Comparative Analysis of Balanced Winnow and SVM in Large Scale Patent Categorization

Katrien Beuls
Vrije Universiteit Brussel
Pleinlaan 2
1050 Brussels, Belgium
katrien@arti.vub.ac.be

Bernhard Pflugfelder
Matrixware
Operngasse 20b
1040 Vienna, Austria
b.pflugfelder
@matrixware.com

Allan Hanbury
Information Retrieval Facility
Operngasse 20b
1040 Vienna, Austria
a.hanbury@ir-facility.org

ABSTRACT

This study investigates the effect of training different categorization algorithms on a corpus that is significantly larger than those reported in experiments in the literature. By means of machine learning techniques, a collection of 1.2 million patent applications is used to build a classifier that is able to classify documents with varyingly large feature spaces into the International Classification System (IPC) at Subclass level. The two algorithms that are compared are Balanced Winnow and Support Vector Machines (SVMs). Contrary to SVM, Balanced Winnow is frequently applied in today's patent categorization systems. Results show that SVM outperforms Winnow considerably on all four document representations that were tested. While Winnow results on the smallest sub-corpus do not necessarily hold for the full corpus, SVM results are more robust: they show smaller fluctuations in accuracy when smaller or larger feature spaces are used. The parameter tuning that was carried out for both algorithms confirms this result. Although it is necessary to tune SVM experiments to optimize either recall or precision - whereas this can be combined when Winnow is used - effective parameter settings obtained on a small corpus can be used for training a larger corpus.

Categories and Subject Descriptors

H.3.3 [**Information storage and retrieval**]: Information search and retrieval—*clustering, information filtering, retrieval models, search process, selection process*

General Terms

Algorithms, Performance, Experimentation

Keywords

Patent Classification, Intellectual Property, IPC taxonomy

1. INTRODUCTION

The Intellectual Property domain with its more than 70 million patents is characterized by the continuing need to make critical scientific and technical decisions in the face of the exponential growth in the quantity of potentially relevant information. A fundamental need in the Intellectual Property domain is the availability of sophisticated and trustworthy systems for automatic categorization of patent applications.

Various systems have been developed to provide the automatic categorization of patent documents. In general, patent categorization requires the assignment of test documents into a very large taxonomy. In addition, predefined categories are often ambiguous and difficult to distinguish. The processing of patent documents introduces a large number of distinctive terms that are often very technical due to the domain-dependent vocabulary and trigger a low term occurrence frequency over the entire patent collection. The total number of extracted terms to build class profiles is therefore huge. This negatively affects both the categorization quality in terms of Precision and Recall as well as the efficiency of state-of-the-art learning methods.

Text categorization is often defined as the task of assigning a Boolean value to each pair $< d_j, c_i > \in D \times C$, where D is a domain of documents and $C = \{c_1, \ldots, c_{|C|}\}$ is a set of pre-defined categories [13]. Patent document categorization differs from prototypical text categorization on at least three more grounds related to the set of predefined categories. *First*, this set is characterized by a large imbalance as the number of inventions varies in different parts of the taxonomy. *Second*, because the target of the categorization problem is a very small subset of the huge feature space, the scalability of training and testing can become problematic. And *third*, patents are mostly assigned to multiple categories, which means we are dealing with a multi-categorization task.

The Winnow algorithm belongs to the family of on-line mistake-driven learning algorithms such as the Perceptron. It differs from the latter as the algorithm does not learn the linear separation between examples assigned to different categories additively but rather multiplicatively. Most current patent categorization systems, e.g. IPCCAT[1], are based on a variant of the Winnow algorithm, Balanced Winnow.

This paper contrasts Balanced Winnow with an SVM (Support Vector Machine) learner, known to be more robust considering the unbalancedness in class distributions. The computation of the discriminative function of an SVM happens by solving an optimization problem based on a maximum margin separation. SVMs are frequently applied in applications based on images, music, etc. In text categorization, multiple publications [6, 7] state that SVMs outperform other machine learning methods in terms of accuracy but with the disadvantage of needing far more calculation time. The evaluation that is presented in this paper incorporates

[1]Automatic categorization system of the World Intellectual Property Organization (WIPO).

considerably more train and test documents than was the case in previous studies.

