FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Feature Selection for Patent Classification based on Entropy

Diplomarbeit

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Computational Intelligence

eingereicht von

Thomas Kern
Matrikelnummer 0326577

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer: Privatdoz. Dr. Allan Hanbury
Mitwirkung: Projektass. M.A. Linda Andersson

Wien, 26.08.2011

(Unterschrift Verfasser) (Unterschrift Betreuer)

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.

Amstetten, August 2011

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschliesßlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Amstetten, August 2011

Thomas Kern

Great minds discuss ideas.
Average minds discuss events.
Small minds discuss people.

— Eleanor Roosevelt

ABSTRACT

In this thesis I develop three feature selection algorithms for automatic classification of patents based on the International Patent Classification (IPC) categories. All three of them help reduce the number of features, the length of the training and the resources needed for the training, while improving the classification performance. The number of features is reduced to 4%. Furthermore I deliver a comparison between several linear classifiers including a Support Vector Machine (SVM) and an L2-regularized linear classifier.

In 2010 Montemurro and Zanette proposed a method to identify important words in a text. On this method a feature selection metric was built to distinguish between valuable and negative features. Furthermore three new filter selection algorithms based on the new metric are proposed.

ZUSAMMENFASSUNG

In dieser Studie wird ein Framework für die Auto-Klassifizierung von Patenten entwickelt. Besonderes Augenmerk wird auf einen Vergleich von einigen linearen Klassifizierungsalgorithmen wie etwa SVM und einem L2-regulierten linearen Klassifizierungsalgorithmus gelegt. Weiters wird eine Feature-Selektion vorgenommen um die benötigten Resourcen während des Trainings zu minimieren, ohne dabei die Precision bzw. den Recall zu verschlechtern.

Im Jahr 2010 hat Montemurro und Zanette eine Methode vorgestellt, mit der man wichtige Wörter in einem Text erkennen kann. Auf dieser Methode aufbauend wurde eine Metrik und ein Algorithmus zur Feature-Selektion erstellt. Dieser Algorithmus kann die Anzahl der Features auf 4% reduzieren, ohne dabei die Klassifikationsperformance zu verringern.

ACKNOWLEDGMENTS

First of all I would like to thank my two supervisors **Dr. Allan Hanbury** and **M.A. Linda Andersson** for the opportunity they gave me to realize my diploma thesis at the IRF and TU Vienna and for the interesting interdisciplinary topic. I want to thank her for her outstanding support, answering all my questions on an almost daily basis and encouraging me to succeed. Moreover, the weekly meetings with Dr. Hanbury were inspiring and focusing my efforts on issues deserving more attention. Thanks for the professional and warm support.

In addition I want to thank all my university colleagues and friends, who have helped me a lot, not only during the thesis. I would like to emphasize my friends **Florian Haselmayer** for reading through the overlong experiments section and **Michael Grafl** for giving me feedback on a lot of my charts.

Also of course my beloved brother **Martin** and my amazing **parents**, who always supported and believed in me.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ACO Ant Colony System

CF Collection Frequency

DF Document Frequency

EPO European Patent Office

ERM Error Risk Minimization

FIBIS U.S Foreign Broadcast Information Service

FOA First Office Action

IDF Inverse Document Frequency

IG Information Gain

IPC International Patent Classification

IQR Interquartile Range

IRF Information Retrieval Facility

MPEP Manual of Patent Examining Procedure

MAREC MAtrixware REsearch Collection

MDC Medium Data Collider

MI Mutual Information

MMV Montemurro Metric Value

NLP Natural Language Processing

PU Production Units

PP Percentage Points

RBF Radial Basis Function

ROC Receiver Operating Characteristic

SNOW Sparse Network of Winnows

SRM Structural Risk Minimization

SVM Support Vector Machine

TF Term Frequency

TF-IDF TF-Inverse Document Frequency

UPR Utility, Plant, Reissue

USPTO United States Patent and Trademark Office

USA United States of America

WEBF Word-Entropy Based Filter

WEBF1 Word-Entropy Based Filter 1

WEBF2 Word-Entropy Based Filter 2

WEBSOL Word-Entropy Based Stop wOrd List

WIPO World Intellectual Property Organization

1 INTRODUCTION

A patent is an exclusive right granted for an invention, which is a product or a process that provides, in general, a new way of doing something, or offers a new technical solution to a problem. In order to be patentable, the invention must fulfill certain conditions.

— European Patent Office (EPO) [9]

problem

In 2010 the United States Patent and Trademark Office (USPTO) received approximately 550 000 patent applications. 244 000 were granted in the same year, the remaining patent applications were either rejected or are still pending. This is added onto the pile of already granted patent applications by the USPTO, which in December 2010 is over 8 000 000 strong. The backlog at the USPTO is constant at 720 000 documents over the last 24 months. This number is stable despite the currently 6 430 patent examiners employed by the USPTO, the highest number of patent examiners ever being employed at the same time. A patent examiner evaluates a patent application first and decides what to do with it — Granting, Rejecting, Pending.

The arising problem comes from the way a patent document is handled and classified. The USPTO is using a pre-defined hierarchy to classify a patent document called the IPC. It is a 4-level hierarchical classification system developed by a handful of patent offices around the world. There are 70 000 different categories at the lowest level of the hierarchy (subgroups). Due to the specificity of such a subgroup a patent examiner examines only a handful of subgroups. Since a patent document is assigned to more than one subgroup, patent examiners from different subgroups have to work together, slowing the process of classification down. This makes for a multi-level classification task.

The patent corpus is highly skewed in terms of domain distribution. The third hierarchical level of the IPC is called subclasses and is highly im-

balanced. There are subclasses with more than 300 000 patent documents assigned to them while others have only a few hundred. This makes automatic classification more difficult because subclasses with fewer documents might be penalized. All the statistics come from the USPTO[1].

A patent application contains several different text sections, amongst other things, a section for legal definitions and claims. The claims section of a patent application cannot be compared to regular English text. It is highly jurisdictional, certain words are therefore over-proportionally used in patent applications. This is true for almost all text sections in a patent document but it is most visible in the claims section. Sentences do not necessarily follow the grammar structure of regular English text, which is a necessity to circumvent certain rules employed by the USPTO in regards to patent applications (e.g. one claim has to be formulated with a single sentence, the abstract section has to be below 150 words, ...). A patent document is a relatively long text document, sitting between a news article and a book. Due to the many different domains covered by the USPTO and the long documents, the patent corpus is diverse, containing millions of unique words.

Since the characteristics of a patent application are different to text documents from other corpora used in machine learning (e.g. Reuters corpus, U.S Foreign Broadcast Information Service (FIBIS), ...), methods coming from academia have to be thoroughly tested although they are proven to improve the classification performance on other corpora.

solves

This thesis tries to solve the problem of the classification task the USPTO faces every day. It specializes in the processing of patent applications written in the English language. Therefore a feature selection algorithm is proposed. In text classification a unique word is translated into a feature, which is used by the classifier to differentiate two different categories.

The proposed feature selection algorithm reduces the number of unique words used for the classification task. The algorithm removes words that do

[1] `http://uspto.gov` – visited on 22 Aug, 2011

not contribute to the meaning of a patent application as well as redundant words. When two words only occur in physics patent applications, the classifier would need only one of the two words to classify the application as belonging to physics. Keeping both words is redundant and wasting CPU, hard disk and RAM resources. Furthermore a metric is tested which identifies the most and least important unique words in the patent corpus. While this does not necessarily lead to an improved classification performance, patent offices are still interested in identifying keywords in a patent application.

contribution

In this thesis I propose a new metric and three feature selection algorithms. The metric is based on the work of Montemurro and Zanette [33]. They proposed a Mutual Information (MI) based metric to assign a numeric value to every word in a book. Those numeric values are used to sort the words to find out which are the most and least important words. Books they used were “Moby Dick" by Herman Melville, “Analysis of the Mind" by Bertrand Russell and “The Origin of Species” by Charles Darwin. The metric splits the book up into several segments and computes the importance of a word within the segment. The results of the segments are then combined. Such an approach has the advantage of picking up the distribution of a word within the entire book rather than only looking at the total number of occurrences. Those books are on average 11 times longer than a single patent document.

Therefore an artificial corpus is created, which takes the multi-level property of the patent corpus into consideration. The metric is applied to the corpus, resulting in millions of numeric values. By utilizing those values, the proposed algorithm is capable of detecting redundant words and removes them. Due to the large number of unique subgroups, the fourth hierarchical level is disregarded in this thesis. All the experiments are run on the third hierarchical level (subclasses). Only three sections of a patent application are used: abstract, description and claims. Neither bibliographical information nor other metadata is used for the classification task. For the actual classifica-

tion task two linear classifiers (LIBLINEAR and SVM^{light}) are used. Due to the large number of features, the SVM^{light} is trained linearly and not utilizing kernels. This leads to a study drawing a comparison between patent classification using all the unique words (baseline) and patent classification with feature selection based on Montemurro and Zanette's metric. The performance of the two classifiers is compared too.

In addition to comparing it to the baseline it is compared to three other feature selection methods, namely MI, TF-Inverse Document Frequency (TF-IDF) and Information Gain (IG). The patent documents are provided by the Information Retrieval Facility (IRF) and their MAtrixware REsearch Collection (MAREC) repository. It contains approximately 3.5 million patent documents.

For the experiments three corpora are created. They are all sub sampled from the 3.5 million documents, containing 130 000 (100k), 220 000 (200k) and 100 000 (b100k) documents. The sub sampling was done by an algorithm designed by me and is explained later on. They contain 49, 75 and 50 subclasses respectively. In the b100k corpus every subclass has exactly 2 000 documents, making it a balanced corpus. This is used to see if a balanced corpus makes a difference compared to a highly skewed corpus distribution.

A framework is developed to run an efficient grid search on the parameters that exist in the metric, the algorithm and the classifiers.

results

As is shown in Chapter 5 the proposed algorithm is highly concurrent and is capable of removing 98% of the features while improving the classification performance compared to the baseline (which uses all the available features) by 8%. Such a reduction speeds up the classification task by up to 800%.

Chapter 4 shows the results of an online survey. In the online survey 25 people participated, evaluating the results of the metric in terms of its capability of predicting whether or not a word is useful in the patent corpus.

outline

Chapter 2 gives an introduction to the necessary background information needed to follow the rest of the thesis.

Chapter 3 explains the USPTO, the patent system, the corpus used and the patent classification hierarchy.

Chapter 4 explains feature selection on patents with the MI based metric of Montemurro and Zanette [33].

Chapter 5 shows the results of the experiments, divided into three sections — one section for each algorithm.

Finally, Chapter 6 concludes.

2 BACKGROUND

CONTENTS

This chapter provides the background needed for this master thesis. The explanations are kept short but references are provided for readers trying to deepen their understanding on the discussed subjects.

First current research in patent classification is presented. It is followed by an introduction to text classification as well as to classifiers, which are necessary for text classification. Since we are working with text, linguistic features and how they were handled is shown as well. The different evaluation measures are explained, concluded by what feature selection is and when and how it is used.

2.1 INTERNATIONAL PATENT CLASSIFICATION

Before I show related works, I introduce the IPC because most of the research involving patents and patent offices require knowledge about it. The IPC is the foundation of the pre-defined categories. As previously stated, each patent document is assigned at least one category manually by an IP reviewer. The categories are pre-defined in the IPC, which was established by the Strasbourg

	Category	Size	Example	
			Symbol	Title
1.	Section	8	G	Physics
2.	Class	129	G01	Measuring; Testing
3.	Subclass	639	G01N	investigating or analyzing materials by determining their chemical or physical properties
4. Group				
	Main	61 847	G01N 1/00	Sampling; Preparing specimens for investigation
	Sub	69 199	G01N 1/32	Polishing; Etching

Table 1: The hierarchy of the IPC and the four hierarchy levels (level four is split into two groups).

Symbol	Name
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

Table 2: The symbols and the names of the IPC sections.

Agreement in 1971[1] (entered into force in 1975) and is still valid today, although in 2006 there was a reform, including adding more definitions and categories to the IPC. There is an explanation of the IPC and its hierarchy on the EPO website[2].

The hierarchical structure of the IPC is shown in Table 1, including an example subgroup "G01N 1/32". Source for categories[3] and statistics[4]. There are four levels, the highest one is the section.

[1] http://www.wipo.int/export/sites/www/treaties/en/classification/strasbourg/pdf/trtdocs_wo026.pdf – visited on 22 Aug, 2011

[2] http://www.epo.org/patents/patent-information/ipc-reform.html – visited on 22 Aug, 2011

[3] http://www.wipo.int/classifications/ipc/en/guide/guideipc2009.pdf – visited on 22 Aug, 2011

[4] http://www.wipo.int/classifications/ipc/en/ITsupport/Version20090101/transformations/stats.html – visited on 22 Aug, 2011

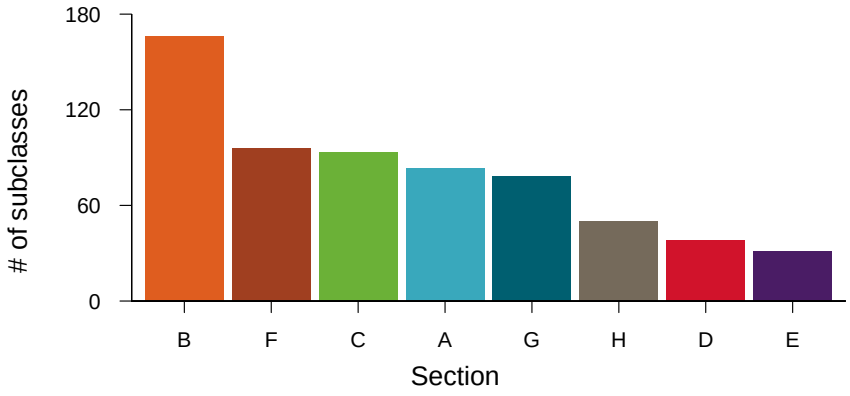


Figure 1: Number of subclasses per section.

The second column of the table shows the size of each level, which is the number of unique categories per level. Multiple categories can be assigned to the same document. At the subclass level, some categories are almost always assigned with a second subclass, where the two subclasses do not have to be within the same section. This makes text classification all the more difficult. A category search is available at the website of the WIPO[5], which can be used to see these connections too. Table 2 shows the symbols of the first hierarchical layer of the IPC and their names.

Noticeable is the fourth level called Group. It is solely an aggregation level. When on the fourth level the hierarchy splits up into two groups: either a main group or a sub group.

Some patents are misclassified by the patent examiners employed at the patent offices. Even if the misclassified document is known, revision can take years to take place. It might even happen that such a revision will never take place due to a shift in priorities (e.g. high load of unclassified documents). In addition Fall et al. [11] claim that most changes happen on group and main group level.

The number of subclasses per section varies. Figure 1 shows the difference between the sections. Section B is the largest section with almost 170 subclasses. This is roughly 5.5 times larger than section E. Such a skewness makes the classification task more difficult.

I used the third level for my thesis. There are not enough samples in the fourth level, Group, to justify training almost 70 000 classifiers. Furthermore the training of 70 000 classifiers is infeasible with the available computing resources! In addition this thesis focuses on feature selection and not patent classification, which is why the emphasis lies on reducing the number of features and not making the classifier perform better. This is the reason why no sophisticated classification method is used (e.g. hierarchical or ensemble based classification).

[5] http://www.wipo.int/classifications/ipc/en/ – visited on 22 Aug, 2011

2.2 CURRENT RESEARCH IN PATENT CLASSIFICATION

Patents are a lucrative field, it is only reasonable that there has been previous research into the subject of patent classification. Many research articles in the field of patent classification use linear classifiers. Especially the SVM enjoys great popularity, hence three articles closely related to my thesis are mentioned below, all using SVM.

In 2003 Fall et al. [11] compared various multi-classification tools, among them an SVM but also an adapted Winnow algorithm, namely Sparse Network of Winnows (SNOW). They did not use feature selection and therefore were using 1.4m different and unique words. They kept all of the chemical and biological words, most notably whole DNA sequences. Overall SVM is one of the top classifiers, most often it is even the best, but surprisingly the Naive Bayes method was among the best for certain tasks.

In 2006 Kim and Choi [26] proposed a methodology for automatically categorizing patent documents by looking at the structural information of the patent document. They utilized the structural information of a patent in question such as “Purpose of invention”, “Title", “Application field" and “Prior art". They applied this method to Japanese patent documents and achieved an improvement of approximately 74% over the baseline system. The latter did not use any structural information of the patent document.

In 2007 Li et al. [28] used an SVM to automatically classify patents. Instead of using a linear optimization approach they used a kernel-based approach. Their kernel was of a unique design, specifically created for patent classification using citation information. Their experiments showed an averaged F-measure of about 89%, which is high compared to patent classification systems using only abstracts, claims and/or descriptions.

In 2009 Beuls et al. [4] investigated the differences between the Winnow algorithm [29] and an SVM when applied to the patent classification problem. The result of their experiments was a superiority of the SVM compared to the Winnow algorithm. They achieved an F1 measure — this measure will be explained later in this chapter — of around 70%.

