to hold for a bigger corpus. The biggest question of course is whether

these values are also optimal for the full (1200k) corpus. An automated corpus-based tuning would be the best way to get to the optimal results.

3. Discussion

As Koster and Beney [**Koster and Beney, 2007**] wrote, "Tuning is much more important than term selection". We have indeed seen that especially for SVM parameters, a increase in accuracy of 14% on top of the baseline can be achieved, depending on what the preferences of the user are.

As the goals of the study were trying to estimate the effects of different document representations and two different learning algorithms, there are still many questions that remain open: what happens for other forms of Term Selection? What is the effect of tuning on MACRO instead of MICRO-average, i.e. when small categories are emphasized instead of big ones? Also the effect of Term Weighting and Document Size Normalization technique could be investigated? More theoretical analysis is needed in those fields.



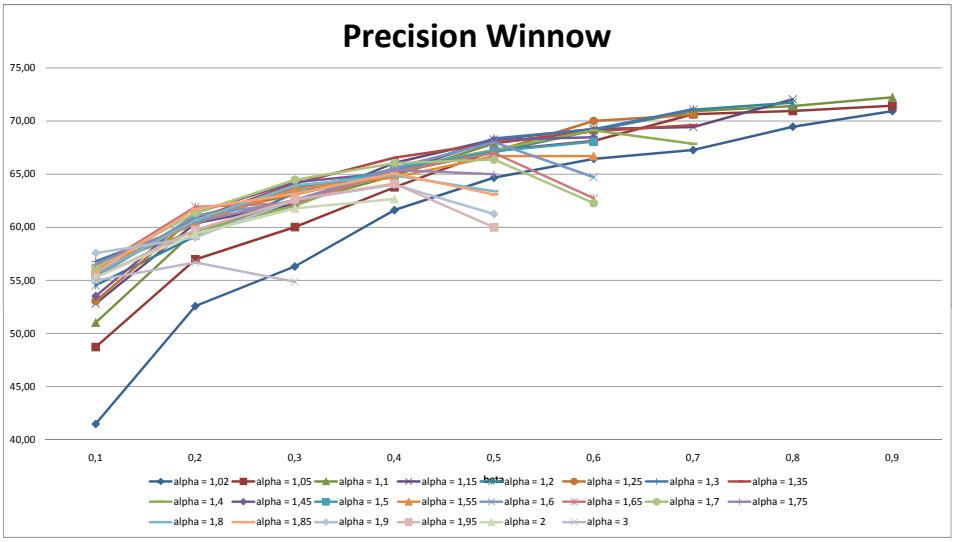


FIGURE 6. Winnow Tuning

CHAPTER 6

Baseline extensions

Next to the parameter tuning, there are more experiments possible that could improve the baseline results. One problem for instance with the SVM experiments, for instance, was the scalability of the model. The more terms, the more expensive the optimization function becomes. Therefore, one extension could be to perform an aggressive terms selection on the train set before training the model. Experiments with the `itabsdesc` 100k corpus are presented in Section 1. The second extension exists of a validation of the baseline results. A random categoriser was built that delivered similar results as the LCS Winnow classifier. Similar in a sense that the amount of classes a test document was given was simulated as well as the probability distribution of the class labels. The results of the experiments with the random categoriser are included in Section 2.

1. Aggressive Term Selection for SVM

As the collection size was growing, SVM needed much longer than Winnow to train the classification model. This could mainly be due to the increased feature space the classifier had to deal with. Figure 13 in Section 3.5 showed that, whereas for the small corpus collections the CPU time for SVM and Winnow are relatively equal, the calculation time of SVM will increase to approximately the double of LCS Winnow when the feature space grows significantly. To cut the time calculation, a more aggressive selection of terms could improve the scalability of the SVM classifier. The experiments in this section illustrate how this is possible and what the effects are.