Extensive comparisons between algorithms have been published in the literature [13, 4, 15]. This study uses a corpus of 1.2 million patent documents extracted from the Alexandria patent repository of Matrixware[2] and sets up categorization experiments on the basis of the International Patent Classification (IPC) taxonomy on the sub class level (639 classes). The use of such a large corpus makes this study closer to solving the problem such as it exists in the real world. Next to comparing two different learning algorithms, Balanced Winnow and learning with SVMs, the effect of training different patent representations is investigated.

The Linguistic Classification System[3] (LCS), developed in the course of the DORO and PEKING Esprit Projects, was used as a workbench to carry out the experiments. The LCS represents a full text categorization framework with the possibility to adjust various components for optimal performance according to the categorization problem.

2. RELATED WORK

One of the first patent categorization systems was developed by Larkey [9], in which an automatic search and categorization tool for U.S. patent documents according to the US Patent classification (USPC) scheme. The service categorizes patent documents at the subclass level of the USPC by applying the k-nearest neighbor algorithm on a pre-selected set of patent documents retrieved by the system's search component. As the subclasses of those pre-selected patents are known to the system, the k-nearest neighbor algorithm computes a subclass ranking based on the document similarity of the unseen patent and the pre-selection. Unfortunately, an evaluation of the categorization quality was not made public. In 2002, the European Patent Office (EPO) published a comparative analysis of categorization systems in terms of pre-categorization [16]. All compared categorization systems perform pre-categorization on the basis of 44 directorates, 549 teams and 624 subclasses, i.e. three different category structures, with considering only the given prediction. The results show precision scores of 72 %, 57 % and 61 % respectively. The CLAIMS project [2] has participated in a public call of World Intellectual Property Office (WIPO). Again, it does pre-categorization by categorizing patents into the first four hierarchy levels of IPC. While Larkey deployed the k-nearest neighbor algorithm, CLAIMS made an effort to utilize the more accurate Balanced Winnow algorithm using a restricted document representation comprising only the first 600 different terms of every document. In fact, CLAIMS won the public call and the patent categorizer IPCCAT has been directly derived from CLAIMS and is still in use. In [8], another patent categorization system using Balanced Winnow is introduced.

In recent years, international IR evaluation campaigns started patent retrieval and mining tasks. The Japanese evaluation project NTCIR for information retrieval systems was the first IR evaluation campaign which included a separate task on patent mining. A patent categorization subtask [5] was firstly introduced at the NTCIR-5 workshop and categorization (sub) tasks were organised in the following NTCIR campaigns. Most recently, the NTCIR-7 campaign [11] investigated the categorization of research publications into IPC based on training with patent data.

[2] `http://www.matrixware.com`
[3] `http://www.cs.ru.nl/peking/`

3. CLASSIFICATION METHODS

The categorization of patent documents into the IPC comprises a large number of categories, depending on the level of the IPC at which the categorization will be done. As the standard implementations of Balanced Winnow and SVM only work on binary class problems, ensemble learning is additionally applied by transforming the multi-category problem into an ensemble of binary (one-vs.-rest) problems. For every IPC subclass a separate binary SVM or Balanced Winnow classifier was trained. The categorization is done by selecting the prediction that has the highest confidence among the predictions of all binary classifiers.

3.1 Winnow

The implementation of the algorithm reported in this paper is Balanced Winnow [10, 3]. 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 the difference between the positive and negative weights, which 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,while the negative part of the weight is demoted. When a mistake occurs on a negative example the positive part of the weight is demoted and the negative part is promoted. 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^-.

3.2 SVM

In the text categorization process the training data can be separated by at least one hyperplane h'. This presupposes 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 $t_i((\mathbf{w}^T \times \mathbf{x}_n) + b^T) > 0$ 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[4] [14] only the hyperplane that maximizes the margin δ separating positive and negative examples is selected. The small set of training examples that determines the best surface is called the support vectors. They have a distance of exactly δ to the hyperplane.

One problem with the implementation of SVMs is that training fails when the training examples are not linearly separable. Even though this is almost never the case in text

[4] More information on how SVMs implement structural risk minimization can be found in [7]

categorization, flawless training can result in overfitting of the data and therefore affect the testing accuracy. To avoid this, soft-margin SVM [1] is used. 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. In the SVM implementation SVM Light, the parameter C controls this trade-off between training error and margin. $C = 0$ refers to a hard margin learner, while $C > 0$ represents soft-margin SVM.

4. EXPERIMENTAL SETUP

This section describes the patent data corpus used for the categorization experiments, including some important features of patents. The hardware environment and the setup of the categorization process is also summarized.