Feature selection on patent documents is a different task to feature selection on regular text documents. The larger number of unique words makes it harder to find a connection between documents and subclasses. One might be able to increase an evaluation measure on the test corpus by keeping only the unique words but this is prone to overfitting. Once the trained model is used outside of the test corpus, it will fail because there is no guarantee those unique words will ever occur.

In 2004 Yu and Liu [46] introduced an efficient way to check the relationship between each of the features by approximating a Markov blanket for each feature. Symmetrical uncertainty is used to compute the information a feature adds to a single class, features falling below a certain pre-defined threshold are removed permanently. This results in a smaller set of features which then gets checked pairwise. The symmetrical uncertainty is computed for each pair and if a pair doesn't carry more information than one of the features carry within the class, one of the features will be removed.

In 2005 Kim et al. [25] applied Term Frequency (TF), TF*IDF (Term Frequency-Inverted Document Frequency) and TF*ICF (Term Frequency-Inverted Category Frequency) on Japanese patent documents. Both approaches are threshold-based. A threshold is set and all the features below this threshold are removed. In their experiments TF-ICF was slightly better performing than TF*IDF but clearly outperformed TF.

2.3 TEXT CLASSIFICATION

The text classification problem is defined in Feldman and Sanger [14] as:

Definition 1. *The task is to classify a given data instance into a pre specified set of categories.*

There are several distinctions. The three most important are:

binary A binary setting is the simplest of all settings. There are exactly two classes, e.g. an email can be "spam" or "no-spam". This means the class label y is either +1 or -1 (for notational convenience). In general terms this is expressed as $y \in \{-1, +1\}$.

The most common loss function for the binary case is the 0/1-loss. The classifier function predicts a class label $h(x)$ (x being the vector representation of a document) and the loss function returns either 1 if the predicted value is the same as y or 0 otherwise.

$$L_{0/1}(h(\mathbf{x}), y) = \begin{cases} 0, & h(\mathbf{x}) = y \\ 1, & otherwise \end{cases} \tag{2.1}$$

multi-class A multi-class setting is more difficult. Here, y can be chosen from l pre-defined class labels. In the email example, an email could be "spam", "friends" or "family". This results in $y \in \{-1, +1, +2\}$ or in general terms $y \in \{1, \ldots, l\}$.

This problem can be reduced to a binary setting under mild assumptions, as can be seen in Joachim [22].

multi-label A multi-label setting is similar to a multi-class setting but instead of only assigning one class label to a sample, one can assign as many as there are predefined class labels, e.g. an email could be both “spam" and “family". In general terms this is expressed as $\mathbf{y} \in \{+1, -1\}^l$.

This leads to the following loss function, where the Hamming distance measures in how many positions the prediction and the observed class label differ:

$$L_{Hamming}(h(\mathbf{x}), \mathbf{y}) = \sum_{i=1}^{l} L_{0/1}(h^{(i)}(\mathbf{x}), y^{(i)}) \quad (2.2)$$

This problem too can be reduced to a binary setting under mild assumptions, as can be seen in Joachim [22].

The patent classification task is a multi-label classification problem.

In text classification, a document is put into a vector representation (bag of words), where each entry is independent of the others and each of the token types in the document is considered a feature. Since text has many unique words, the result is a very large feature vector and therefore training in text classification takes longer than other classification tasks. The value of each feature in this feature vector is application-specific. One of the most used weighting methods is TF-IDF. In Subsection 2.5.2 I explain TF-IDF and other weighting methods further.

In general, misclassifications have serious implications, therefore high precision is favorable over high recall. This means a document containing four categories should rather be classified with two categories, from which both are correct, than eight categories, from which only three are correct.

2.4 CLASSIFIERS

This section will introduce all the classifiers that are implemented by the framework which was developed for this master thesis. It consists of two different classifiers: an SVM and an L2-regularized L2-loss linear classifier. Different implementations of an SVM are shown as well.

2.4.1 Support Vector Machine

An SVM is a binary classification method that tries to reduce the Structural Risk Minimization (SRM) [24] as opposed to the Error Risk Minimization (ERM). SRM tries to find the optimal subset (i.e. model complexity) minimizing the risk bound [41]. This is normally done by separating the training data through a hyperplane. Figure 2 shows the hyperplane (thick, solid line) separating the binary classification problem. The margin between the hyperplane and both training sets has to be maximized and only the training samples closest to the hyperplane on both sides are taken and used to predict the data. Those training samples are named Support Vectors and lie directly on the margin.

An additional feature of SVM is the possibility to use kernels. Kernels are a way to project the original data from a certain dimensionality onto another, higher dimensionality. This is useful for not linearly separable data because they might become separable in a higher dimensional space.

The impact of a higher dimensional kernel can be seen in Figure 3. It shows a dataset containing two classes. All data samples shown with the rectangle shape belong to one class, all shown with filled circles belong to the second. The left plot shows the dataset in its original form. Since the samples of the two classes mingle, it is not feasible to draw a hyperplane between the rectangles and the circles. This makes the dataset linearly inseparable. To solve this, the two dimensional dataset is transformed into a three dimensional dataset with the help of a kernel and is now linearly separable, which is seen on the right side.

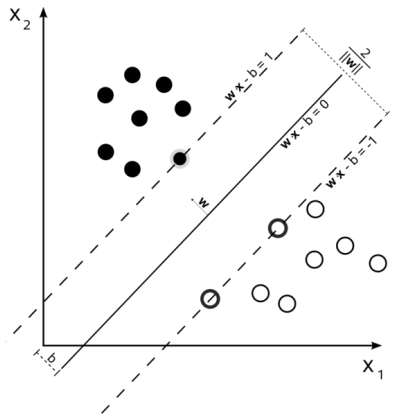
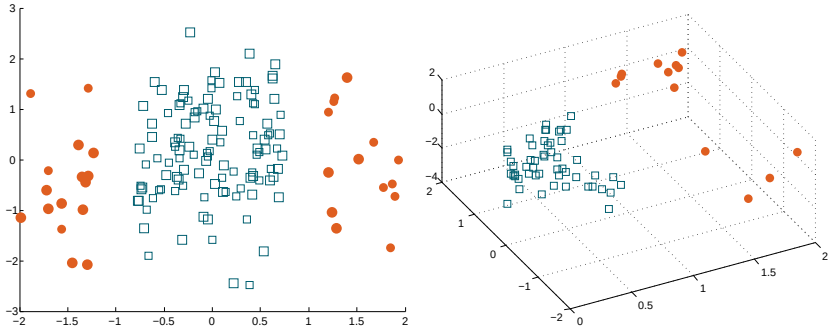


Figure 2: Trained support vector machine with the optimal hyperplane (solid line). Empty circles are class A, filled circles are class B. Hence this problem is linear separable. Dashed lines are the maximized margin. Source: Wikipedia[6] – visited on 22 Aug, 2011

[6] http://en.wikipedia.org/wiki/Support_vector_machine



(a) Linear inseparable 2D dataset **(b)** Linear separable 2D dataset with kernel projection onto 3D space

Figure 3: Transforming the data can make it linearly separable. a) Shows a 2D 2-class dataset. b) shows the same 2D 2-class dataset projected onto a 3D space with an appropriate kernel. The dataset is now linear separable.

To calculate the weight vector of an SVM with kernels [5], one normally uses the dual representation in terms of inner products:

$$\mathbf{w}^T\phi(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i t_i \phi(\mathbf{x_i})^T \phi(\mathbf{x}), 1 \leqslant i \leqslant N \tag{2.3}$$

where

$\mathbf{w}$	weight vector
$\mathbf{x}$	vector of training samples
α	used in dual representation for $\mathbf{w}$ during training
t	target labels $\in \{1, -1\}$
ϕ	kernel function
N	total number of training samples

As can be seen in Figure 2, the weight vector $\mathbf{w}$ is responsible for the location of the hyperplane in the weight space. The sum indicates that each training vector places a constraint on the possible choices of $\mathbf{w}$.

ϕ is the kernel and can be any function taking $\mathbf{x}$ as a parameter. For example, a randomly chosen nonlinear mapping $\phi : \mathbb{R}^2 \to \mathbb{R}^6$ is given by

$$\phi(\mathbf{x}) = \phi(x_1, x_2) = \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \sqrt{2}x_1x_2 \\ x_1^2 \\ x_2^2 \end{pmatrix} \tag{2.4}$$

In Equation 2.3 we have the product of two ϕ functions. By introducing the kernel from Equation 2.4 we have a way of computing the inner product directly as a function

$$\phi(\mathbf{x})^T \phi(\mathbf{y}) = K(\mathbf{x}, \mathbf{y}) \tag{2.5}$$

Such a direct computation method is called kernel function. This can be used in Equation 2.3, resulting in

$$\mathbf{w}^T \phi(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i t_i K(\mathbf{x}, \mathbf{y}), 1 \leqslant i \leqslant N \tag{2.6}$$

where

$K(\mathbf{x}, \mathbf{y})$	kernel function taking $\mathbf{x}$ and $\mathbf{y}$ as parameters
$\mathbf{x}, \mathbf{y}, \boldsymbol{\alpha}, \mathbf{t}$	same as in Equation 2.3

Other popular kernel functions are Radial Basis Function (RBF) kernels or polynomials of a certain degree [1].

Kernels can be non-linear as well. The usage of these kernels normally comes with a higher computational cost. For patent classification — or text classification in general — a linear separable dataset is assumed [23]. Therefore kernel based training will not be used in this thesis.

If no kernel is used, the SVM training is still costly since the training time of an SVM scales quadratically with the number of training instances, hence the need for a feature selection solution.

There are also two important parameters, C and J, which can be used in a grid search to fine tune the training. C is used when the training data is not linear separable. It balances a trade-off between the training error (which automatically occurs if an SVM is not linearly separable) and the margin. J is used to influence the training, especially when the categories have different number of documents assigned to them. If the training error occurs on positive training samples, J puts a higher cost on the training error compared to negative training samples.

To see how an SVM is used for patent classification, Section 2.2 lists some current research using SVM to classify patent and text documents from various repositories.

2.4.1.1 *LaSVM*

LaSVM [6, 7] is an adaption of the SVM. It is an online learning algorithm whereas the normal SVM is learned in a batch process. An online learning algorithm reads the samples not in a batch but one after another. The selection of the sample is random. In addition to this feature an online learning algorithm has the advantage of stopping the training at any time, therefore not

being forced to read, parse and train every available training samples. Bordes et al. [6] use this technique to reduce the number of samples the SVM has to look at. The early stopping criteria for LaSVM is a stable number of support vectors. If the number does not change further, LaSVM stops, no matter how many training samples have been used at that point.

LaSVM uses this technique to learn in situations where one has a multi-class classification problem and a very large and imbalanced dataset [10]. This leads to a significantly faster training, achieving a speed-up in the range of 200-2200% on real world datasets without sacrificing classification performance by more than 3% [7].

2.4.1.2 SVM^{light}

SVM^{light} is a state-of-the-art generic SVM library and is used to contrast any possible differences to the more advantageous LaSVM implementation. It was released in 1999 and is still maintained by Thorsten Joachims [22, 23].

It was chosen because it is a regular, straightforward implementation of an SVM, written in C. It proved to be as fast as other open-source SVM for the patent classification but it also provides access to various SVM parameters, like J and C. Furthermore the documentation is splendid and in depth.

2.4.2 L2-regularized L2-loss Linear Classifiers

LIBLINEAR [12] has been chosen as an L2-regularized L2-loss linear classification algorithm. It supports L2-regularized classifiers as well as L2-loss linear SVM, L1-loss linear SVM and logistic regression. L2-regularized L2-loss LIBLINEAR solves the following problem, shown in primal form:

$$\min_{w} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}(max(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i)) \tag{2.7}$$

where

- $\mathbf{w}$ weight vector
- C penalty parameter. Has to be $C > 0$
- y target labels $\in \{1, -1\}$
- $\mathbf{x}$ vector of training samples
- l total number of training samples

This primal problem is very similar to what an SVM tries to solve. The difference between an L2-regularized L2-loss linear classifier and a linear SVM is the different hyperplane. The hyperplane of SVM^{light} is of the form $y = wx - b$, whereas LIB-

LINEAR 's hyperplane is of the form $y = wx$. This means the hyperplane of a LIBLINEAR has to pass through the origin and does not have a bias term. The bias is useful to adjust the hyperplane when the training data is skewed or noisy. For a further account of the capabilities of LIBLINEAR the reader is directed to their official website[7].

An L2-regularized L2-loss linear classifier provides fewer features than an SVM. It does not provide any kind of kernel functionality, hence no data projection onto a higher dimension is possible.

Results of this particular classifier seem to be solid compared to an SVM[8], at least for text classification problems, but it requires fewer resources. I confirm this in Chapter 5. To train the Reuters Corpus Volume 1 (rcv1) takes only several seconds whereas training the same corpus on a state of the art SVM would take several hours. Since LIBLINEAR lacks features, a grid search testing all the parameters and their best settings is far less intensive as well.

2.5 LINGUISTIC FEATURES

This section briefly explains the used linguistic features as it is important for the results obtained in later chapters.

2.5.1 Normalization process

During the indexing, a few normalizing processes can be performed, such as tokenization and stemming. In addition, word specific rules have to be applied to the token types:

- every token type containing anything but alphabetic characters is removed (for simplicity reasons),
- lowercasing every token type,
- removing token types shorter than 3 characters,
- words separated by a hyphen are removed.

Whole XML-tags in the MAREC-XML file are removed as well. So far, only the tags “table" and “figure" are removed at the time of the extraction. They contain lots of additional information added by the IRF as well as non-textual data.

After this procedure, an optional word stemmer can be applied.

Stemming is a common linguistic process [37]. During the process of stemming the stem, base or root form of the token types

[7] http://www.csie.ntu.edu.tw/~cjlin/liblinear/ – visited on 22 Aug, 2011
[8] http://largescale.ml.tu-berlin.de/summary/ – visited on 22 Aug, 2011

Token	stemmer	stemming	stemmed
Stem	stemmer	stem	stem

Table 3: Token types and their stems.

in the corpus are extracted. However, the stem does not have to be identical to the morphological root of the word. Stemming solely depends on the language, since the patents used in my thesis are all composed in the English language, the Snowball stemmer[9] [3] is used. Another option, which has been tried on patents, is the Porter stemmer [11].

An example for stemming for the English language is shown in Table 3. It is not possible to perfectly extract all stems. For example a regular Snowball stemmer cannot resolve “stemmer" properly, hence yielding “stemmer" as its stem.

In text classification the stems replace the token types during the classification task (in both training and testing). Since the “new" corpus consists solely of stems, the number of features is lower than before. This leads to lower resource requirements for the training.

A stop word list is implemented as well. It is executed right before the classifier and removes token types from the feature list based on their frequencies in the corpus. However, it is only used for the baseline, which uses all the token types for the training and testing of the classifier. If a feature selection algorithm is used prior to the classifier, the stop word list is not used.

Why the stop word list is only used for the baseline and none of the developed feature selection algorithms becomes clear in Chapter 4.

2.5.2 Term Weighting

Term weighting is used for training and classifying with the various classifiers. The classifier receives an index of all the documents through a training and test file, where a single line represents one document. Such a document line contains all the token type IDs and the weight for the token type in the specific document. There are a few possible term weights commonly used for classification tasks [38]:

term frequency Term Frequency (TF) shows how often a term occurs in a single document D, calculated by $tf_j = n_j/|D|$, where n_j is the number of occurrences of the token type j in D.

[9] `http://snowball.tartarus.org/` – visited on 22 Aug, 2011

inverse document frequency Inverse Document Frequency (IDF) is the inverse of the Document Frequency (DF). df_j is defined as the number of documents in the corpus containing term j. IDF is an inverse because the number of documents in the corpus is divided by the DF and fed into a logarithm as follows: $idf_j = \log \frac{N}{df_j}$, where N is the total number of documents. This means the higher the number of documents containing that term, the lower the IDF.

tf-idf TF-IDF combines the above two measures by multiplying them as follows: $tfidf_j = tf_j * idf_j$. This gives a weight that shows how important a word is to a document in the whole corpus. The importance increases when the number of documents in which the term occurs is low but in the documents where it does occur it is heavily used, resulting in a high TF.

tf-idf mutual information Lu et al. [31] suggest to use TF-IDF and combine it with the MI by multiplying TF-IDF with MI. Since the algorithm from Montemurro [33] provides a MI-like metric, this weighting was tried but showed no success.

2.6 EVALUATION MEASURES

		predicted	
		p	n
actual	p	TP	FP
	n	FN	TN

Table 4: Confusion matrix, the positive (p) and negative (n) and how they represent the meaning of true positive (TP), false negative (FN), false positive (FP) and true negative (FN).