First, the actual term distribution has to be considered before any experiments can be set up. The terms used in the training set of the `itabsdesc` 500k collection are displayed in Figure 1. There are seven classes that contain more than 80 000 terms in their profile: A61P (140915), C07D (128693), C12N (113284), C07K (105757), C08L (83381), C07C (97610) and G01N (99025). The biggest sub class according to the number of terms is A61P. This class comes second in the document ranking with 49363 examples. The class with the highest number of examples was A61K, absent in the 500k collection.

In order for SVM to work more efficiently it was decided to cut the number of terms per class, i.e. the local term maximum, down to at least 20 000. This means for the 500k collection that 91

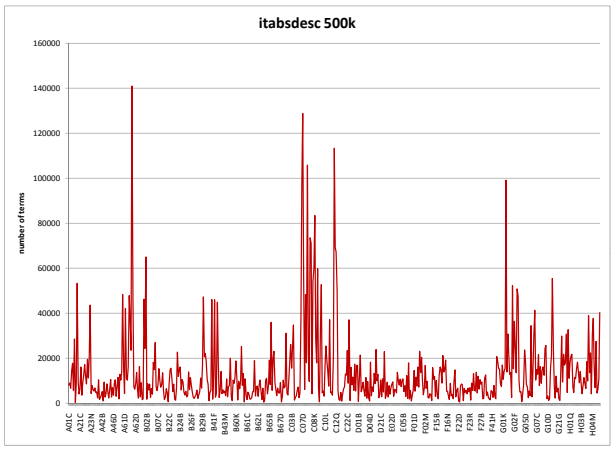


FIGURE 1. The distribution of terms for the `itabsdesc` 500k collection

TABLE 1. Statistics of various MaxTerms settings for 100k `itabsdesc`

MaxTerms	% of affected classes	total number of affected classes	total number of terms	nb of terms reduced	% of terms affected
5000	80%	53	310896	606699	66%
10000	44%	29	504671	412924	45%
12500	35%	23	566748	350847	38%
15000	24%	16	615645	301950	33%
17500	21%	14	653069	264526	29%
20000	14%	9	678171	239424	26%

classes would have to be reduced in terms. This number corresponds to 16% of the total number of classes.

Second, experiments with smaller samples were set up as to get an idea of what term selection can actually do to the accuracy and performance of an algorithm. The 100k corpus of `itabsdesc` was used for this task, as it had a training that was 20 times faster than the 500k corpus on the SVM baseline experiments. One would assume that an upper bound of 20 000 would be relatively loose for this corpus as there are much less terms involved (917 595 vs. 7 508 385). However, the construction of the subsets was done so that the proportion of big and small classes would be similar in all the corpora.

TABLE 2. Accuracy + CPU time on the 100k corpus

MaxTerms	MICRO			MACRO			CPU time
	precision	recall	F1	precision	recall	F1	
local 5000	90.50	79.98	84.91	90.12	64.54	72.85	12949
local 10000	90.19	79.66	84.6	90.92	64.98	73.75	13881
local 12500	90.28	79.59	84.6	90.62	64.39	73.05	13920
local 15000	90.25	79.59	84.59	90.99	64.62	73.16	13959
local 17500	90.28	79.89	84.76	90.05	63.9	72.52	14063
local 20000	90.62	79.74	84.83	89.69	65.65	73.96	16007
baseline	90.40	79.86	84.81	92.03	65.33	74.39	16052

Table 1 shows the number of affected classes for six different local MaxTerms settings on the 100k corpus. The baseline system for 100k `itabsdesc` contains 66 classes and 917595 terms in total. There are different degrees of term selection, depending on the aggressiveness. The percentage of classes that is cut to a lower number of terms ranges from 14% for MaxTerms 20000 to 80% when maximally only 5000 terms are selected. The latter functions with one third of the amount of global terms of the baseline system.

If exactly the same percentages of affected classes are chosen for the 500k corpus, the following MaxTerms settings arise: 4677, 9406, 11107, 15510, 16782 and 21727. As these settings, as well as the percentage of reduced terms, do not diverge that much from the ones used on the 100k corpus, they can reliably be used for the bigger corpus. It was chosen here to work with the same MaxTerms numbers, and therefore only approximating the relative percentages.