4.1 Features of Patent Documents

The content of a patent is governed by legal agreements and is therefore semi-structured. In a European patent document, for instance, the bibliographic data field contains information such as technical details (e.g. the invention title, citations, etc.) 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. A patent's 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 of a patent document states the prior art and the novelty of the patent application and often contains standard expressions. A final field that is part of a patent document is the patent's legal status. This status indicates whether the patent is still an application or whether it is an already granted patent.

4.2 International Patent Classification

Patent documents receive specific codes that refer to the class they belong to. The International Patent Classification[5] (IPC) provides a hierarchical system of language independent symbols for the categorization of patents and utility models according to the different areas of technology to which they pertain. In the past, the IPC has been updated every five years and it is currently in the IPC-2009 edition. Each IPC code is a unique combination of the hierarchical structure codes of the patent identity. The three levels in the patent hierarchy that are used in this paper, shown with the number of classes in parentheses, are Section (8), Class (121), and Subclass (631). The official IPC hierarchy contains two deeper levels: Main Group and Sub Group. Table 1 shows a portion of the IPC specification at the start of section G.

4.3 Description of the Corpus

4.3.1 General Statistics

The complete corpus contains 1 270 185 patent documents that are split up into two sub collections: EP (563 248) and PCT (706 937). The patents were extracted from the patent document repository Alexandria Patent created by Matrixware . The patent document archive is provided in a common XML format, created by merging data from various sources. The collection contains EP and PCT patent applications in the period 01/01/1985 - 31/12/2006 having the sections title, abstract and description in English. The patent corpus covers 621 different IPC subclasses, covering 94 % of the possible 639 subclasses according to IPC AL[6]. The focus of research and development varies over the last 20 years and more, resulting in highly varying numbers of patent applications filled for a particular subclasses and year. Consequently, the population of the subclasses with examples is unbalanced in real world and in the data corpus that is illustrated in Figure 1. For instance, the 3 most frequent subclasses in the data corpus are given in Table 2.

Table 1: Sample portion of the IPC taxonomy at the start of Section G

Category	Symbol	Title
Section	G	PHYSICS
Class	G06	COMPUTING; CALCULATING; COUNTING
Subclass	G06N	COMPUTER SYSTEMS BASED ON SPECIFIC COMPUTATIONAL MODELS
Main group	G06N 5/00	Computer systems utilizing knowledge based models
Sub group	G06N 5/02	Knowledge representation

Table 2: The 3 most frequent IPC subclasses in the data corpus

Sub class	Description	Examples
A61K	Preparations for medical, dental, or toilet purposes	121955
G06F	Electric digital data processing	76575
A61P	Therapeutic activity of chemical compounds or medicinal preparations	65655

4.3.2 Sub Collections

Sub collections were created for better indicating both the categorization quality and scalability of the learners on smaller data sets of respectively 100k, 200k, 400k and 500k. A statistics of all sub collections, including coverage of IPC classes and subclasses, is listed in Table 3. Note that the full data set contains double the amount of documents of the biggest sub collection. Train and test sets are built for each of sample collection (cf. 4.4)

Since the class distribution of a collection used for training influences the categorization quality, we defined an algorithm that takes the class distribution of the original corpus into account. Instead of selecting classes that are included into a sub collection randomly, the algorithm follows the following idea. *Firstly*, all classes are ordered based on the number of examples. *Secondly*, k sample sets are created. *Thirdly*, n classes are chosen out of every sample set and all

[5]`http://www.wipo.int/classifications/ipc/en/`

[6]IPC Advanced Level (AL)

Table 3: Sub collections statistics

Dataset	#Docs	#Classes	#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%)

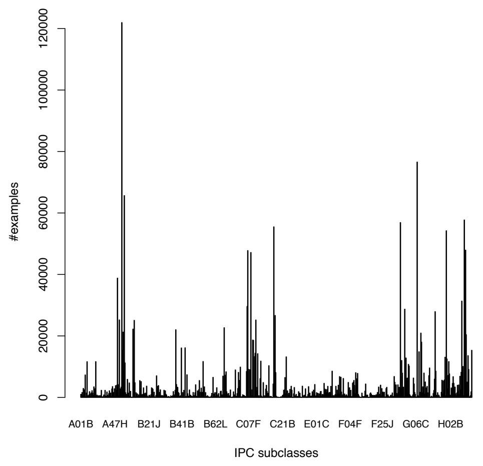


Figure 1: Class distribution of the whole data corpus containing 1.2 million patent applications

examples, which are assigned by those selected classes, are added to the sub collection.