To be able to compare various classification algorithms, many different performance measures were introduced over the last few years, some of which are presented by Sebastiani [39]. Below is a list of the most used measures, abbreviations according to Table 4:

precision It is the number of correctly positively classified documents divided by the total number of positively classified documents, defined as: $\pi = \frac{TP}{(TP+FP)}$.

recall It is the number of correctly positively classified documents divided by the actual number of positive documents, defined as: $\rho = \frac{TP}{(TP+FN)}$.

f1 It puts recall and precision in a relationship to obtain a single metric where both are evenly weighted: $F_1 = \frac{2*precision*recall}{precision+recall}$.

accuracy It shows how many predictions made by the classifier were correct divided by the total number of predictions: $A = \frac{TP+TN}{TP+FP+FN+TN}$.

When having a multi-class or multi-label classification problem, each class has its own precision, recall, F1 and accuracy. There are two ways to combine one of the metrics from all the classes. The difference between these two is how they weight the individual classes:

micro-averages They are obtained by summing up the metric and dividing them by the number of categories, building an average over all categories. For precision this gives: $\pi^{\mu} = \frac{\sum_{i=1}^{|C|} \pi_i}{|C|}$, where μ indicates micro averaging.

macro-averages They are obtained by summing up the metric but weighing the influence of each category on the end result by the size of each category. For example, macro-averaged precision is defined as: $\pi^{M} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|}(TP_i+FP_i)}$, where M indicates macro averaging.

2.7 FEATURE SELECTION

Feature selection is the art of reducing the feature space by removing some features that are deemed unworthy of keeping. The motivating idea is to try and find a simpler model that has the simplest explanation that accounts for the data the best.

Figure 4 shows how feature selection works, how one gets from the entire feature set of the whole corpus to a new feature set containing only a few features. There are a lot of different ways how to arrive at just a few features and most of them are application specific. In this thesis we use a metric to rank the features and filter them accordingly.

The whole patent corpus used for this thesis has roughly 10 million token types, which translates into 10 million features.

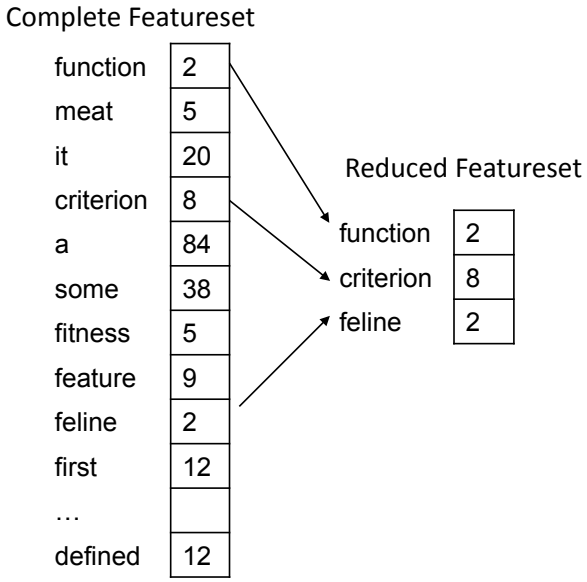


Figure 4: Mechanism of feature selection done on a feature set in vector representation. The number to the right of the feature represents the number of occurrences within the corpus.

This is not humanly interpretable and manageable, making the creation of a model more difficult.

An example, where the process of feature selection and the reasoning of the feature selection algorithm is easier to interpret, is the medical field. It has fewer features, most come from clinical measures through x-ray, CT and MRI scans, blood tests, . . . , but only a few relevant features will be able to predict one particular illness.

The other point is building a classifier, which is neither overfitted nor underfitted and will perform well on unseen documents. In such a case the priority lies on the feature set performing well and not on the individual feature. If both the negative, those with no linguistic value like function words, and too in-discriminative features are reduced, the quality of the predictions of the classifier should rise.

There are three major branches of feature selection, presented in Guyon et al. [19]:

wrapper Wrappers use the task at hand — classification, retrieval, clustering, . . . — to evaluate the found feature subset and evolve it. In this classification task, one would generate a new feature subset and train it against a classifier. The classifier returns various measures like precision and recall back to the wrapper. The wrapper incorporates this knowledge and generates more feature subsets and trains the predictor again. This is done until

a certain stopping criteria is met. Figure 5a shows the flow chart of a wrapper method.

One of the advantages of such a method is the direct application of the task at hand for the feature reduction. This results in a true relationship between the filter selection and the classification task.

The obvious disadvantage is the training and testing. As mentioned earlier, training a classifier with a few million (or even a few hundred thousand) samples is a time consuming task. Since many different solutions are created and evaluated, the evaluation has to be very fast. For the patent classification task evaluating all the classifiers with the whole corpus would take days though and is prone to overfitting. Therefore no wrapper method was used.

filter Filters are independent of the classifier. They select subsets of variables and deliver the subset as an index to the classifier. Figure 5b illustrates the process. The advantage of such a process is its classifier independence, it is even independent of the task. The same filter method could be used for both a classification task and a retrieval task. Another advantage is the performance. A filter method is generally considered to be very fast, especially on a high dimensional dataset. The reason is the lack of a classifier involved in the subselection. However, most filter methods are not as good as their wrapper methods, simply because they are too generic and normally approximate the classification task with another optimization function. In our case we approximate the classification task by measuring the relevance of a feature and remove the features that fall beyond a certain threshold. The metric used to measure the relevance is the metric suggested by Montemurro.

embedded Embedded feature selection algorithms embed the classifier in their method, creating a feature subset as well as a trained classification model. Figure 5c visualizes the process. The difference with embedded methods is they perform feature selection in the process of the learning process and output both a feature set and a prediction.

They also incorporate local searches or genetic algorithms to not have to run every possible feature subset from scratch.

The main disadvantage of the embedded methods is that the exact classification algorithm used has to be known prior to designing the feature selection because it returns a trained model using only a fraction of the features. With wrapper methods one can use any classification algorithm and write an evaluation method around it.

Though embedded systems are not new — Breiman et al. [8] introduced a decision tree incorporating feature selection in 1984 — they are becoming more and more popular because they are less prone to overfitting (since they use the classification algorithm during the feature selection process) but are still less computationally expensive than wrapper methods.

2.8 USEFUL INFORMATION

This section presents useful information about often used presentational mechanisms for this thesis.

2.8.1 Visualization

Many visualizations in this thesis use a so-called box plot. Figure 6 shows a box plot and highlights the most important parts.

(a) Wrapper

(b) Filter

(c) Embedded

Figure 5: Flow chart of major branches of feature selection.

For more information on box plots, see the books by Tukey and Tufte [43, 44].

The dots are outliers. Most of the charts have quite a few outliers and they are overlapping. This is why the outliers are jittered. The horizontal position of an outlier has no meaning and is only done to show the actual number of outliers.

When patent statistics are presented, showing eight or nine box plots within a single chart, the width of a single box plot visualizes the number of subclasses of a section. On some charts, I not only show the eight section box plots but also a column to the right named “All". It shows the data for the entire corpus, without any kind of grouping. This means if a box plot shows the data for subclasses grouped by the eight sections, the “All" box plot shows the data for all subclasses regardless of their sections. This might visualize outliers not shown by any of the eight other box plots.

Unless otherwise noted, the order of the section box plots is not alphabetical but by the medians in descending order.

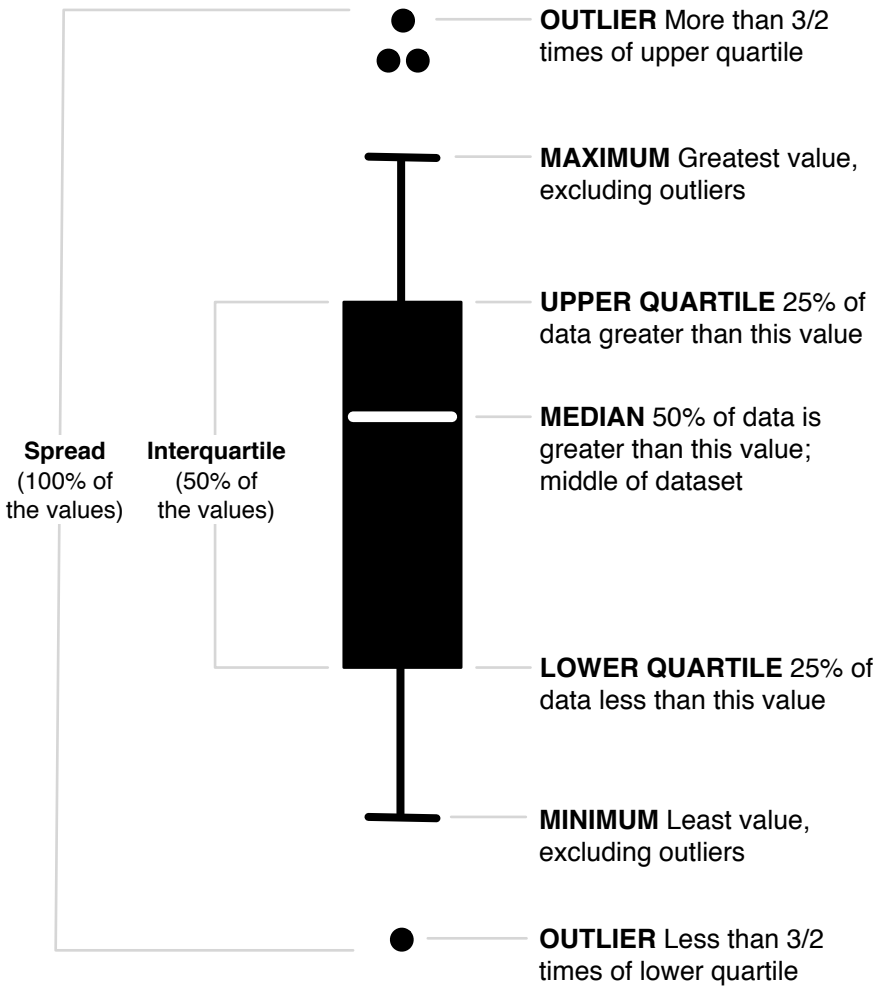


Figure 6: Box plot with most important bits highlighted.

2.8.2 Tables

In the tables of the upcoming chapters subheadings are used on occasion. The subheadings show the unit used in the column. Though normally subheadings are not orange and bold, they are shown this way for demonstrational purposes in Table 5. Data courtesy of Box Office Mojo[10]. There are a total of five different units used in the sub headings:

% The column shows regular percentages. This is especially used for F1, precision and recall.

pp The column shows Percentage Points (PP). This is used when the column shows the absolute difference between two columns that use percentages.

k Shorthand for 1000. The base of all the numbers in the column would be 1000, e.g. instead of 4000 it shows 4.

10k Shorthand for 10 000. The base of all the numbers in the column is be 10 000. For example instead of 90 000 it shows 9.

m Shorthand for 1 000 000. The base of all the numbers in the column is be 1 000 000. For example instead of 66 000 000 it shows 66.

	Weekend	Total	Cinemas	Weeks
	m	m	k	
Cars 2	66	66	4	1
Bad Teacher	31	31	3	1
Green Lantern	18	89	4	2
Super 8	12	95	3	3
Mr. Popper's Penguins	10	39	3	2

Table 5: Example table of box office results showcasing sub headings.

[10] http://boxofficemojo.com/weekend/chart/ – 2 Jul, 2011

3 CORPUS MATERIAL

CONTENTS

3.1 USPTO

This section presents information about the USPTO because the experiments use the USPTO repository of the MAREC collection.

	Total Backlog	Avg. Pendency	Examiners	Budget
	k	months		m
1988	268	20	1550	144
1998	481	24	2600	567
2008	751	32	5950	1915

Table 6: Development of four important figures over the last 20 years.

3.1.1 General Statistics

Table 6 shows the number of examiners, the budget, the total backlog and the average pendency time for a patent application for three specific years within the last 20 years. The statistics come from a report issued by the U.S. Chamber of Commerce[1].

If a table or chart is annotated with “Fiscal Year", the data covers annual periods that extend from October 1 to September 30. Otherwise the data covers January 1 to December 31.

All the statistics following about the USPTO only show Utility, Plant, Reissue (UPR) patents. Besides UPR patents the USPTO has design patents. Those patents are a small fraction of the daily business of the USPTO, since it assigns only 99 patent examiners to design patents. This equals roughly 1.5% of the total patent examiners.

The annual budget exploded in the last 20 years. In 2008 the USPTO had almost 2 billion dollars at their disposal, a fifteen-fold increase over their budget in 1988. The USPTO is not allowed to work under a budget deficit. Therefore it is responsible for its own budget. Most of the income is generated by patent application fees. Because of this the number of patent examiners depend on the number of applications per year. The higher the number of applications, the higher the budget, the more patent examiners can get hired. The increase of the budget helped hire more patent examiners and yet the backlog increased too, the average pendency went up as well.

[1] `http://ipo.informz.net/ipo/data/images/14492_patentoffice_fm4_12.25.08.pdf` – visited on 22 Aug, 2011

3.1.2 Budget

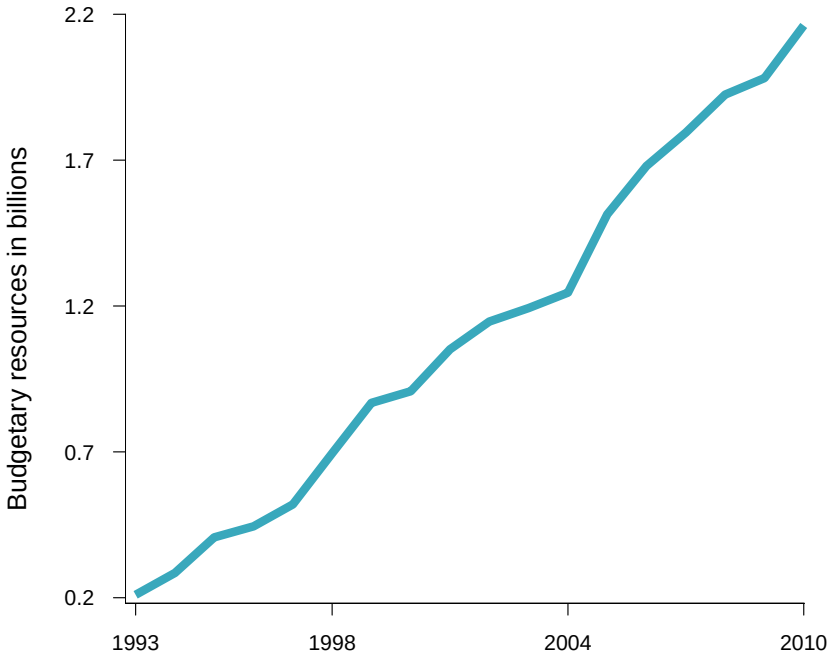


Figure 7: Total Status of Budgetary Resources.

Considering the financial independence of the USPTO, the increase of their budget without spending the money of the US tax payers is astonishing.

Figure 7 shows the total status of budgetary resources from 1993 to 2010. The y-axis is in billions of dollars. The budget is monotonically increasing. This comes from the increase in the number of filed patent applications as well as the number of granted patent applications. The patent applicant has to pay a fee for the initial patent examination and, if granted, a maintenance fee after 3.5, 7.5 and 11.5 years.

3.1.3 Reclassification

Since the USPTO uses the IPC hierarchy, they have to classify every patent application. I use their classification work for supervised learning, hence depending on their accuracy. When the USPTO has wrongly classified a patent application, my classifiers change their hyperplanes to accommodate the document, which yields a mis-modeled classifier.

Because of this Figure 8 shows the reclassification activity of the USPTO for the last 20 years. The y-axis is shown in $\log_{10}$. The top line represents the number of reclassified patents while the bottom shows the number of newly established subclasses. The USPTO calls them subclasses, due to the sheer quantity it can be deduced that they talk about groups. Not only does the USPTO use

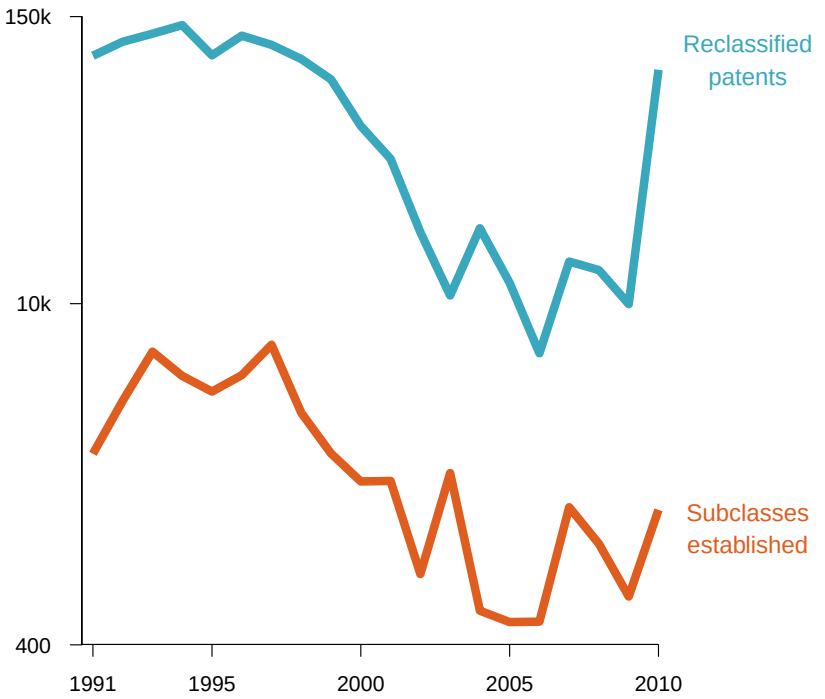


Figure 8: Professional patent reclassification activity.

the word subclass for those numbers but also at various places on their website. The source for this figure is the annual reports published by the USPTO, available on their website. Data prior to 1995 is shown in the annual report of 1995.