The accuracy of the aggressive term selection on the 100k corpus as well as the CPU time are included in Table 2. When only the micro results are considered, all percentages are equal for precision (90%), recall (79%) and F1 (84%). It seems thus that even the most aggressive cut in terms does not affect the accuracy at all. The only difference lies in the CPU time of the different training runs. Training with a maximum of 5000 terms per class needs ±3000 seconds less than with the baseline system, or even with 20 000 MaxTerms. This is a time reduction of ±20%. Also the macro results do not differ very much across the different term limits.

It should of course be tested now whether the results on the 100k corpus also hold on a bigger corpus. This was not tried out in this study as it would take up too much time and memory on the server to rerun such a big experiment. However, a time reduction of 20% would still not be sufficient

to equal Winnow CPU time on the 500k `itabsdesc` corpus. The baseline SVM experiments needed almost double the amount of time needed for a Winnow experiment with the same corpus size. Aggressive term selection can thus be a step in the right direction but other methods to cut the CPU time could be considered too. An example would be to change the maximum quality degree on class level. This could lead to the elimination of more noisy terms if the threshold is raised by a certain amount. Further research is necessary to find out the exact threshold levels that would make further contributions to the time reduction problem.

2. Random classification

The opposite of a research-driven classifier is a system that predicts categories at random for a given document. No meticulous engineering but rather a good dose of luck is the key to a random classification engine. If our baseline system as presented in Chapter 4 is able to render a better classification than a random categoriser, the results presented earlier are more than mere chance and can be reproduced in future research.

An application that could simulate a random categoriser was written in Java using the Colt library (http://acs.lbl.gov/ hoschek/colt/). A train catalog was taken as input file (document name + categories to which it belongs) to build a probability density function (pdf) of the training distribution. Next, the test catalog was read in and a discrete random sampler selected a number from the pdf that could then be linked to a category name that had exactly the selected number of training documents. The amount of classes that each test document could get assigned was also generated randomly from another pdf that captured the distribution of the number of categories per training document. As mentioned earlier, each training document had on average 1.9 categories.

The results included in Figure 2 were generated using the `itabsdesc` 1200k collection based on Winnow preprocessing. The train and test catalogs[1] that are delivered just before the LCS Winnow training starts, were extracted and used as input to the random classifier. The difference between the baseline classifier and the random one is big. For the smallest collection, the F1 measure of the random classifier reached 6.81%, while the LCS achieved 82.54%. The difference between 100k and 200k is proportionally also bigger for the random classifier. More classes means more chance to be wrong in randomly picking a category. There are only minor differences between the 400k, 500k and 1200k collections. This is probably related to the increase of classes at 400k, that makes further smaller increases less significant.

[1] A catalog is a file in which each line starts with a document name followed by a list of class names the document belongs to.

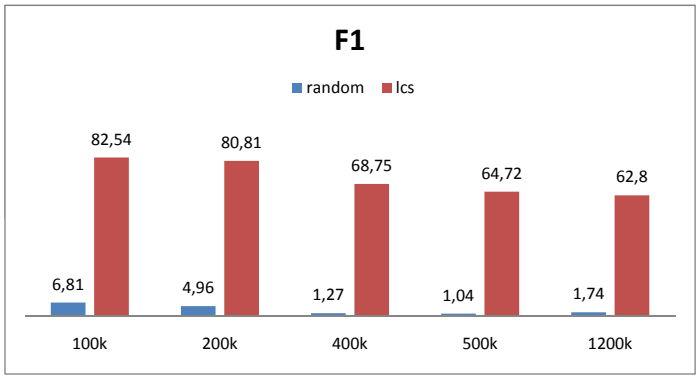


FIGURE 2. F1 results random vs. LCS Winnow classification

All results are summarised in Table 3. As the results are very low in general, Micro and Macro results do not differ much this time. precision is always higher than recall which indicates that the random classifier is slightly better in returning correct classes than making sure that as many correct classes as possible are found.