We investigate four different document representations including different sections of the patent documents, shown in Table 4.

4.4 Test Environment and Settings

All the experiments were run on the LDC (Large Data Collider), an SGI Altix 4700 machine provided by the IRF. This server runs under SUSE Linux Enterprise Server 10 and has 40 Dual Core Itanium 2 (1.4 GHz) processing units as well as 320 GB memory. In contrast with the newest version of the LCS, the system that we used for our experiments did not apply parallelization in the training phase.

The experiments were executed using the LCS classification system, using the following setup:

Pre-processing: de-capitalization, removal of special characters such as braces. Term profiles are created by LCS's proprietary indexer. Class profiles containing the term distributions (per class) are generated for the term selection step.

Table 4: Document representations

tabs	title, abstract
itabs	inventor, title, abstract
tdesc	title, description
itabsdesc	inventor, title, abstract, description

Global term selection: document frequency (min=3), term frequency (min=3), noise reduction based on the Uncertainty measure introduced in [12].

Local term selection (class-specific): Simple Chi Square. We chose the LCS feature to automatically select the most adequate number of relevant terms for every class.

Term strength calculation: LTC algorithm [8], extension of TF-IDF.

Training method: Ensemble learning based on one-vs.-rest binary classifiers.

Classification algorithms: Balanced Winnow and SVM, using LCS's propriety implementation of Balanced Winnow and the SVM Light implementation developed by Thorsten Joachims[7].

Parameter settings for the Winnow experiments: A settings of $\alpha = 1.1$, $\beta = 0.9$, $\theta^+ = 1.1$, $\theta^- = 0.9$ and 4 iterations was chosen according to an evaluation carried out within the domain of patent categorization [8].

Parameter settings for the SVM experiments: Based on an evaluation regarding the optimization of F1 using a small subset (±50000 documents) of the corpus, we selected $C = 1.5$, $J = 1.0$ along with a linear kernel.

Class assignment: min/max number of classifications for each document (resp. 1 and 4)

Evaluation: Split ratio for train/test split: 80/20. The quality of the categorization is determined by Precision, Recall and F1 measure according to the definition in [8]. Both micro-averages and macro-averages are calculated to measure the quality over all classes.

5. EXPERIMENTAL RESULTS

This section presents the experimental results of comparing both learning algorithms SVM and Balanced Winnow on the basis of four document representations defined in Section 5.1. In Section 5.2, the investigation of the SVM parameters complexity (C) and cost factor (J) and the search for an optimal SVM parameter settings concludes this section.

5.1 Comparative Analysis

In Table 5, the micro-averages of Precision, Recall and F1 are listed based on the five sample collections (Table 3) and the four document representations (Table 4). Due to scalability limits, the experiment of SVM with *tdesc* and *itabsdesc* representations on the full data corpus could not be executed, while the corresponding experiment deploying Winnow succeeded. Macro-averages are not presented here due to the space limitations.

Although primary results confirmed the superiority of SVM training over Balanced Winnow with regard to evaluation scores, a detailed analysis of both algorithms should take a complete range of different aspects into account.

First, there is the size of the feature space. On the one hand, the dimensionality of the feature space depends on the document representation that is used in the evaluation. As shown in Table 5, the difference in performance between the

[7]`http://svmlight.joachims.org`

Table 5: Micro-averaged Precision (P), Recall (R) and F1 results for SVM and Balanced Winnow

Doc.Rep.	**100k**			**200k**			**400k**			**500k**			**1200k**		
	SVM														
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
tabs	87,97	71,58	78,93	84,94	47,42	58,16	82,38	51,09	63,07	81,01	46,79	59,32	79,21	45,31	57,64
itabs	88,18	70,80	78,54	84,82	48,48	59,00	83,18	52,59	64,44	82,13	48,27	60,81	**80,08**	**47,57**	**59,69**
itabsdesc	**90,40**	79,86	**84,81**	**90,66**	58,54	68,04	**84,47**	**62,26**	**71,68**	**83,00**	**57,48**	**67,92**	x	x	x
tdesc	90,31	**79,90**	84,79	90,51	**76,79**	**83,09**	84,31	61,84	71,35	82,32	56,86	67,26	x	x	x
Doc.Rep.	**Balanced Winnow**														
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
tabs	72,75	77,87	75,23	70.29	74.49	72.33	59,20	57,77	58,48	58,14	53,83	55,90	56,43	52,81	54,56
itabs	73,67	78,57	76,04	71,84	75,07	73,42	62,03	59,68	60,84	60,37	54,85	57,48	59,64	53,94	56,65
itabsdesc	**80,86**	84,30	82,54	**79,66**	**81,99**	**80,81**	**70,30**	**67,27**	**68,75**	**68,22**	**61,57**	**64,72**	**66,37**	**59,59**	**62,80**
tdesc	80,71	**84,67**	**82,64**	79,48	81,82	80,63	67,74	61,46	64,45	67,74	61,46	64,45	65,63	59,49	62,41