The reclassification of the patents goes hand in hand with the subclasses. In the 90's many new subclasses were introduced, forcing many patents to be re-evaluated. In recent years the reclassification activity is relatively low, between 10 000 and 20 000 in most years. However in 2010, 90 000 were reclassified. Unfortunately the repository of MAREC contains only documents up to the fiscal year 2008, hence not catching any of those 90 000 patent applications from the last reclassification project.

3.1.4 Patent Examiners

Figure 9 shows the number of personnel from 2001 up to 2010. It is split up into patent examiners and the remaining federal personnel. While the USPTO has committed to hiring 1200 patent examiners each year up to the year 2013, the estimation is not shown in the figure. Because one for every two hired patent examiners left the USPTO in the last few years, it is unclear what the number of patent examiners will be in the next three years, despite having the early commitment to hiring new examiners.

[1] `http://uspto.gov` – visited on 22 Aug, 2011

[1] `http://www.uspto.gov/about/stratplan/ar/index.jsp` – visited on 22 Aug, 2011

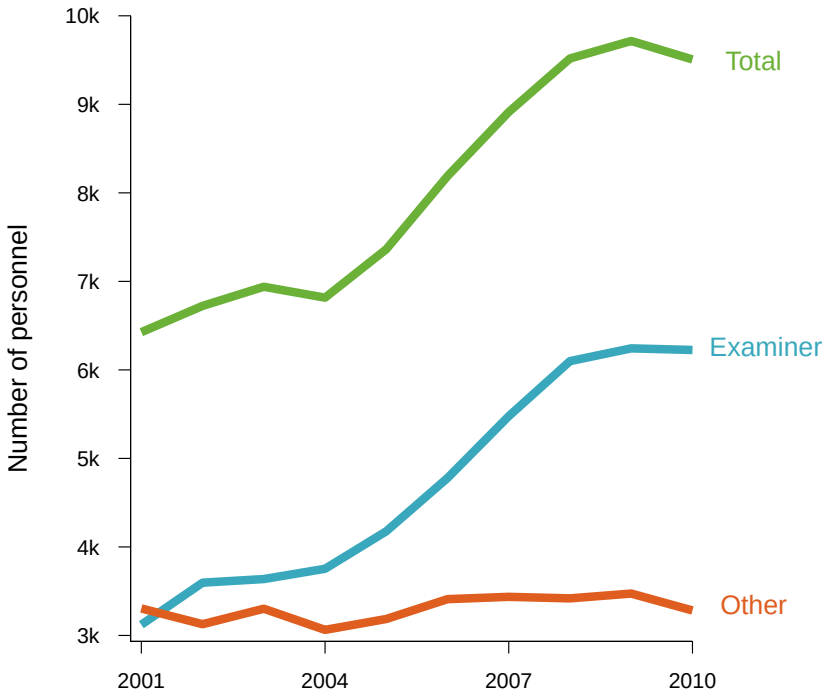


Figure 9: USPTO personnel of the last ten years.

The number of non-examiners at the USPTO remained almost the same over the last nine years while the number of patent examiners doubled. Most of the budget increase goes into hiring new patent examiners not other federal personnel. Patent examiners are federal personnel too. If the government of the United States of America (USA) orders a hiring stop, the USPTO is not allowed to hire new patent examiners, regardless of the annual budget. This is necessary because of a hiring stop in the mid 90's, where the government decided to cut federal personnel and hence make it impossible for the USPTO to get new patent examiners despite an increase of the workload.

The problem the USPTO is facing since 2002 is the high fluctuation in the patent examiner workforce[2]. Ever since 2002 one patent examiner left for nearly every two the agency hired. Worse, 70% of those who left had been with the agency for less than five years. This is worse because the new recruits are hired to fight the backlog and require additional training.

The United States Government Accountability Office[3] issued a report in September 2007 investigating the Patent Application Backlog at the USPTO. They asked the patent examiners for the reason they were leaving:

> *According to USPTO management, patent examiners leave the agency primarily for personal reasons, such as the job*

[2] `http://www.uspto.gov/web/offices/com/annual/2007/2007annualreport.pdf` – visited on 22 Aug, 2011

[3] `http://www.gao.gov/` – visited on 22 Aug, 2011

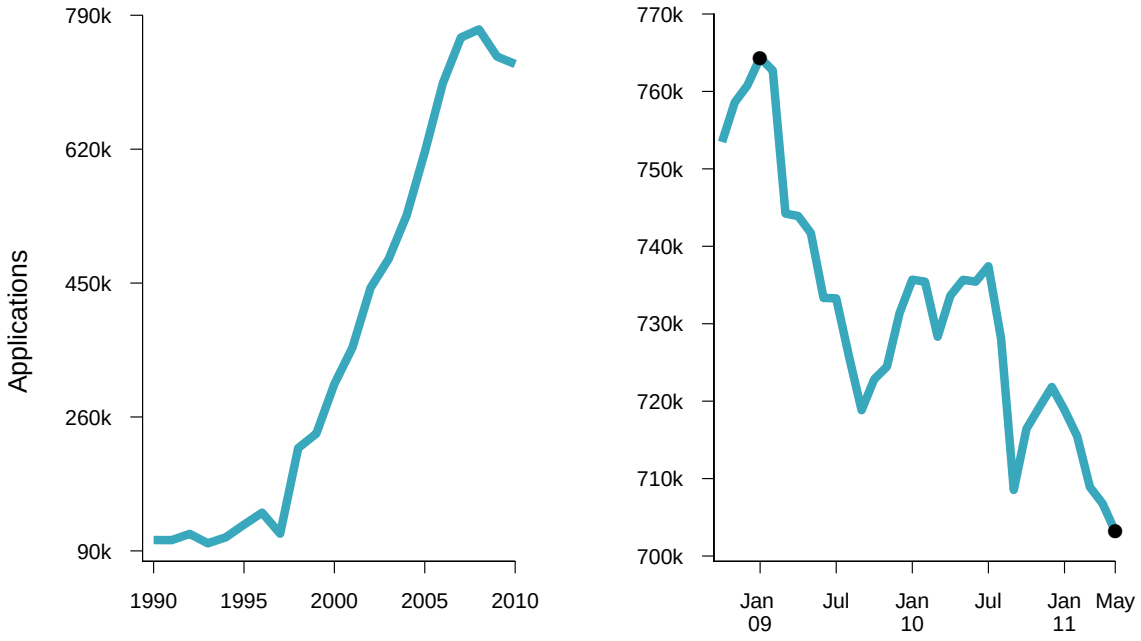
> *not being a good fit or family reasons.* ***In contrast, 67 percent of patent examiners identified the agency's production goals as one of the primary reasons examiners may choose to leave USPTO.***
> — U.S. Government Accountability Office [35]

> *These production goals are based on the number of applications patent examiners must complete biweekly and have not been adjusted to reflect the complexity of patent applications since 1976.*
> — U.S. Government Accountability Office [35]

3.1.5 Backlog

As mentioned before, the USPTO has a problem keeping their workforce, which in turn has an impact on the backlog. Figure 10 shows the backlog in detail from the beginning of fiscal year 2009 to now, resulting in 32 months. Additionally the last 20 years are shown on a per-year basis.

For the USPTO a patent application belongs to the backlog if no First Office Action (FOA) has taken place, meaning no patent examiner has seen, touched or classified it. Ergo whenever a FOA is applied on the patent application, it gets removed from the backlog stack. Therefore the backlog does not equal the set of pending patent applications, which is by far larger.



(a) Last 20 years.

(b) Last 32 months.

Figure 10: Backlog.

In the last 32 months the backlog decreased overall but unfortunately not in each month. From January 2009, the highest peak of the last 32 months, to May 2011 the reduction was a mere ~60 000 patent applications. With such a performance it does not surprise that the USPTO thinks it cannot reduce the backlog within the next 6 years. Despite hiring 1200 patent examiners each year.

3.1.6 Patent Applications

Figure 11 shows the number of electronically filed patent applications in percent for the last five years as well as the estimation for the next two years. Amazingly only 14.7% of all the patent applications at the USPTO were filed electronically as recently as five years ago. Worse, before March 2006 the USPTO printed electronically filed patent applications and then re-scanned those prints. Nowadays electronically filed patent applications are processed all digitally and paperless. This not only has an impact on the number of federal personnel needed to process a single patent application but also on the backlog, since the number of non-patent examiner personnel stayed the same. Today approximately 90% are filed digitally and it is expected to raise to almost 100% in the next few years.

Figure 12 shows the number of patent applications the USPTO has received and granted each fiscal year for the last 46 years[4]. The backlog and abandoned lines start at 1988 due to a lack of prior data.

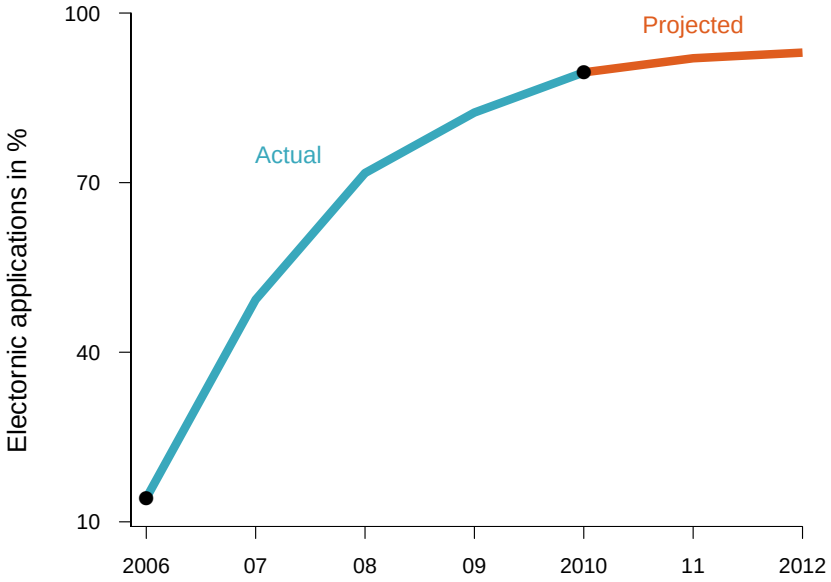


Figure 11: Percentage of electronically filed patent applications.

[4] `http://www.uspto.gov/web/offices/ac/ido/oeip/taf/us_stat.htm` – visited on 22 Aug, 2011

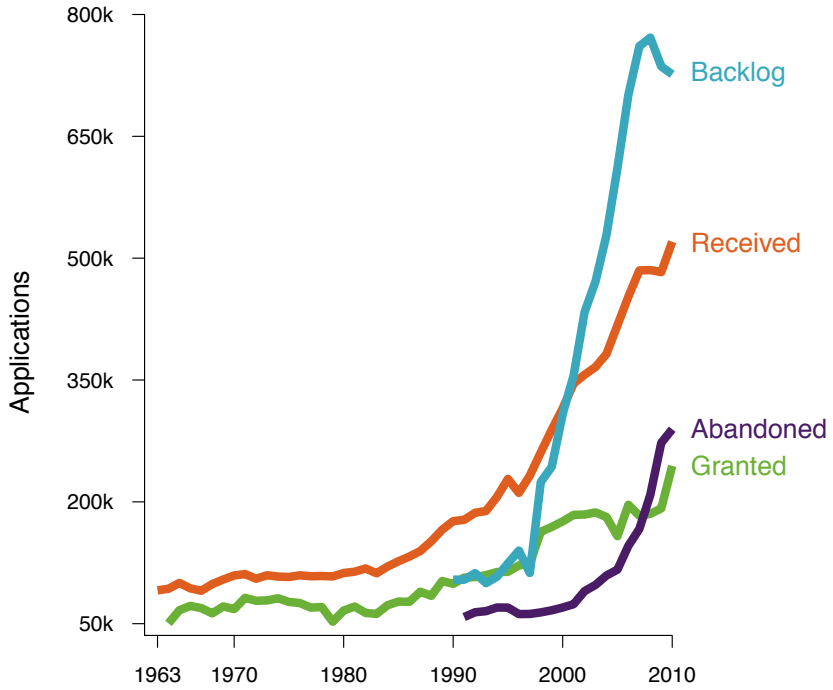


Figure 12: Received and granted patent applications.

It takes more than one year to decide on a patent. Therefore the line “Granted" does not show how many of the filed patents in a specific year have been granted. Rather it shows the total number of granted applications that year without paying attention to the filing date of the application. The same is true for “Abandoned".

In the last ten years the USPTO received more patents each year than the year before. The increase in those ten years is almost 200 000 patents — or in other words: 50 percent. On the other hand patent examiners did not grant more patent applications in relation to the increase of applications. The number of granted applications has stayed around 190 000 per year. The backlog increased in more or less the same way the number of received applications did. Coupled with the lack of patent examiners, budget problems and rapid increase in patent applications, there seems to be no way the USPTO can eliminate the backlog any time soon.

The number of abandoned patent applications took a steep increase from the year 2000 onwards, closing in on granted applications and finally overtaking it in 2009. Considering the raising number of filed patent applications, this might not be surprising. This is why Figure 13 shows the relation between filed and abandoned/granted applications in percent with the same data used in Figure 12.

More than half the patent applications were abandoned in 2009 and 2010, from 2005 onwards it is monotonically increasing. The percentage of granted applications on the other hand is decreas-

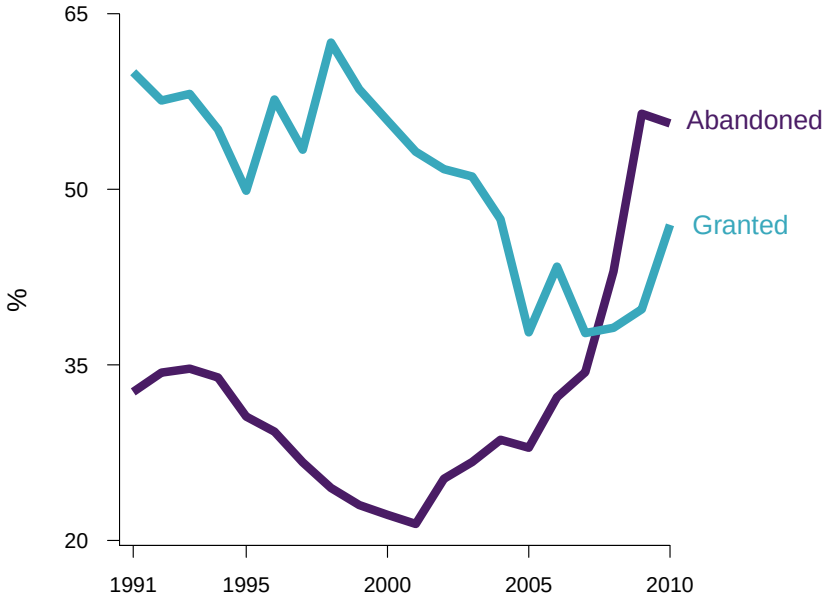


Figure 13: Abandoned/granted applications divided by received applications.

ing and is steady between 40% and 50% for the last ten years. Increasing abandonment and rejection means a lower total number of granted applications in relation to the filed applications. This leads to a lower budget in the mid- and long-term because of the lack of maintenance fees for granted applications.

Figure 12 showed the data until 1963. Unfortunately FOA and disposals, which are basically rejections, are not available as far back. For this reason the next figures rely on the annual reports issued by the USPTO, forcing me to only analyze the last ten years. In some cases the USPTO provides estimation for the next few years as well. These are shown when appropriate. The annual reports from 2001 to 2010 can be found on the website of the USPTO[5].

There is also a slight discrepancy between the annual reports and the data from their website, which shows the last 46 years. Additionally the 2010 data is preliminary and will be finalized in the 2011 annual report.

It is very difficult to determine the number of processed patent applications per year. The USPTO provides a number of processed items per year called Production Units (PU). PU are calculated by the number of FOA in a year plus the number of disposals divided by two. The backlog does not necessarily get reduced when the number of received applications is lower than the PU as the backlog is all the patents that have not yet been pushed into at least the FOA state.