TABLE 3. Results from the random classifier

	MICRO			MACRO		
Size	precision	recall	F1	precision	recall	F1
100k	7.13	6.51	6.81	7.14	6.46	6.68
200k	5.27	4.69	4.96	5.26	4.76	4.89
400k	1.46	1.12	1.27	1.49	1.13	1.20
500k	1.23	0.90	1.04	1.22	0.90	0.92
1200k	2.07	1.50	1.74	2.07	1.47	1.50

The gap between the random classifier and the LCS experiments is huge. A random application is successful if the results it generates are more than chance ($> 50\%$). In the case of multi-classification, however, the scenario is more complicated as there are multiple categories that the classifier has to predict.

CHAPTER 7

Conclusions

This study investigated the effect of different document representations in combination with two machine learning algorithms, Balanced Winnow and SVM. Both algorithms try to find a discriminative function between positive and negative training examples but do so in a very different way. Winnow works by changing the separating line up and downwards (promote vs. demote) as the weights of the features are either multiplied by a promoting or a demoting factor, depending on whether they are on the right or the wrong side of the line. This is fast and flexible. SVM has a more complicated optimization function. The distance to the discriminative function has to be calculated for each example and the largest margin between the support vectors has to be found. The construction of an SVM model is time-consuming and restricted to a smaller feature space.

Comparing the results for both algorithms, the following key conclusions can be made (cf. [**Pflugfelder, 2009**]). The LCS Winnow experiments can be summarised as follows:

- Moderate to low micro-averaged precision and recall results have been achieved
- Moderate robustness in case of an increasing corpus size
- The representation `itabsdesc` significantly outperforms all other document representations.
- Parameter tuning can be done to improve precision (and probably recall)
- Training time is acceptable with approximately 10 hours CPU time on the LDC.

The SVM experiments, on the other hand, entail the following conclusions:

- High precision has been found over all sample collections.
- Exceptional balancing of the precision values over all sub classes.
- SVM is highly robust in terms of number of sub classes to be learned.
- The representation `itabsdesc` significantly outperforms all other document representations.
- Parameter tuning is indispensable

As has been shown, a careful parameter tuning on the training corpus can have a positive effect on the classification results. Therefore, every practical classification system should be made

self-tuning. Rather than using default values, automatic tuning methods are the best way to deal with variation in corpora or applications consistently. Moreover, depending on the needs of the user, precision or recall directed tuning can be made possible.

In order to speed up the training process, farming could be used more efficiently. The creation of the index can happen on several machines (different documents on different machines) and the multiple sub-indexes can be merged into one final index (cf. Google's MapReduce). Also in learning class profiles, classes can be trained separately, i.e. independent from each other, and data can be mirrored in order to avoid memory problems. The final testing then, once the model is stored, can be done on one machine.

Other than a bag of single words, the use of semantic information in the creation of the index is another issue that deserves attention in future investigation. The incorporation of such linguistic information is the next component that will be added to the LCS. Primary research into the use of phrases has been conducted by Koster et al. [**Koster and Seutter, 2003**].

Bibliography

[Bishop, 2006] Bishop, C. M. (2006). Pattern Recognition and Machine Learning (Information Science and Statistics). Springer.

[Cai and Hofmann, 2004] Cai, L. and Hofmann, T. (2004). Hierarchical document categorization with support vector machines. In CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management, pages 78–87, New York, NY, USA. ACM.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. In Machine Learning, pages 273–297.

[Croft et al., 2009] Croft, B., Metzler, D., and Strohman, T. (2009). Search Engines: Information Retrieval in Practice. Addison Wesley, 1 edition.

[Dagan et al., 1997] Dagan, I., Karov, Y., and T, D. R. (1997). Mistake-driven learning in text categorization. In In EMNLP-97, The Second Conference on Empirical Methods in Natural Language Processing, pages 55–63.

[Dietterich, 2000] Dietterich, T. G. (2000). Ensemble methods in machine learning. pages 1–15. Multiple Classifier Systems.

[Esuli et al., 2008] Esuli, A., Fagni, T., and Sebastiani, F. (2008). Boosting multi-label hierarchical text categorization. Inf. Retr., 11(4):287–313.