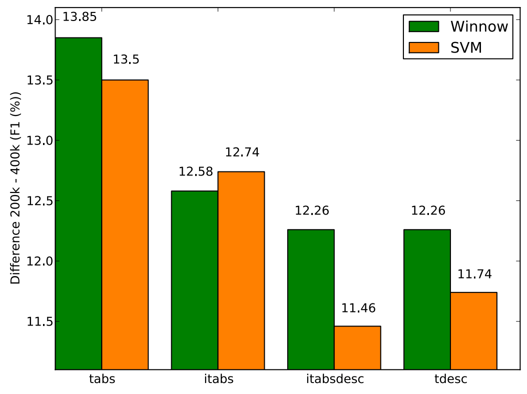


Figure 2: Decrease in F1 scores between 200k and 400k

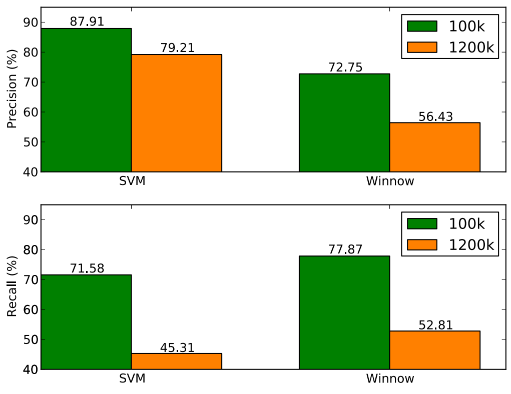


Figure 3: Precision/Recall results on 100k and full corpus for SVM and Winnow (*tabs*)

algorithms becomes smaller with respect to the term space, ranging from the smallest document representation *tabs* to the largest *itabsdesc* and *tdesc* representations. This means that the gap in terms of achieved Precision and F1 between SVM and Winnow narrows when longer documents are used.

On the other hand, the SVM optimization problem seems to be affected more by an increase in the amount of documents that is used. In our experiments, no single documents were added to the corpus but complete classes. The biggest increase in classes occurs when going from the 200k to the 400k collection (70 classes in 200k, 400 in 400k). Different reactions are triggered dependent on the learner. When the feature space is smallest (*tabs*), Winnow is slightly more affected by the increase of classes than SVM. Running from smallest to largest feature space (*itabsdesc*), the SVM bars in Figure 2 decrease more steeply than the Winnow bars. 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 in combination with the rapid increase in classes. The difference between 200k and 400k across the learning algorithms is maximized at this point (*itabsdesc*).

Second, both Precision and Recall should be considered when drawing a comparison between both learners. The bar charts in Figure 3 show that both SVM and Winnow expose a less steep decrease in Precision than Recall when moving from the smallest to largest dataset. Whereas Recall drops by 26.27 % for SVM and 25.06 % for Winnow, 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.

When looking at the micro-averages in greater detail, there are further interesting trends visible. The training results for Winnow show that over the collection sizes, Recall is higher than Precision for 100k whereas the opposite scenario is found for the 1200k corpus. This indicates that Winnow Recall decreases more rapidly than Winnow Precision when more classes are used for training. This is not the case for SVM, where Precision stays always higher than Recall. What is more, SVM Precision scores proved to be so stable that the difference between *tabs* and *itabsdesc* for the 400k collection results in only 2 %, while the exactly same settings yield a difference of 11 % for Balanced Winnow.