[5] `http://www.uspto.gov/about/stratplan/ar/` – visited on 22 Aug, 2011

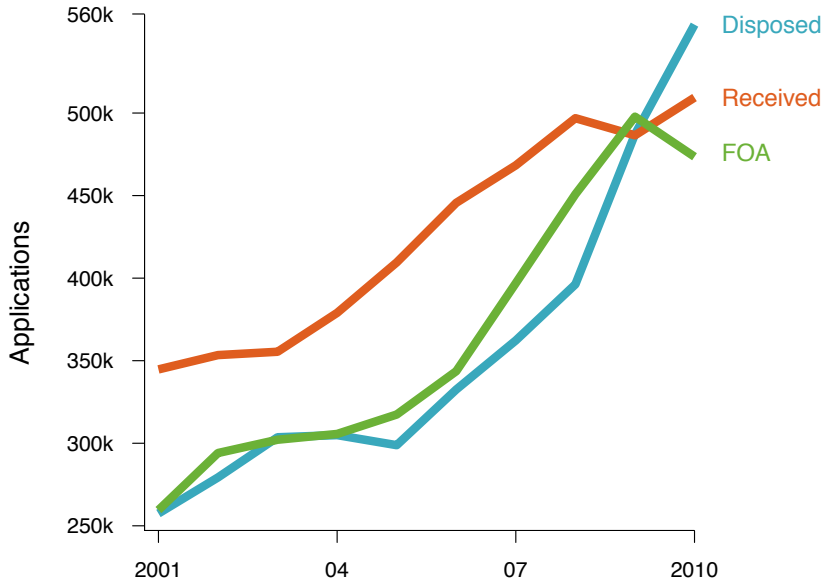


Figure 14: Total number of processed applications.

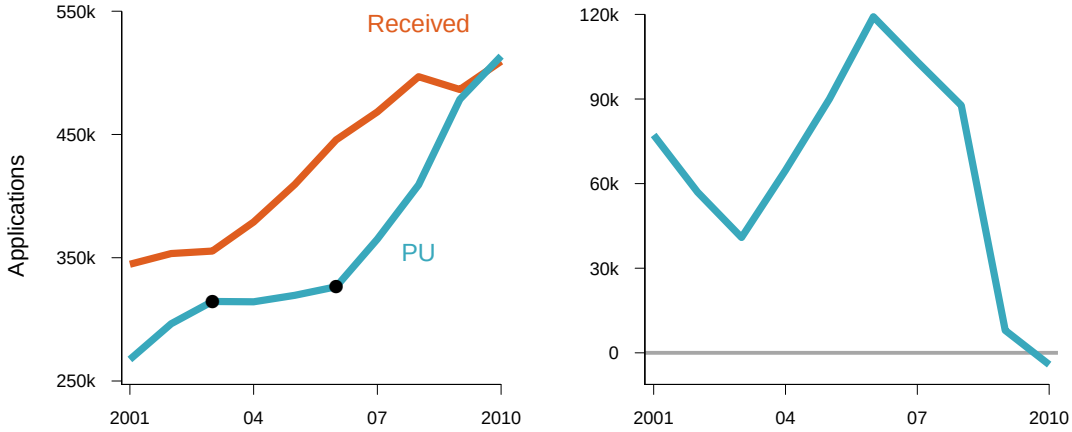
I do not understand how PU shows the total number of processed patent applications per year. For example if the USPTO pushes ten applications into FOA state and zero applications into the disposed state in a single year, the PU is five. Since a patent application takes on average 25 months until it reaches FOA state and 36 until its disposal, it is unlikely both events fall into the same fiscal year. Dividing by two is useless.

Therefore I present both the number of disposed as well as the number of FOA applications in Figure 14. For this figure the number of received patent applications comes from the annual reports. Neither of those two numbers represent the number of processed applications in a single year as they might have overlapping patent applications.

The number of disposed patent applications is important because it finalizes the application, removing it from the pending stack. For the first time in over 20 years, the USPTO finalizes more patent applications in a year than it receives. This creates a possibility to work on the existing backlog. Unfortunately the number of FOA applications declined in the last two years, the first time in ten years. A valid theory would be that the USPTO sacrificed FOA applications for disposals to reduce the number of pending applications.

Figure 15 shows two things. First the number of received applications minus the number of disposals (abandoned plus granted plus rejected applications). Second the number of received applications minus the number of FOA applications. This shows why the backlog has been increasing steeply every year from 2001 up to 2009. 2009 is the only year in which the number of FOA applications is higher than the number of received patent applications.

Figure 15: Received applications minus disposals/FOA.



(a) Received and PU applications

(b) Received minus PU

Figure 16: Applications versus PU.

The last two years are the only years in which the USPTO was able to reduce the number of pending patent applications in the last ten years. With the estimated rising number of patent examiners, the number of disposals should go up and the difference to the number of received applications should go down even further.

Figure 16 comes back to the previously mentioned official number of processed patent applications named PU. It compares PU to the number of received applications in the last ten years. Additionally the deviation between the two is shown. Over the last few years PU is catching up to and even surpassing the number of received applications.

Between 2003 and 2006 the PU remained at almost the same level. This overlaps exactly with the time frame in which very few patent examiners were hired, regardless of the steep increase in the number of patent applications during those years.

3.1.7 Pending Applications

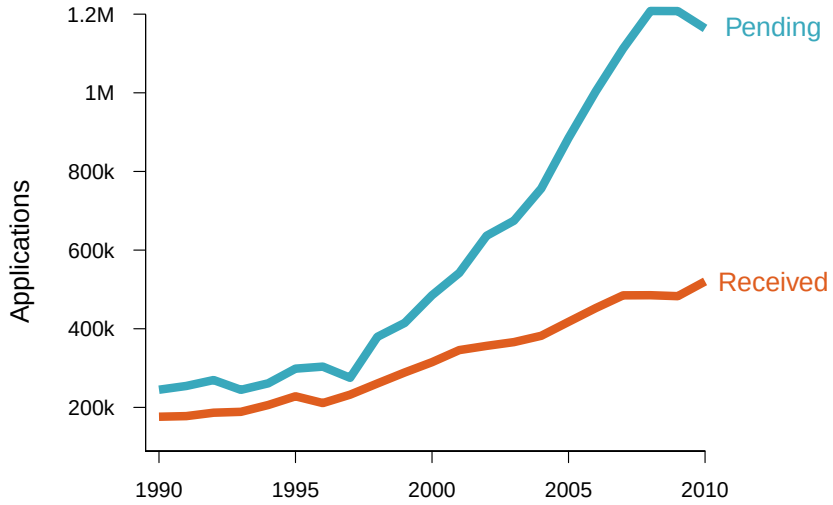


Figure 17: Pending applications over the last 20 years.

As stated before, the backlog issued by the USPTO are not all unfinished patent applications, only those with no FOA. Pending patent applications on the other hand are all unfinished patent applications, including the backlog.

Figure 17 shows the number of pending and received patent applications for the last 20 years. In the last three years it is almost three times larger than the number of received patent applications, despite the USPTO having the most voluminous filed patent application years to date.

One of the reasons for the sharp increase to almost 1.2 million pending patent applications is the massive backlog. The other is pendency in general, as I will show later in Subsection 3.1.8. There is at least a three month period after the patent reaches FOA state in which the patent applicant has to respond to the questions and issues of the USPTO. To shorten the number of questions and issues and reducing the period after FOA, the USPTO now makes it possible to get an interview with the patent examiner directly. This should cut down total pendency and should be directly responsible for a reduction in the number of pending patent applications.

3.1.8 Pendency

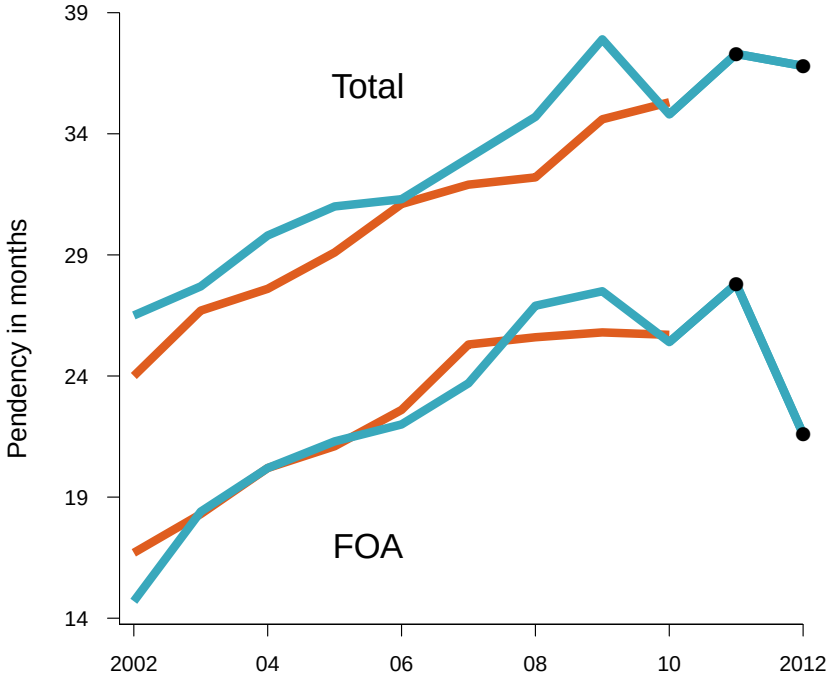


Figure 18: Average target and actual application pendency.

Figure 18 shows the average for both the target and actual patent application pendency of the last ten years. The time frame of pendency is given in months. The top two lines show the total pendency of a patent application, the bottom two show the FOA pendency. Total pendency is measured as the average number of months from the application filing date to the date the application has reached final disposition (e.g. granted or abandoned). The goal of the USPTO is to reduce the total pendency to 20 months by 2015. The date of FOA is measured as the average number of months from the application filing date to the date the application gets a first response by a patent examiner. For both FOA and total pendency, the lower the y-axis value, the lower the pendency, the better.

Both pendencies increased over the last ten years, as could be expected with the sharp increase of patent applications. However, the estimation for 2011 and 2012 are specially interesting. FOA is predicted to drop to 21 months but seems to have no impact on the total pendency, which stays above 35 months. This is 15 months below their goal, with only three more years to go at that point.

3.1.9 Origin of Inventors

3.1.9.1 *By Assignees*

Table 7 shows the top 20 patent assignees in 2010. The source of the table is a technical report by the IFI[6]. The darker a colour, the more patents a country has filed in 2010. The origin of a patent is determined by the residence of the first named inventor.

Only four out of the top ten patent assignees come from the USA. Considering Germany files the second most patents per year, it is surprising there is no German company in the top 20. The top 20 patent assignees get granted approximately 40 000, which translates into roughly 18% of all granted patents in 2010.

	Company	Country	#
1	IBM	USA	5896
2	Samsung Electronics	S. Korea	4551
3	Microsoft	USA	3094
4	Canon	Japan	2552
5	Panasonic	Japan	2482
6	Toshiba	Japan	2246
7	Sony	Japan	2150
8	Intel	USA	1653
9	LG Electronics	S. Korea	1490
10	HP Development	USA	1480
11	Hitachi	Japan	1460
12	Seiko Epson	Japan	1443
13	Hon Hai Precision Industry	Taiwan	1438
14	Fujitsu	Japan	1296
15	General Electric	USA	1225
16	Ricoh Co	Japan	1200
17	Cisco Technology	USA	1115
18	Honda Motor	Japan	1050
19	Fujifilm	Japan	1041
20	Hynix Semiconductor	Japan	973
	Total		**39 835**

Table 7: Top 20 Patent Assignees in 2010.

[6] `http://www.ificlaims.com/news/top-patents.html` – visited on 22 Aug, 2011

As of July 2011 the price tag for a basic patent application filing is US$ 330, hence IBM spent a minimum of two million dollars on filing at the USPTO last year alone.

3.1.9.2 *By Country*

Figure 19 shows the 20 countries with the most filed patents per year at the USPTO for the last four years, excluding patents from the USA. With almost 220 000 last year, the USA files by far the most patent applications. Since some countries behave almost identically, I built seven groups, their IDs ranging from A1 to A7. The darker a country, the more patents it has. Data provided by the USPTO[7].

The only country with a decrease in filed patent applications over fours years is Taiwan. Overall the fewer patents a country has registered in the last four years, the lower the absolute increase. Not surprisingly Japan has the highest increase. The absolute increase over four years by Japan is higher than the combined number of applications of the last two years of any of

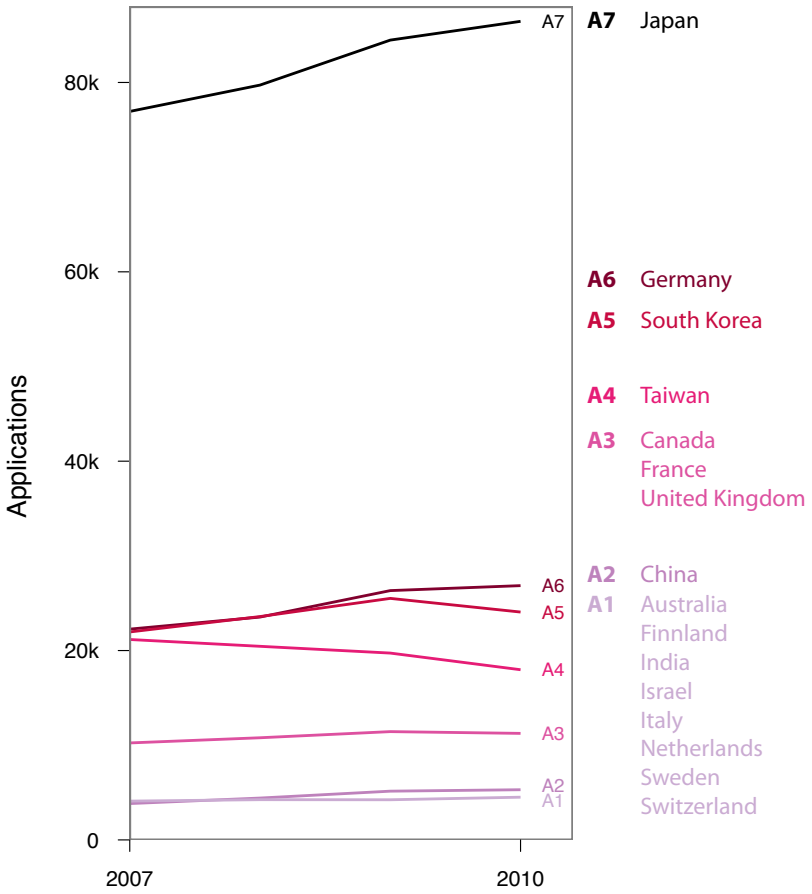


Figure 19: Origin of filed patent applications over last four years.

[7] http://www.uspto.gov/web/offices/ac/ido/oeip/taf/cst_utl.htm – visited on 22 Aug, 2011

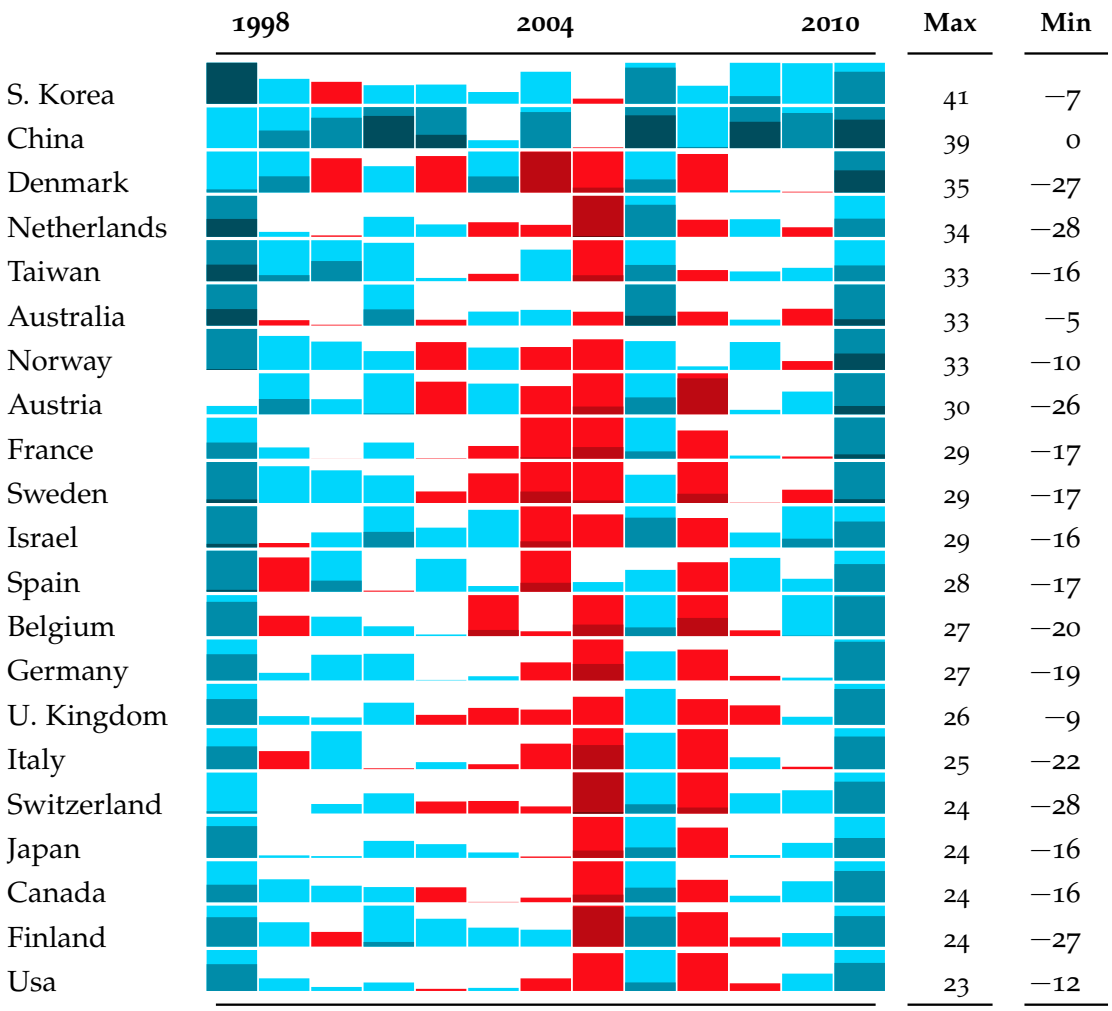


Table 8: Granted patents deviation in descending order of maximum deviation in percent. Red = negative, blue = positive. Darker > Lighter. Visualization explained in Few [16], Heer et al. [20].

the countries from A1. In a single year Japan gets more patents filed at the USPTO than the last four years combined for any of the countries in A1, A2, A3 or A4. On the other hand there is no country with a steep increase; the number of patent applications did not explode in the last four years. Considering all countries but Japan are below 25 000 applications per year, a steep increase would be very much possible.