[Fall et al., 2003] Fall, C. J., Törcsvári, A., Benzineb, K., and Karetka, G. (2003). Automated categorization in the international patent classification. SIGIR Forum, 37(1):10–25.

[Forman, 2003] Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. J. Mach. Learn. Res., 3:1289–1305.

[Grove et al., 2001] Grove, A. J., Littlestone, N., and Schuurmans, D. (2001). General convergence results for linear discriminant updates. Mach. Learn., 43(3):173–210.

[Hearst, 1998] Hearst, M. A. (1998). Support vector machines. IEEE Intelligent Systems, 13(4):18–28.

[Joachims, 1998] Joachims, T. (1998). Text categorization with support vector machines: learning with many relevant features. In Nédellec, C. and Rouveirol, C., editors, Proceedings of ECML-98, 10th European Conference on Machine Learning, number 1398, pages 137–142, Chemnitz, DE. Springer Verlag, Heidelberg, DE.

[Joachims, 2002] Joachims, T. (2002). Learning to Classify Text using Support Vector Machines. Kluwer.

[Koster and Beney, 2007] Koster, C. and Beney, J. (2007). On the importance of parameter tuning in text categorization. Perspectives of Systems Informatics, pages 270–283.

[Koster and Seutter, 2003] Koster, C. H. A. and Seutter, M. (2003). Taming wild phrases. In Proceedings 25th European Conference on IR Research (ECIR 2003), Springer LNCS, pages 161–176. Springer.

[Koster et al., 2003] Koster, C. H. A., Seutter, M., and Beney, J. (2003). Multi-classification of patent applications with winnow.

[Koster, 2009] Koster, K. (2009). Linguistic classification system: User manual version 2.5. Technical report, Raboud University of Nijmegen.

[Krier and Zaccà, 2002] Krier, M. and Zaccà, F. (2002). Automatic categorization applications at the european patent office. World Patent Information, 24:187–196.

[Larkey, 1999] Larkey, L. S. (1999). A patent search and classification system. In Fox, E. A. and Rowe, N., editors, Proceedings of DL-99, 4th ACM Conference on Digital Libraries, pages 179–187, Berkeley, US. ACM Press, New York, US.

[Littlestone, 1988] Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. Mach. Learn., 2(4):285–318.

[Mase et al., 2005] Mase, H., Matsubayashi, T., Ogawa, Y., Iwayama, M., and Oshio, T. (2005). Proposal of two-stage patent retrieval method considering the claim structure. ACM Transactions on Asian Language Information Processing (TALIP), 4(2).

[Morik et al., 1999] Morik, K., Brockhausen, P., and Joachims, T. (1999). Combining statistical learning with a knowledge-based approach - a case study in intensive care monitoring.

[Pflugfelder, 2009] Pflugfelder, B. (2009). Internal report on patent categorization. Matrixware private communication.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perception: a probabilistic model for information storage and organization in the brain. Psychological Review, 65(6):386–408.

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

[Tzhang et al., 2002] Tzhang, T. Z., Zhang, T., Damerau, F., Johnson, D., Hammerton, J., Osborne, M., Armstrong, S., and Daelemans, W. (2002). Journal of machine learning research 2 (2002) 615-637 submitted 9/01; published 3/02 text chunking based on a generalization of winnow. Journal of Machine Learning Research, 2:615–637.

[Vapnik, 1999] Vapnik, V. N. (1999). The Nature of Statistical Learning Theory (Information Science and Statistics). Springer.

[wei Hsu et al., 2003] wei Hsu, C., chung Chang, C., and jen Lin, C. (2003). A practical guide to support vector classification chih-wei hsu, chih-chung chang, and chih-jen lin. Technical report.

[WIPO, 2009a] WIPO (2009a). Guide to the ipc. http://www.wipo.int/classifications/ipc/en/guide/guide_ipc_2009.pdf.

[WIPO, 2009b] WIPO (2009b). International patent classification (ipc). http://www.wipo.int/classifications/ipc/en/.