Figure 4 illustrates the empirical distributions of category-based Precision and F1 for two different document representations using box plots. 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. Contrarily, the category-based F1 results vary much stronger in terms of the SVM classifier compared to the Winnow classifier. This gap between Precision and Recall results

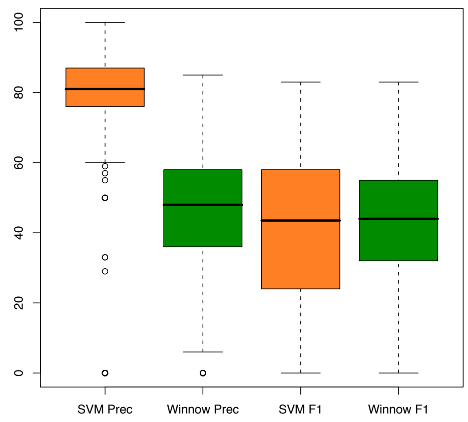


Figure 4: Extreme results of SVM Precision: Boxplots of category-based Precision/F1 results on *itabs* and the 1200k dataset

among the categories suggests the need of parameter tuning in order to balance Precision and Recall.

5.2 Tuning of SVM parameters

So far the experimental results suggest superior performance of the SVM in terms of Precision compared to Winnow. However, the large differences between Precision and Recall for both learning algorithms imply the need of parameter tuning. In fact, the used SVM parameters caused Precision to remain high, while Recall dropped under 50 %. Depending on the goal of the experiment, such a low level of Recall may not be ideal. We therefore target parameter tuning in order to find SVM parameters that optimize either Precision or Recall.

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 categorization even more.

The parameter tuning was carried out with the smallest collection (100k) to speed up the process.

5.2.1 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 maximize this difference (x), the value is squared and inverted. This is done for every training example that has a counter example in its neighborhood, i.e. only for support vectors. In general, C can be any floating point number bigger than 0.

Different values of C ranging between 0 and 50 were tested. Above 2.0 the results do not add much more to the parameter tuning (and are therefore not shown): 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 $C = 50$).

A more detailed parameter tuning between 0.1 and 2.0 with steps of 0.1 shows that 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 $C = 2.0$ (72.22 %). The C value used in the baseline experiments (1.5) is the point where the F1 value reaches its maximum.

5.2.2 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 $C_+ = J \times C$, while the misclassification cost of a negative example C_- remains unaltered [6]. By default $J = 1.0$.

Figure 5 shows the results of a tuning $J \in [1, 10]$ with step sizes of 1. The highest recall value is found at $J = 8$. Although the goal of this parameter tuning is to get Recall possibly at its highest level, Precision, and therefore F1, value should in the best cases not drop significantly. Therefore, $J = 5$ seems a reasonable choice. In all these experiments, C was kept at its default value (1.5).

5.2.3 Grid search

In order to find SVM parameters that can be used to optimize Precision and Recall, a grid search of C/J pairs was carried out. Similar to the previous tuning experiments, we define $C \in [0, 2]$ and $J \in [1, 8]$. Figure 6 shows the results of this grid search as heat maps. It can be seen that the extremes in Precision and Recall are almost reversed. Whereas the highest recall values are obtained when J is largest, Precision peaks when J is smallest. This indicates that a separate tuning for Precision and Recall is necessary when SVM is used. Contrarily, a similar investigation of

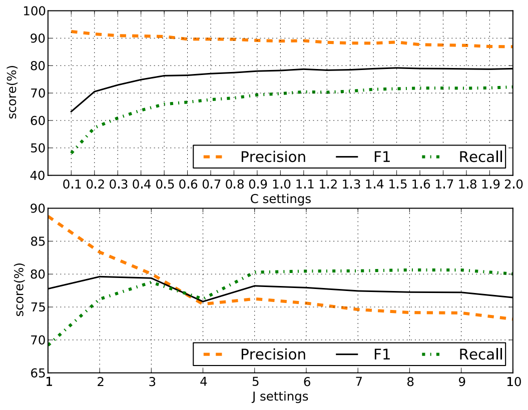


Figure 5: SVM tuning of the complexity parameter C (top) and the cost factor J (bottom) on 100k corpus

Table 6: Optimal Precision/Recall settings for SVM and Winnow on *itabsdesc* 100k

Settings	Precision	Recall	F1
$C = 0.25; J = 10$	68.54 %	85.21 %	75.97 %
$C = 0.05; J = 12$	62.46 %	85.55 %	72.20 %
$\alpha = 1.05; \beta = 0.9$	71.43 %	78.73 %	74.90 %
$\alpha = 1.1; \beta = 0.9$	72.23 %	78.00 %	75.00 %