Table 8 shows the annual deviation of granted patents for 21 countries with the most granted patents. Especially interesting is the timespan 2004–2005. In 2005 all countries but Spain experienced a decrease in the annual number of patents granted, 16 out of 21 in 2004. This means the drop in 2005, seen in Figure 12, does not come from a single country but rather from the USPTO itself.

Figure 20 highlights the origin of the inventors on a world map by adjusting the color of their countries. Again, the darker the color of a country, the more patents its inventors have registered. This time, not the last four years are shown but the number of granted patent applications from 1963 to 2010. The USA is responsible for 58% of the patents and is neglected on the world map. It will be analyzed in more detail in other figures. The number of patents represented by the color is shown in the legend on the bottom left. Countries with less than 2000 patents — less than 0.05% — are not highlighted. Black shows the country with the most patents outside of the US, Japan. Japan has 810 000 patents, more than the combined number of the second, third and fourth country. 96.8% of the countries have less than 35 000 patents registered at the USPTO. On the South American continent, only one country - Brazil - qualifies for inclusion on the map, the same number of countries as on the African continent (South Africa) and in the Middle East (Israel).

Portugal has the lowest number of patents of all the western European countries. It has only approximately 290 registered at the USPTO, on par with the Seychelles.

Figure 21 shows the same data but normalized to one million inhabitants. The number of inhabitants for each country is taken from the most recent estimation, done in 2010, by the United Nations Department of Economic and Social Affairs[8]. No country from Latin America, Africa or mainland Asia has more than 250 patents per one million inhabitants. Japan, despite being a highly populated country, holds the third rank behind Switzerland (7200) and Liechtenstein (8400). Japan, Taiwan and South Korea are still the top three countries in Asia, the latter two have no bracket change. Not surprisingly, smaller countries like Monaco, Gibraltar, Luxembourg, Singapore, Hong Kong and some of the Atlantic and Pacific islands “benefit" from the normalization. Except for Liechtenstein, none of them are in the top three brackets.

On the other end of the spectrum are India and China. Both lose the most ranks after the normalization because of the more than one billion inhabitants residing in each country.

Another trend is the importance of the Nordic countries, all of which are in the fourth, fifth, sixth or seventh highest bracket. Denmark is shown as the Kingdom of Denmark, composed of Denmark, Faroe Islands and Greenland. The latter two only contribute one patent each and only a few tens of thousands of inhabitants. A possible explanation for this trend might be the encouragement of the governments to pursue higher education and providing higher research grants than in other countries.

[8] `http://esa.un.org/unpd/wpp/unpp/panel_population.htm` – visited on 22 Aug, 2011

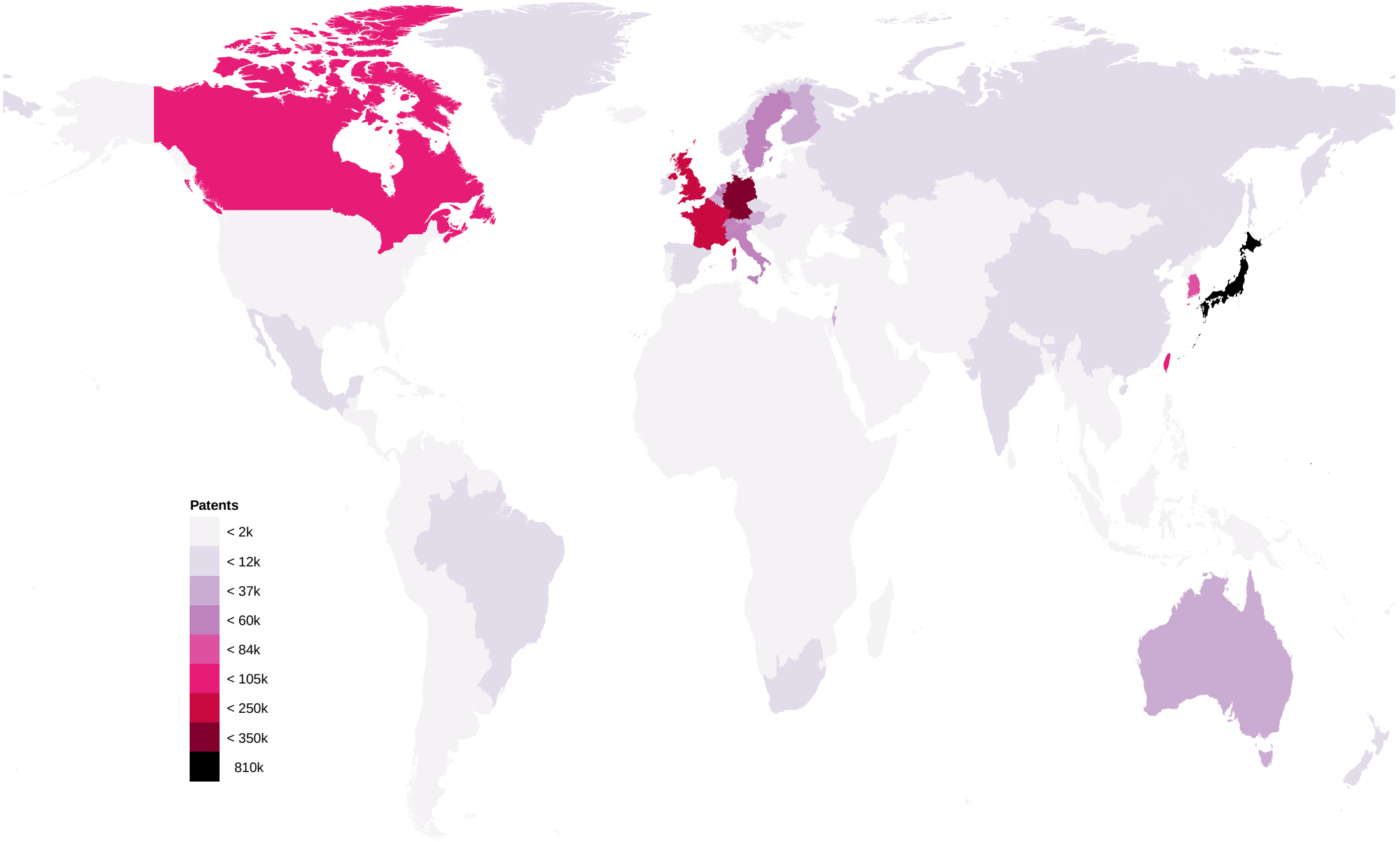


Figure 20: Location of inventors outside the USA with patents registered at the USPTO. The darker a country, the more patents.

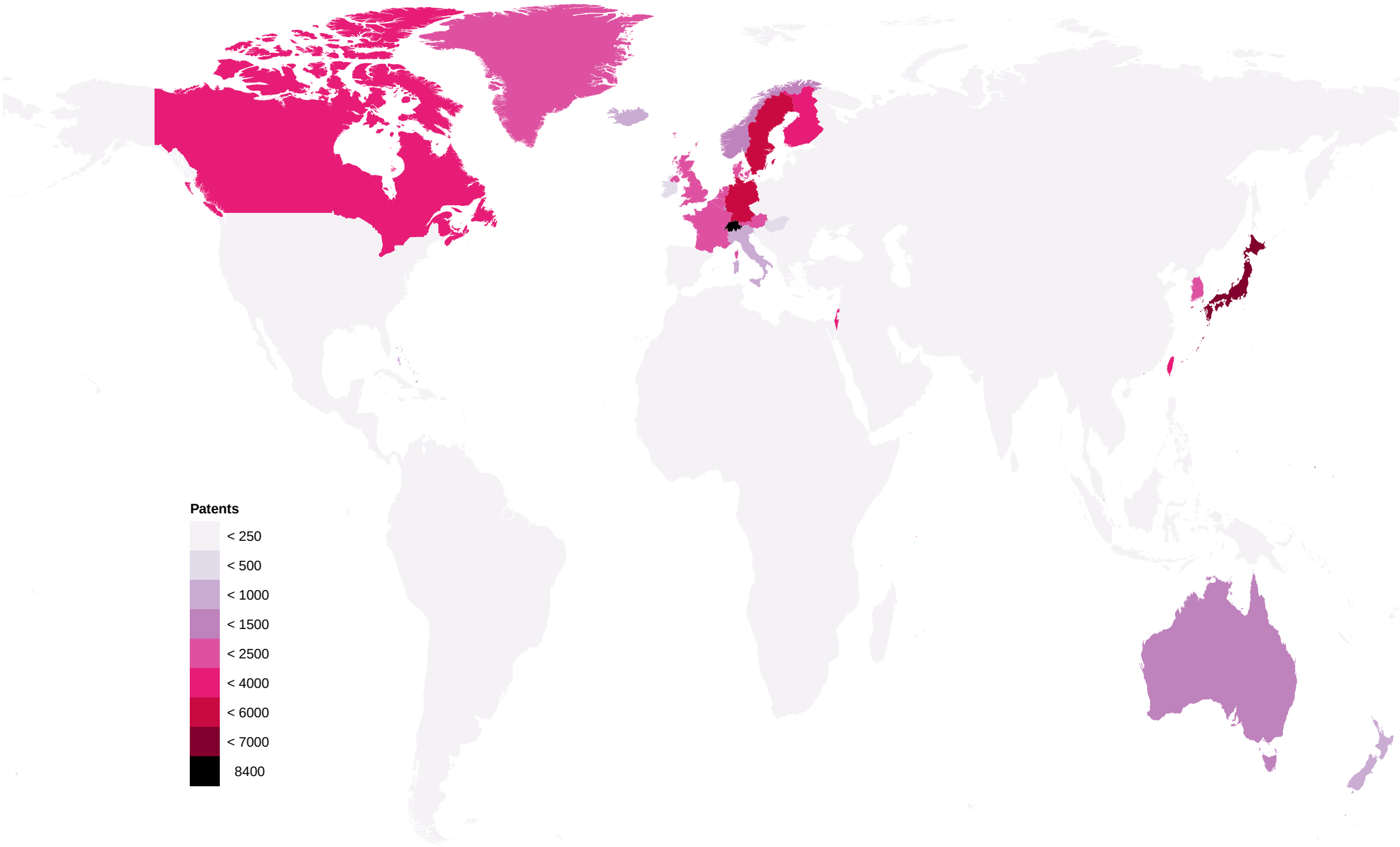


Figure 21: Location of inventors outside the USA with patents registered at the USPTO. The darker a country, the more patents per 1m inhabitants.

3.1.9.3 *By US State*

Figures 22 and 23 show the same data as in Figure 20 but only for the mainland of the USA (neglecting Hawaii and Alaska). The data is grouped by the states. The darker a state, the more patents its inventors have. Black shows the state with the most patents. Figure 23 normalizes the number of patents per state by showing the number of patents per state per 100 000 inhabitants. Since the USA is responsible for more than 2.7 million patents, states with less than 28 000 patents — equals approximately 1% or less — are not shown. In the second figure, states with less than 250 patents per 100 000 are also not shown. In Figure 21 the normalization factor was one million inhabitants and can therefore not be directly compared to Figure 23. Choosing a lower normalization factor for the world map was impossible due to the lack of patents for most countries. Choosing a higher normalization factor for the US map was impractical because eight states have a population of less than one million.

In Figure 22 California has more patents than the next four states combined. Except for Colorado and Arizona, all the states in the Mountain West have less than 28 000 patents. A cluster is seen to set up around the five Great Lakes of Northern America. This is intuitive as those areas have a high population density. California, Texas and New York are the three most populated states in the US. Looking at the normalized data in Figure 23, the clusters remain. Though the order of the states changes. Not one of the states with a high total number of patents improves their rank in the normalized dataset. Delaware, sixth fewest populated state, has the most patents per 100 000 inhabitants, California is only in the fourth highest bracket. Idaho is responsible for the biggest change, as it now ranks at the fifth position instead of the 26th.

This concludes the section about the USPTO. In the next section, MAREC and its repository of the USPTO, provided by the IRF, are presented.

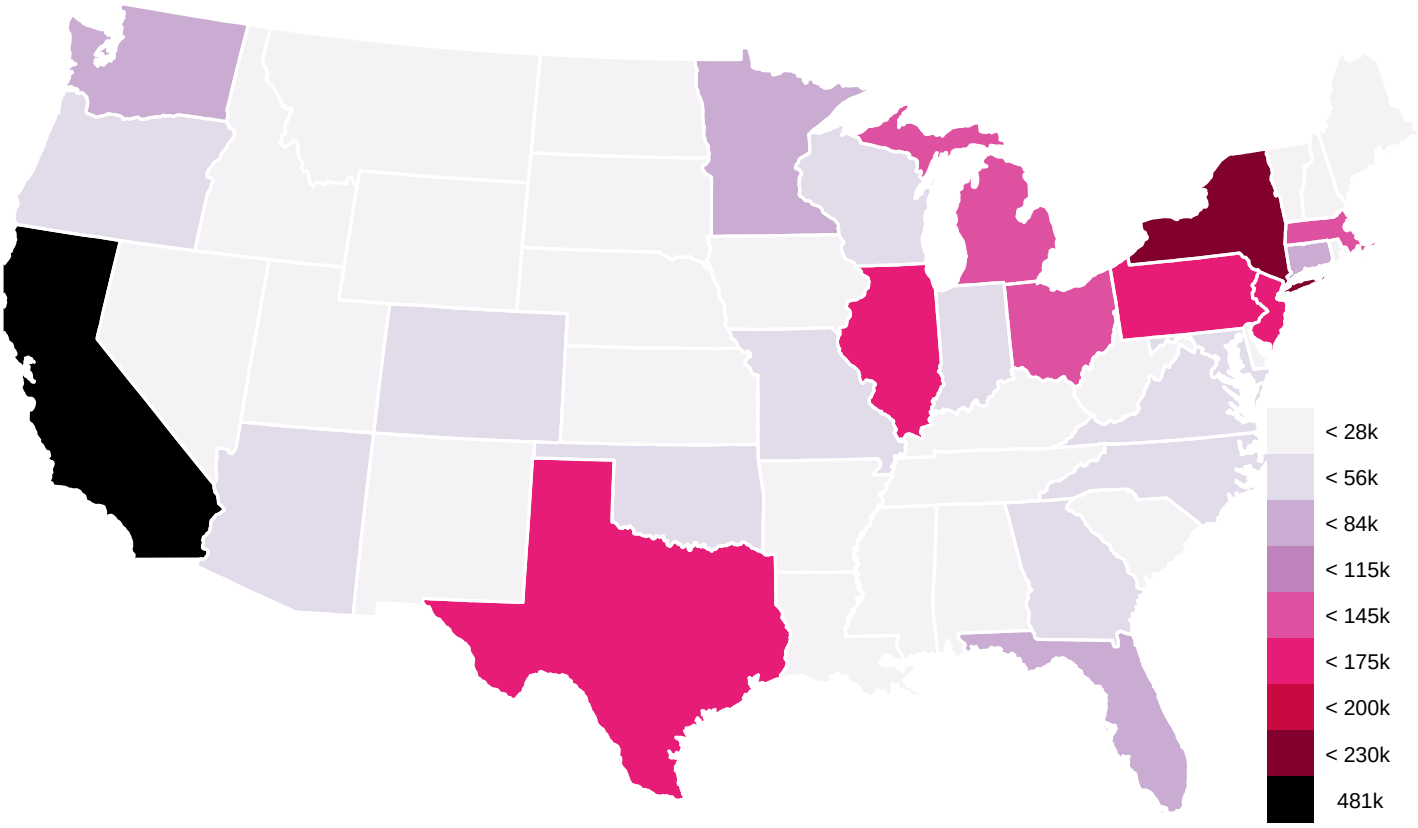


Figure 22: Location of USA-mainland based inventors of all registered USPTO patents. The darker the state, the more patents. Black states have the most patents.