[WIPO, 2009c] WIPO (2009c). International patent classification (ipc), edition 20090101, transformations directory, statistics. http://www.wipo.int/classifications/ipc/en/ITsupport/Version20090101/transformations/stats.html.

[WIPO, 2009d] WIPO (2009d). Literature survey: Issues to be considered in the automatic classification of patents. http://www.wipo.int/ipc/itos4ipc/ITSupport_and_download_area/Documentation/wipo-categorizationsurvey.pdf.

[Yang and Liu, 1999] Yang, Y. and Liu, X. (1999). A re-examination of text categorization methods. In 22nd Annual International SIGIR, pages 42–49, Berkley.

[Yang and Pedersen, 1997] Yang, Y. and Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. Proceedings of the Fourteenth International Conference on Machine Learning, 97.

[Zhang, 2000] Zhang, T. (2000). Large margin winnow methods for text categorization.

APPENDIX A

Parameter Settings

[Basic classifier settings]

- `FileDirectory` is the path where the text files can be found
- `DataDirectory` is the path in which all temporary data is stored
- `PartsList` is a file in which each line consists of a document name and its respective classes
- `OutputLocation` is the path where the results are stored
- `Algorithm` sets the learning algorithm (Winnow or SVM)
- `TrainMethod` defines whether the training is Flat or Hierarchical (*default* Flat)

[Document preprocessing]

- `MinDocTerms` minimum number of terms in a document for it to be considered suitable for training (*default* 1)
- `MinClassSize` minimum number of documents in a class for it to be considered suitable for training (*default* 10)
- `TermSel` sets the term selection algorithm to SimpleChiSquare or InfoGain (*default* SimpleChiSquare)
- `Strength` sets algorithm for strength calculation (*default* Ltc)
- `Normalize` sets normalization method (*default* Vector)

[Global term selection]

- `GlobalMinQuality` sets the minimum Q-value for global noise elimination (*default* 0)
- `GlobalMaxQuality` sets the maximum Q-value for global noise elimination (*default* 100)
- `GlobalTermFreq` sets the minimum number of occurrences of individual terms for global TF selection (*default* 3)
- `GlobalDocFreq` sets the minimum number of documents that an individual term should be in for global DF selection (*default* 3)

[Local term selection]

- `LocalMinQuality` sets the minimum Q-value for local noise elimination (*default* 1)

- `LocalMaxQuality` sets the maximum Q-value for local noise elimination (*default* 100)
- `MaxTerms Local` sets the maximum number of documents in a class for it to be considered suitable for training (*default* 2147483647)

[Class assignment]

- `MinRanks` minimum number of classifications for each document (testing) (*default* 1)
- `MaxRanks` maximum number of classifications for each document (testing) (*default* 4)

[Miscellaneous]

- `NumCpus` specify number of parallel threads for training (*default* 10)
- `SplitRatio` set train/test split ratio (*default* 80% train 20% test)

APPENDIX B

Top 10 subclasses

Sub class	Examples	Description based on IPC
G06F	33573	ELECTRIC DIGITAL DATA PROCESSING (computers in which a part of the computation is effected hydraulically or pneumatically G06D optically G06E; computer systems based on specific computational models G06N; impedance networks using digital techniques H03H)
H01L	31357	SEMICONDUCTOR DEVICES; ELECTRIC SOLID STATE DEVICES NOT OTHERWISE PROVIDED FOR (conveying systems for semiconductorwafers B65G 49/07; use of semiconductor devices for measuring G01; details of scanning-probe apparatus, in general G12B 21/00; resistors in general H01C; magnets, inductors, transformers H01F; capacitors in general H01G; electrolytic devices H01G 9/00; batteries, accumulators H01M; waveguides, resonators, or lines of the waveguide type H01P;line connectors, current collectors H01R; stimulated-emission devices H01S; electromechanical resonators H03H; loudspeakers, microphones, gramophone pick-ups or like acoustic electromechanical transducers H04R; electric light sources in general H05B; printed circuits,hybrid circuits, casings or constructional details of electrical apparatus, manufacture of assemblages of electrical components H05K; use of semiconductor devices in circuits having a particular application, see the subclass for the application)
H04N	28649	PICTORIAL COMMUNICATION, e.g. TELEVISION (measuring, testing G01; systems for autographic writing, e.g. writing telegraphy, which involve following an outline G08; information storage based on relative movement between record carrier and transducer G11B; coding, decoding or code conversion, in general H03M; broadcast distribution or the recording of use made thereof H04H)
A61K	28165	PREPARATIONS FOR MEDICAL, DENTAL, OR TOILET PURPOSES (devices or methods specially adapted for bringing pharmaceutical products into particular physical or administering forms A61J 3/00; chemical aspects of, or use of materials for deodorisation of air, for disinfection or sterilisation, or for bandages, dressings, absorbent pads or surgical articles A61L; soap compositions C11D)