Table 7: Maximum Precision/Recall and default SVM settings on *itabsdesc* 500k

Settings	Precision	Recall	F1
$C = 0.10; J = 1.0$	89.34 %	30.97 %	46.00 %
$C = 0.25; J = 10.0$	62.92 %	73.50 %	66.14 %
$C = 1.50; J = 1.0$	83.00 %	57.48 %	67.92 %

Balanced Winnow tuning[8] identifies one set of parameters that optimize both Precision and Recall. If the C parameter is taken into account, the plots show that the smallest C values exhibit both the highest and lowest Precision and Recall results. Precision peaks at $C = 0.1$ and $J = 1$, whereas Recall reaches its maximum value at $C = 0.4$ and $J = 8$.

The highest Recall scores are situated on the open-ended side of the heat plot. In order to have a quick idea whether we have reached the upper limit in terms of Recall scores, two more experiments were conducted that were expected to improve Recall even more (and therefore worsen Precision). The results, which are summarized in Table 6, indicate that using even higher J values does not improve Recall significantly anymore compared to Figure 6(b).

5.2.4 Robustness

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 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. The results are summarized in Table 7.

The experiment that was run with the optimal Precision settings yielded Precision 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 was 3.6 %. This shows that the optimal Precision settings hold for a bigger corpus as well. The maximum recall settings yielded an increase in Recall of 16.02 % on top of the baseline. A last point to note is the stability of the F1 value, losing only just over 1 % after the tuning has taken place.

6. DISCUSSION AND CONCLUSIONS

The Support Vector Machine is a popular machine learning algorithm, which achieves excellent performance in many categorization applications, including text categorization. Although patent categorization is a sub problem of text categorization, a direct application of SVM must be evaluated due to specific aspects of patents and patent classification systems. This study investigates SVM in patent categorization based on the IPC and compares SVM with Balanced Winnow, which is frequently applied in current patent categorization systems such as the IPCCAT.

SVM outperforms Balanced Winnow in terms of Precision on all sample collections and document representations (see Table 5). The difference in Precision of the two learning algorithms becomes smaller with growing number of terms. In other words, Balanced Winnow benefits more from the increase in the number of terms introduced by the longer document representations *itabsdesc* and *tdesc*. In fact, SVM still outperforms Balanced Winnow in terms of Precision on longer document representations. On the other hand, in the experiments the SVM training does not scale to longer document representations without algorithmic parallelization and distribution, while training with Balanced Winnow still succeeds.

In contrast to the Precision results, SVM training delivered lower Recall compared to Balanced Winnow. The results show large differences between Precision and Recall for each of the SVM experiments. This suggests that the SVM parameters that were applied in the experiments are not optimal and parameter tuning is needed in order to increase both Precision and Recall.

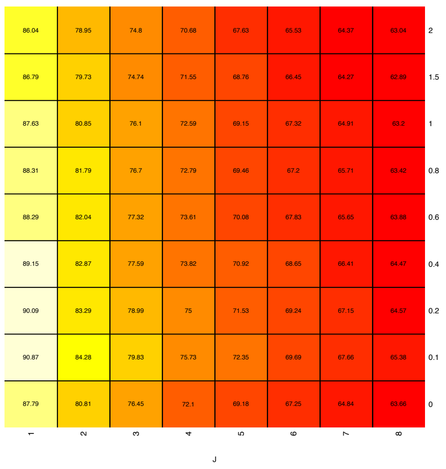
Since the used sample collections do not only grow in terms of document number, but, also in the number of classes (max. 600), the Precision values in Table 5 show that SVM results remain more stable than Winnow results as the corpus size increases (100k $\rightarrow$ 1200k). Another important conclusion is that SVM is more robust regarding the unbalanced class distribution depicted in Figure 1. The Precision values delivered by Winnow are more affected by the imbalance in the training collection than SVM, which is shown in Figure 4.

Depending on either optimizing Precision or Recall, different SVM parameter settings are determined in this study. Using a linear kernel with $C = 0.1$ and $J = 1.0$ achieves highest Precision, while optimal Recall is found with $C = 0.25$ and $J = 10.0$. Table 7 lists those settings in combination with the achieved Precision, Recall and F1 values. On a corpus of 500k patent documents, Precision tuning exceeds the baseline experiments by 6.34%, Recall tuning even by 16.32%.