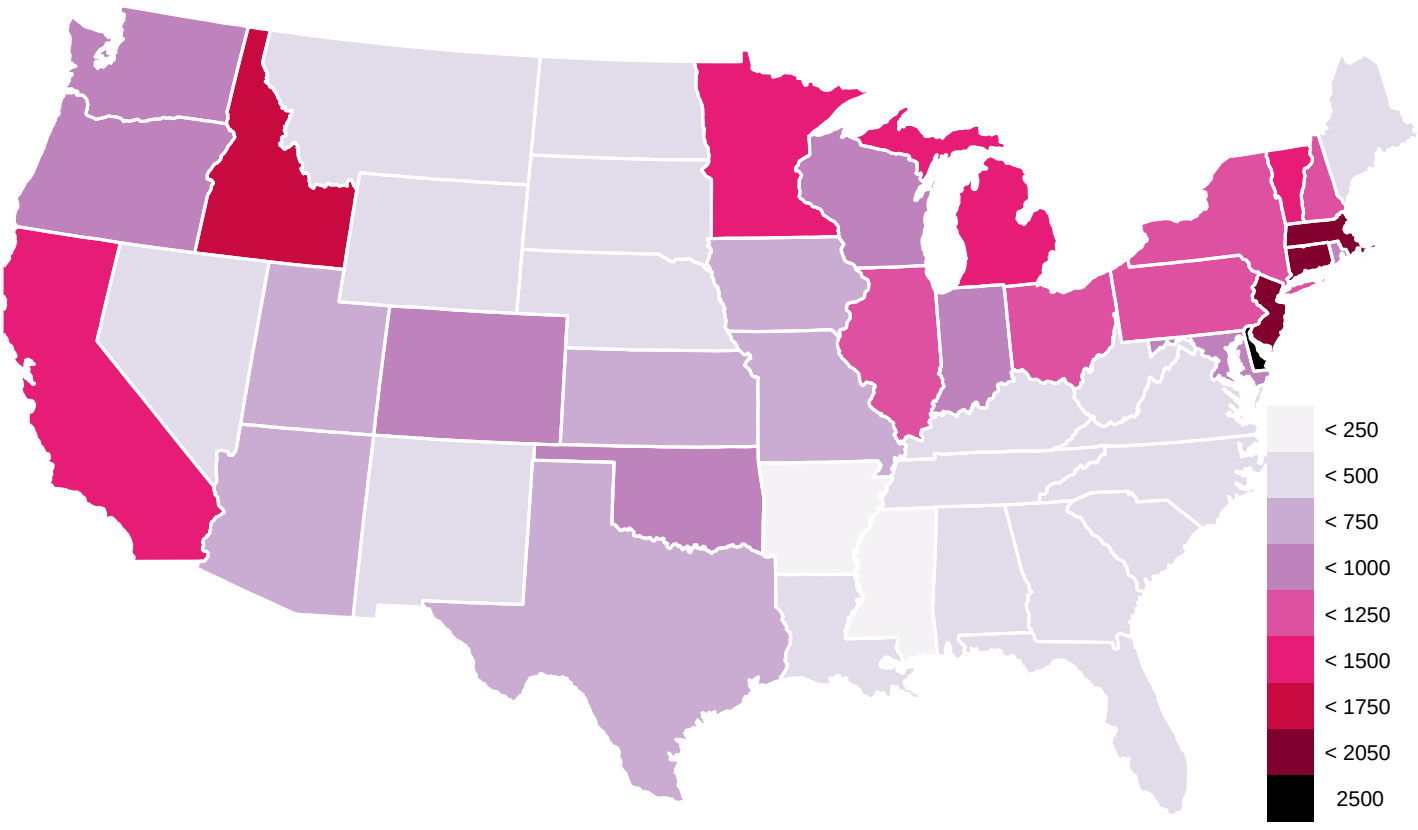


Figure 23: Location of USA-mainland based inventors of all registered USPTO patents. The darker the state, the more patents per 100 000 inhabitants. Black states have the most patents.

3.2 PATENTS AND MAREC

MAREC is a patent repository provided by the IRF[9]. The collection contains patent documents in 19 different languages, the majority being English, French and German. Patent documents are transformed into a highly specialized XML structure and certain metadata is added. Amongst other things the IPC categories, stemming from the USPTO, are added in an XML tag. Special structures from the raw patent document are transformed to XML. For example, if the description section contains a table, the MAREC XML structure contains a tag “table", which is not to be found in the raw patent document.

The full name of all subclasses mentioned in this thesis are shown in Appendix A.2.

For this thesis, only the repository of the USPTO from MAREC is used.

3.2.1 Patent Structure

A patent at the USPTO consists of twelve different sections, some are mandatory whereas others can be added to the initial patent application to clear up uncertainties (e.g. drawings to illustrate the invention more clearly). Drawings are handed in separately, though the description of the drawings has to be included in the patent application.

mandatory	Detailed description of the invention
	A claim or claims
	Abstract of the disclosure
	Title of the invention
	Reference to a “Sequence Listing" or a table
	Background of the invention
	Brief summary of the invention
optional	Description of the several views of the drawing
	Names of parties to a joint research agreement
	Sequence listing
	Cross reference to related applications
	Statement of federally sponsored research

[9] `http://ir-facility.org` — visited on 22 Aug, 2011

I use only the first three sections:

abstract The abstract, which should have a length of no more than 150 tokens according to the Manual of Patent Examining Procedure (MPEP)[10], serves the purpose of giving a concise summary of the proposed invention provided in the other two sections. It should mention the title of the invention, describe the technical field it belongs to and be written so that a layman can understand it.

description In the description section the actual invention is disclosed in such a way that a person working in the field covered by the patent should be able to implement/use the patent. The description section should start with a specification of the field. Technical terms should not be used unless absolutely necessary and should be easily understandable.

claims The claims in a patent are the legal part of the patent. They define the scope of the invention and claim the extent of protection conferred by the patent. The claims section should be understandable even when isolated, i.e. without the description available. Since the terminology is touched by legal consultants, the wording is not comparable to regular English or the description section. One of the rules in the MPEP states that a single claim has to be phrased with a single sentence. This leads to tricks circumventing the rule — this is mostly done by using ';' to concatenate several sentences, which leads to sentences normally not present in regular English text.

3.2.2 Length of Documents

One thing to note, the statistics come from my feature extraction algorithm. Words like "a" or "or" are removed. Therefore it is absolutely possible that the rules of the USPTO, listed in MPEP, might not be fully portrayed by the following statistics. The analysis is done on the entire MAREC corpus, consisting of approximately 3.5 million patent documents.

[10] http://www.uspto.gov/web/offices/pac/mpep/documents/0600_608_01_b.htm#sect608.01b – visited on 22 Aug, 2011

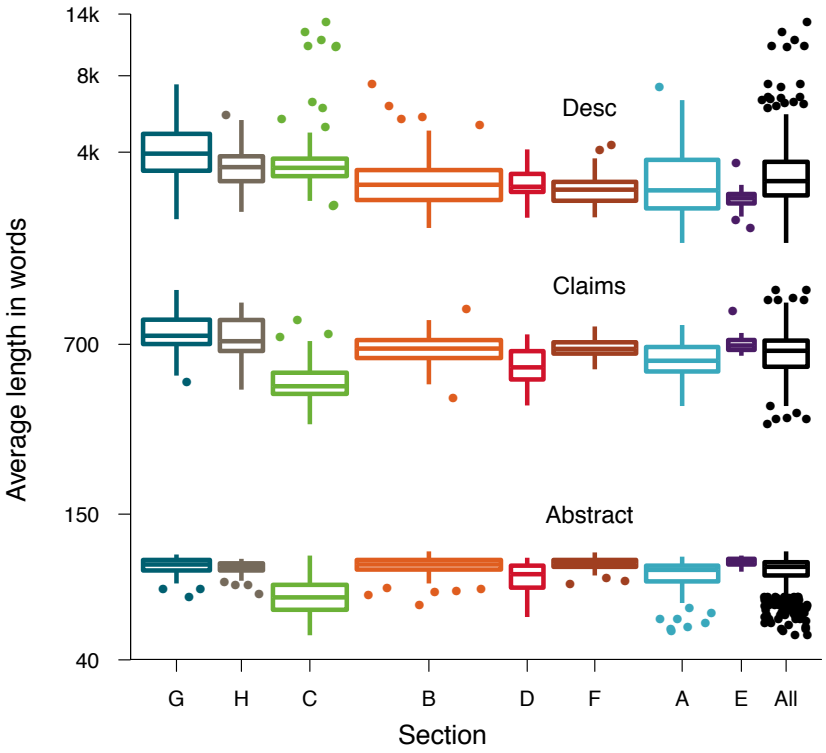


Figure 24: Average length of the three patent document sections. In descending order by sum of sections.

Figure 24 shows the average length of the abstract, description and claims for each subclass grouped by IPC sections. The x-axis is in descending order by the sum of the medians of the three sections. On average, the description section of patents is roughly 3890 tokens long. The abstract is the shortest with 93 tokens. It does not reach 150 tokens because of the removed words. The shortest abstract contains only 52 tokens on average. Claims are interesting as they often have repeating parts, but still only contribute around 700 tokens to the whole patent application. No rule limiting the length of the claims section is currently in place at the USPTO.

In comparison, the Reuters corpus (which consists of a large collection of Reuters News stories[11]) has an average document length of approximately 700 words. This is about the same length as the claims section of a patent document.

3.2.3 Stages of a Patent Document

Patent applications have different stages. At all times, a patent has a current status representing the current stage the patent is in. This status is named "Kind Code". Table 9 shows the most

[11] `http://trec.nist.gov/data/reuters/reuters.html` – visited on 22 Aug, 2011

	Kind Code	Kind of document
A	A	Utility Patent Grant issued prior to January 2, 2001.
	A1	Utility Patent Application published on or after January 2, 2001
	A2	Second or subsequent publication of a Utility Patent Application
	A9	Correction published Utility Patent Application
B	Bn	Reexamination Certificate issued prior to January 2, 2001. NOTE: "n" represents a value 1 through 9.
	B1	Utility Patent Grant (with pre-grant publication)
	B2	Utility Patent Grant (without pre-grant publication)

Table 9: Selection of stages of a patent application.

important and most often used codes, therefore showing only kind codes A and B. The full table of all possible kind codes is listed in Section A.4. The kind codes are taken from the homepage of the USPTO[12].

In MAREC a patent possesses a single directory. If a patent has more than one stage, more documents are added to the directory, all with the same name except for the kind code. For this thesis the directory is merged and seen as a single patent, the first section found in the patent documents is used. Such a merging occurs in less than 0.1% of the patents.

3.2.4 Patent Language

The language used in patents is different to regular text written in books or articles. Atkinson [2] tries to explain why this is the case and bases her reasoning on the introspection of the language used in patents rather than using corpus linguistic studies. At the time of this thesis, no extensive analysis of the patent corpus and its speech was found.

According to Atkinson one of the reasons is that a patent has to hold up in court. If the invention is described unclearly or ambiguously it can result in rejection during prosecution. This drives patent attorneys to achieve clarity that cannot be misunderstood, leading to the use of obtuse linguistic constructions. This becomes very apparent in the claims section of a patent.

[12] `http://www.uspto.gov/patents/process/search/authority/kindcode.jsp` – visited on 22 Aug, 2011

From this stems the term ”Patentese". It references the legal jargon used in patents, which should make the patent impenetrable.

Even the EPO search authority has acknowledged this problem:

> *Grammatical constructions that would be unthinkable in everyday speech or writing are used routinely in patentese. Patentese has words which do not even exist in ordinary languages. Furthermore patentese exists in every conceivable natural language version.*
> — EPO search authority [34]

This makes it hard to apply research done in text categorization on regular text.

Additionally the guidelines on how to formulate and compose a patent application suggest to put the legal content of the patent application into the claims section. Due to this guideline most of the claims sections are difficult to read for humans. Unfortunately it is not just hard for humans but also for computers and Natural Language Processing (NLP) frameworks as shown by Verberne et al. [45], who specifically analyzed the claims section. They found the sentences in the claims section to be longer on average. The average length is 53 and the median sentence length is 22 words, indicating a significant number of outliers with more than 53 words. The claims section has a slightly higher English vocabulary coverage (95.9%) compared to a regular English corpus.

Some words like “wherein", “said" or “comprising" are often used and normally not in the way they occur in a regular English corpus. This does not seem to be a problem for the classification process because such features are removed. Verberne reports of a lower coverage of multi words, which is not a problem for this thesis because my feature extraction algorithm does not extract or recognize multi words at all.

3.3 GENERAL CORPUS STATISTICS

Table 10 shows the IPC distribution, courtesy of the World Intellectual Property Organization (WIPO)[13].

The corpus used for the experiments comes from the repository of the USPTO, which is contained in the repository of MAREC. The corpus contains 3 354 949 unique patent documents, all the duplicates have been removed. For the experiments smaller subsets are used due to performance issues. How the subsets are generated

[13] http://www.wipo.int/classifications/ipc/en/ITsupport/Version20110101/transformations/stats.html – visited on 22 Aug, 2011

Sections		Class	Sub class	Group
B	Performing Operations; Transporting	37	166	16 661
F	Mechanical Engineering; Lighting; Heating; Weapons; Blasting	18	95	8439
C	Chemistry; Metallurgy	21	86	14 470
A	Human Necessities	16	84	8470
G	Physics	14	80	7611
H	Electricity	6	50	8055
D	Textiles; Paper	9	39	2961
E	Fixed Constructions	8	31	3218
Total		129	631	69 885

Table 10: IPC distribution ordered by the number of subclasses per section.

Hierarchy Level	Total	Cvrg
		%
Section	8	100
Class	124	96
Subclass	628	99

Table 11: Coverage (Cvrg) of the levels by the USPTO corpus.

is explained in Section 5.1. Table 11 shows the coverage of the sections, classes and subclasses by the corpus. The sections are fully present in the corpus as well as almost all of the classes and subclasses.

Whenever there are not enough documents for a subclass, the subclass is removed from the experiments. The threshold is set to 100 documents, which leads to a removal of 48 subclasses. This is necessary to execute the text classification task. With less than 100 (out of over 3m) documents a classifier is unable to create a hyperplane to distinguish the documents. Since the thesis categorizes on a subclass level, all the statistics and analysis will be done on a subclass level.

Figure 25 shows the number of documents per section. It is quite visible how skewed the dataset is on a section level. On a class level, as seen in Figure 26, the skewness is worse. There are a few outliers with more than 400 000 and less than 100 documents.

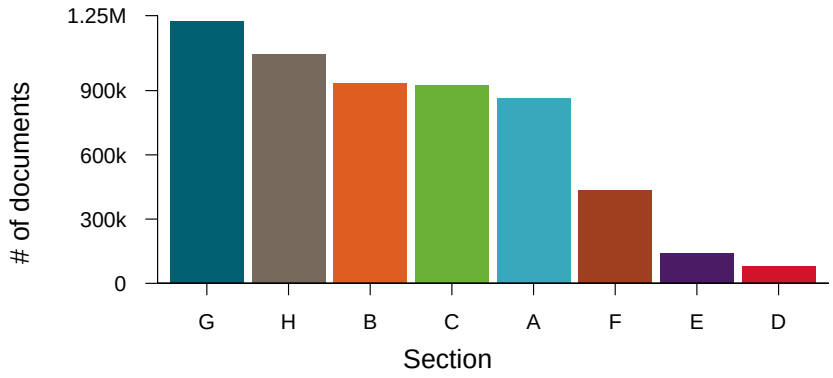


Figure 25: Number of documents per section.

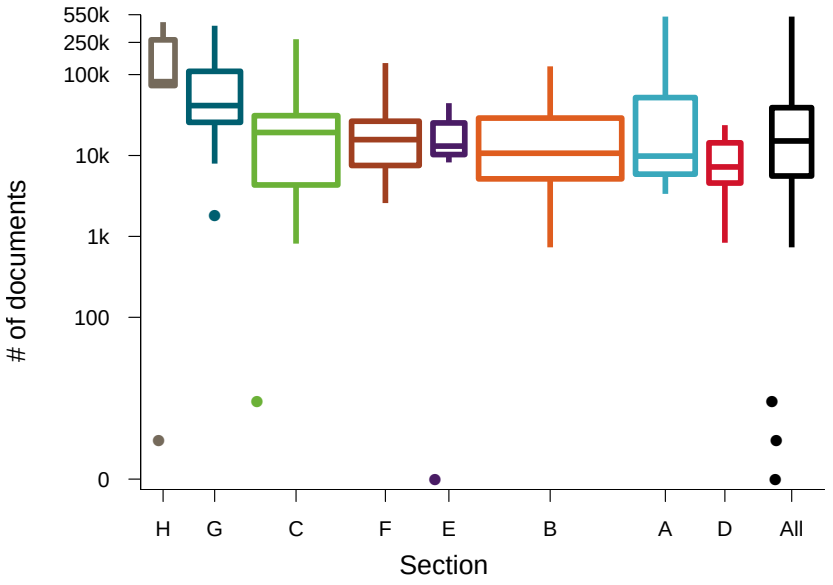


Figure 26: Number of documents per class.