Sub class	Examples	Description based on IPC
H04L	22482	TRANSMISSION OF DIGITAL INFORMATION, e.g. TELEGRAPHIC COMMUNICATION (typewriters B41J; order telegraphs, fire or police telegraphs G08B; visual telegraphy G08B, G08C; teleautographic systems G08C; ciphering or deciphering apparatus per se G09C; coding, decoding or code conversion, in general H03M; arrangements common to telegraphic and telephonic communication H04M; selecting H04Q; wireless communication networks H04W)
G11B	18832	INFORMATION STORAGE BASED ON RELATIVE MOVEMENT BETWEEN RECORD CARRIER AND TRANSDUCER (recording measured values in a way that does not require playback through a transducer G01D 9/00; recording or playback apparatus using mechanically marked tape, e.g. punched paper tape, or using unit records,e.g. punched or magnetically marked cards G06K; transferring data from one type of record carrier to another G06K 1/18; circuits for coupling output of reproducer to radio receiver H04B 1/20; gramophone pick-ups or like acoustic electromechanical transducers or circuits therefor H04R)
G01N	17145	INVESTIGATING OR ANALYSING MATERIALS BY DETERMINING THEIR CHEMICAL OR PHYSICAL PROPERTIES (separating components of materials in general B01D, B01J, B03, B07; apparatus fully provided for in a single other subclass,see the relevant subclass, e.g. B01L; measuring or testing processes other than immunoassay, involving enzymes or micro-organisms C12M, C12Q; investigation of foundation soil in situ E02D 1/00; monitoring or diagnostic devices for exhaust-gas treatment apparatus F01N 11/00; sensing humidity changes for compensating measurements of other variables or for compensating readings of instruments for variations in humidity, see G01D or the relevant subclass for the variable measured; testing or determining the properties of structures G01M; measuring or investigating electric or magnetic properties of materials G01R; systems in general for determining distance, velocity or presence by use of propagation effects, e.g. Doppler effect, propagation time, of reflected or reradiated radio waves, analogous arrangements using other waves G01S; determining sensitivity, graininess, or density of photographic materials G03C 5/02; testing component parts of nuclear reactors G21C 17/00)

Sub class	Examples	Description based on IPC
A61P	16396	THERAPEUTIC ACTIVITY OF CHEMICAL COMPOUNDS OR MEDICINAL PREPARATIONS
C07D	16263	HETEROCYCLIC COMPOUNDS
G02B	14709	OPTICAL ELEMENTS, SYSTEMS, OR APPARATUS (G02F takes precedence; optical elements specially adapted for use in lighting devices or systems thereof F21V 1/00-F21V 13/00; measuring-instruments, see the relevant subclass of class G01, e.g. optical rangefinders G01C; testing of optical elements, systems, or apparatus G01M 11/00; spectacles G02C; apparatus or arrangements for taking photographs or for projecting or viewing them G03B; sound lenses G10K 11/30; electron and ion "optics" H01J; X-ray "optics" H01J, H05G 1/00; optical elements structurally combined with electric discharge tubes H01J 5/16, H01J 29/89, H01J 37/22; microwave "optics" H01Q; combination of optical elements with television receivers H04N 5/72; optical systems or arrangements in colour television systems H04N 9/00; heating arrangements specially adapted for transparent or reflecting areas H05B 3/84)