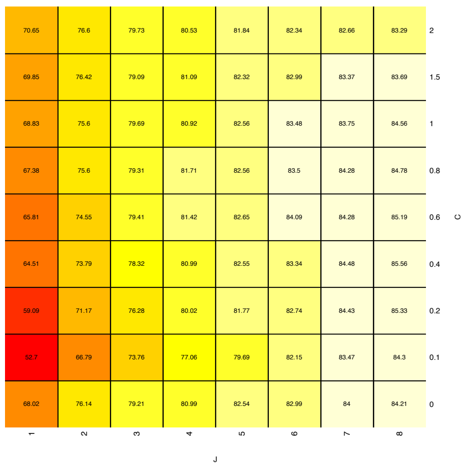
Due to its robustness, SVM learning can be tuned on a small corpus and does not take up too much time if a careful grid search is applied. Keeping the cost parameter (J) low improves Precision, whereas a bigger J maximizes Recall. This finding can be extended to tuning other types of text corpora. A low cost parameter means allowing fewer mistakes (Precision) but therefore reduces the number of positive examples being retrieved (Recall). The complexity parameter (C) can be chosen in the interval $[0, 2]$.

Although detailed results on Balanced Winnow tuning could not be included due to the space limitations, it was also a part of the study. Low α-values of 1.01 and 1.02 along with 5 to 8 learning iterations yield optimal Precision/Recall on *itabsdesc* 500k, while α-values of 1.1 and 1.05 yield optimal Precision/Recall on *itabsdesc* 100k. Comparing with SVM two important differences can be observed. Firstly, Winnow tuning is not robust on different training collections. Secondly, the gap between Precision and Recall issignificantly smaller using the same parameter setting.

[8]Optimal Precision/Recall settings are given in Table 6. Due to space limitations, more detailed tuning results could not be included.



(a) Precision



(b) Recall

Figure 6: Grid Search of SVM parameters C and J. More reddish (darker) colors denote low Precision/Recall, while brighter colors indicate high Precision/Recall.

To summarize the most interesting conclusions, all LCS Winnow experiments reveal moderate to low micro-averaged Precision and Recall results as well as moderate robustness in case of an increasing corpus size. *itabsdesc* significantly outperforms all other document representations. Parameter tuning showed that same parameter setting improves both Precision (72.23 %) and Recall (78 %).

The SVM experiments, on the other hand, achieve high Precision over all sample collections with an exceptional balancing of the Precision over all sub classes. SVM is highly robust in terms of number of sub classes to be learned. *itabsdesc* significantly outperforms all other document representations, while parameter tuning is indispensable.

The next step in this study is the development of automatic tuning methods. Depending on the requirements of the user, specific Precision- or Recall-directed tuning can be realized. Another issue that deserves attention in future investigation is the use of semantic information in the creation of the index. The incorporation of such linguistic information is the next component that will be added to the LCS.

7. REFERENCES

[1] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

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

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

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

[5] M. Iwayama, A. Fujii, and N. Kando. Overview of classification subtask at NTCIR-5 patent retrieval task. In *Proceedings of the Fifth NTCIR Workshop*, Tokyo, Japan, 2005.

[6] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *ECML '98: Proceedings of the 10th European Conference on Machine Learning*, pages 137–142, London, UK, 1998. Springer-Verlag.

[7] T. Joachims. *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms.* Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[8] C. H. A. Koster, M. Seutter, and J. Beney. Multi-classification of patent applications with Winnow. In *In Proceedings PSI 2003*, volume 2890, pages 545–554, LNCS, 2003. Springer.

[9] L. S. Larkey. A patent search and classification system. In *Proceedings of DL-99, 4th ACM Conference on Digital Libraries*, pages 179–187. ACM Press, 1999.

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

[11] H. Nanba, A. Fujii, M. Iwayama, and T. Hashimoto. Overview of the patent mining task at the NTCIR-7 workshop. In *Proceedings of the Seventh NTCIR Workshop Meeting*, Tokyo, Japan, 2008.

[12] C. Peters and C. H. A. Koster. Uncertainty-based noise reduction and term selection in text categorization. In *Proceedings of the 24th BCS-IRSG European Colloquium on IR Research*, pages 248–267, London, UK, 2002. Springer-Verlag.

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

[14] V. N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[15] Y. Yang and X. Liu. A re-examination of text categorization methods. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49, New York, NY, USA, 1999. ACM.

[16] F. Zaccá and M. Krier. Automatic categorisation applications at the European Patent Office. *World Patent Information*, 24:187–196, 2002.