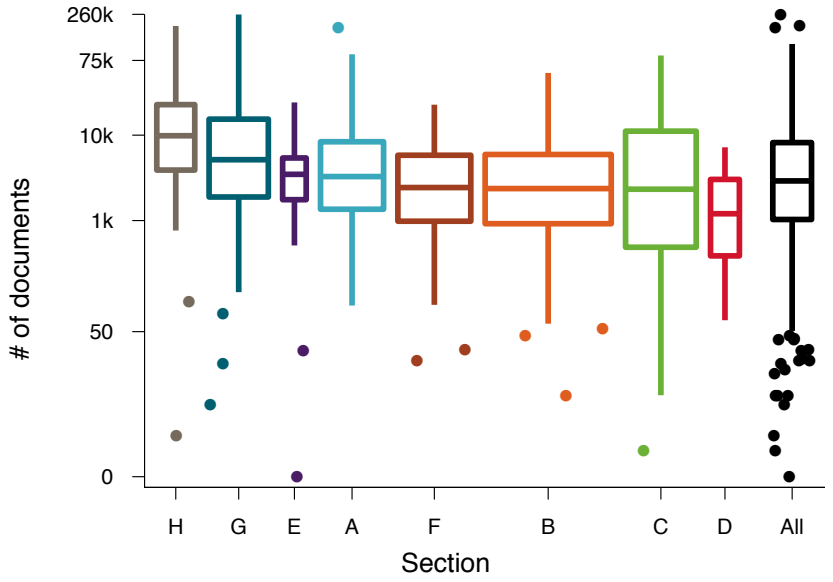


Figure 27: Number of documents per subclass.

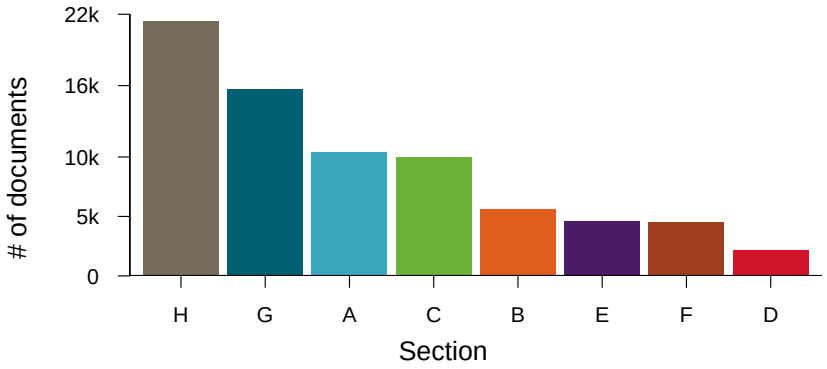


Figure 28: Avg. number of documents per subclass, grouped by section.

The number of documents per subclass is shown in Figure 27. It peaks at roughly 250 000 documents for a single subclass.

There are subclasses with less than ten documents, which leads to a ratio of 1:26 000. The figure shows how just a few subclasses are responsible for most of the corpus. The five largest subclasses (0.7% of all the available subclasses) are responsible for approximately 26% of all the documents. Most subclasses are around or below the 15 000 documents mark.

Figure 28 shows the average number of documents per subclass, grouped by their section. The imbalance of the dataset is quite visible. Subclasses in section H have on average almost five times more documents than subclasses from sections E and F. Since the classification task is split into a binary classification task, as described in Section 2.3, there is a discrepancy when training a subclass in sections D, E or F simply because there are so few positive documents. Future work should focus on how to handle this kind of imbalance.

3.4 UNIQUE FEATURES

A unique feature only occurs in a single subclass, the number of occurrences within the subclass are irrelevant to the definition.

For any classification task the number of unique features is considered to be important. For example, in a binary classification problem with six unique features, it is easier to distinguish a document when documents of category A only use the features $\{1, 2, 3\}$ and documents of category B the remaining features $\{4, 5, 6\}$. This makes the classification problem linearly separable. Whether or not the patent corpus provides enough unique features, which are evenly spread among the subclasses, will be analyzed and visualized in this section.

Figure 29 shows the total number of unique features in a subclass, grouped by sections. As shown in the figure the highest

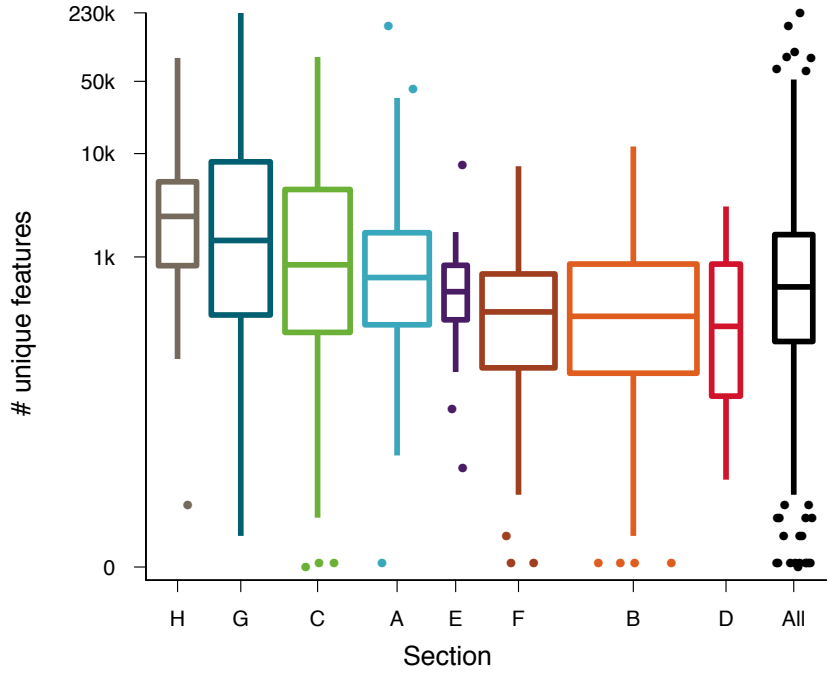


Figure 29: Total number of unique features per subclass.

subclass in section G named G10H. It is a subclass that has a wide range of groups, all of which are in electrophonic musical instruments, which as can be seen, provides many unique words not used in other subclasses. Additionally it is also one of the largest subclasses with 95 000 documents. Except for two subclasses, all subclasses have less than 100 000 unique features.

When those numbers are compared to the total number of features within a subclass, the picture changes. Figure 30 shows that most subclasses are below 5% and that section G has the most subclasses over 5%. This however does not show how often a unique feature occurs. If a unique feature does not occur often enough, the chances of it being removed by the feature selection algorithm are high. If it would not be removed by the feature selection algorithm, overfitting is possible because a unique feature, occurring only in one or two documents, would automatically identify the documents belonging to a specific class. While this increases F1 during the training phase, it might lead to a reduction in F1 once the trained model is deployed because of overfitting.

This is why Figure 31 compares the length of the unique features (UF) to the total length of the subclass on the full corpus. The length of the unique features is defined by the sum of the number of occurrences of all the unique features in the subclass. According to the figure unique features do not occur often. This holds for all subclasses. Unique features occur on average 1.8 times, reducing the chance of occurring in more than one document. This would lead to the previously mentioned overfitting.

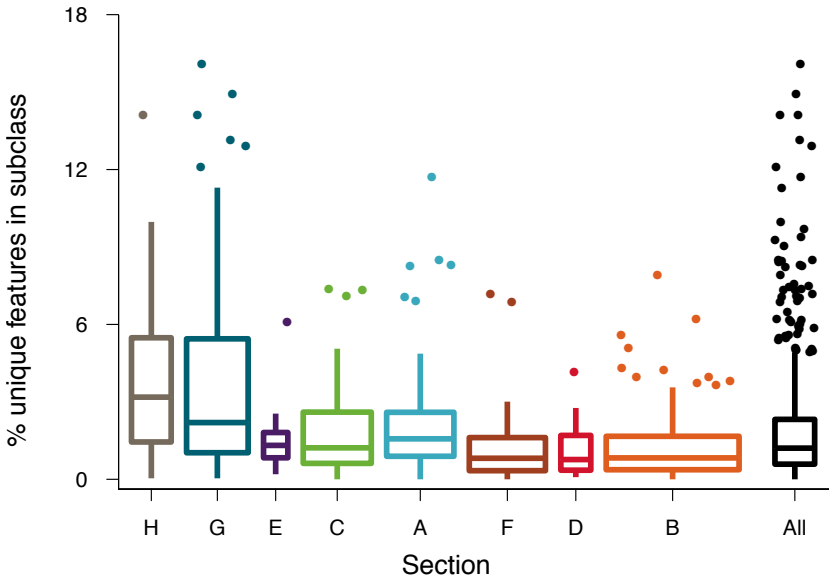


Figure 30: Percentage of unique features in a subclass.

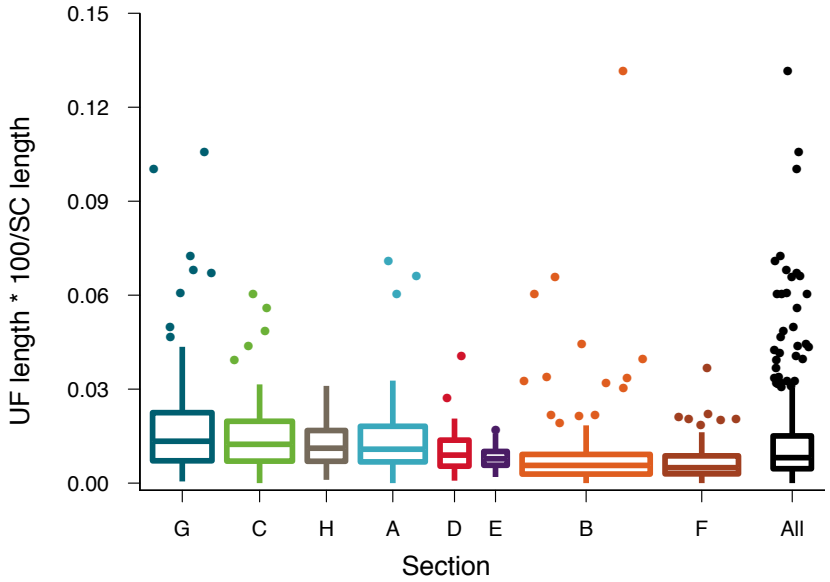


Figure 31: Length of the unique features divided by total length of subclass.

As for most subclasses the length of all the unique features is only around 0.02%. In words this means one hits a unique feature roughly every 6000th word. Using only unique features is out of the question as some documents do not have a single unique feature. This automatically translates into a classifier producing only negative predictions and a recall value close to zero.

3.5 CONNECTIVITY AMONG IPC CLASSES

This section explains the connectivity of subclasses. Connectivity means that a subclass is mentioned with another subclass in the same document. The more often a subclass is mentioned with another subclass, the stronger the bond between them is and the more features are shared between them. Sharing many features and documents creates hyperplanes for the classifiers that are too similar. With such a similarity both binary classifiers of the two subclasses might classify a document positively, although it belongs to only one of the two subclasses.

Number of assigned Categories	**Documents**	
1	1 926 000	~3 150 000 **94%**
2	873 000	
3	351 000	
4	127 000	
5	46 000	
6	18 000	
7	7500	
8	2900	
9	1100	
10	480	
11	330	
12	150	
13	83	
14	41	
15	7	20 **0.0006%**
16	4	
17	2	
18	6	
20	1	

Table 12: Number of categories per document.

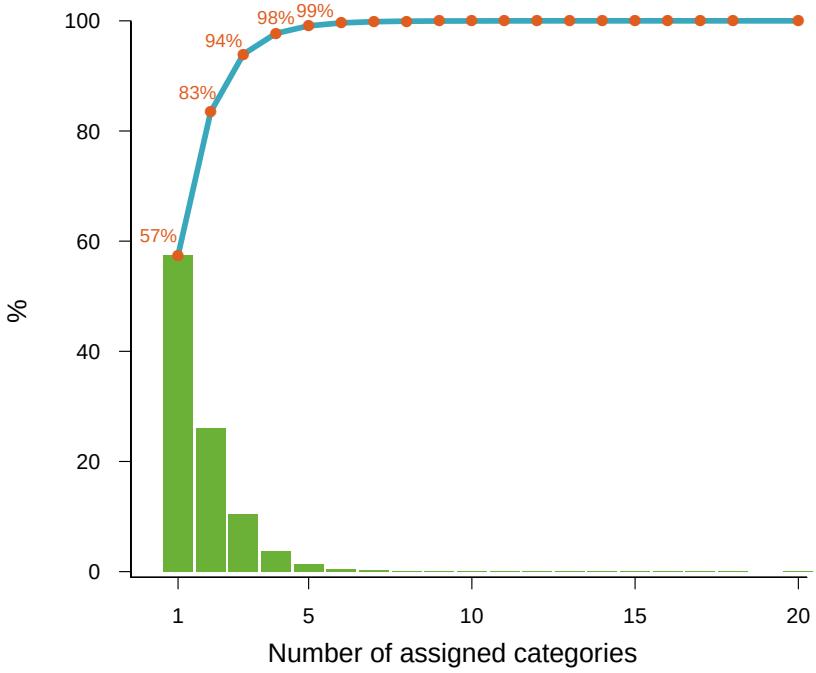


Figure 32: Percentage of documents grouped into “assigned categories per document".

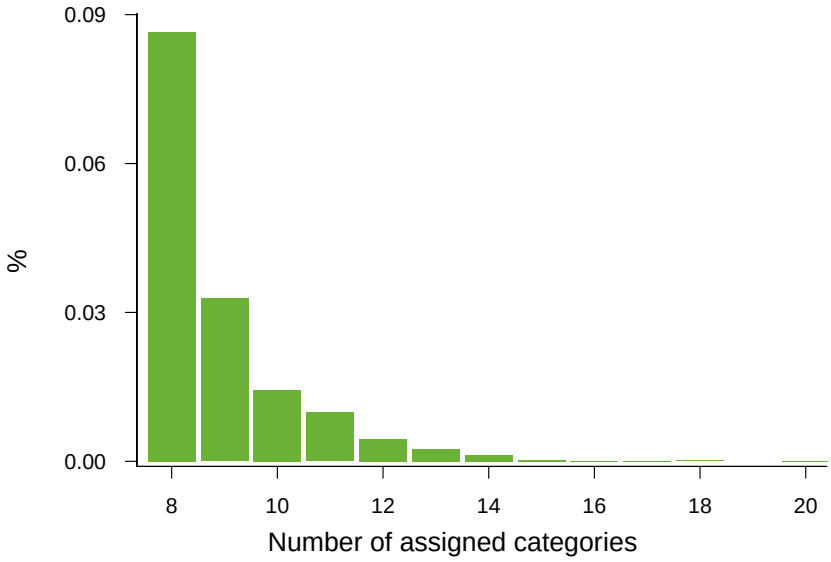


Figure 33: Eight or more categories only.

3.5.1 Categories per Document

As mentioned before, patents are multi-label documents. This means a patent can have more than one category. Table 12 shows the number of categories per document, Figure 32 visualizes the percentage of documents grouped by the number of assigned categories. The mean is 1.8 categories per document, minimum is one, peak is at 20. Not even every second document is connected to at least two categories, putting the median at one.

Approximately 2.8 million documents have one or two categories, translating into ~83% of approximately 3.4 million documents having hardly any connections.

On the other end of the spectrum are those documents that have between 15 and 20 categories assigned to them. This obviously does not occur often, as those six bins are responsible for only 20 documents (~0.0006%). Those 20 documents can be removed to improve the classification result. Going even further, another possibility is to remove all highly connected documents. For example, the number of documents assigned to seven or more categories is exactly 12 675 (or ~0.38%), visualized in Figure 33. The removal of so few documents has little impact on macro averaged metric numbers (e.g. precision or recall).

3.5.2 Number of Connections per Subclass

To see if there might be a pattern in the connectivity between subclasses from a section (e.g. subclasses from section A having fewer connections than subclasses from section C), Figure 34 is presented. As one can see from the Interquartile Range (IQR), most sections have a wide range of differently connected subclasses. The peak is at 552 (out of 628) subclasses, leading to being connected to ~88% of the subclasses. There are also subclasses with just a few connections. Such a subclass is always a small subclass with less than 50 documents assigned to it. Most notable are subclasses ending in 99Z, for example G99Z (name: "Subject

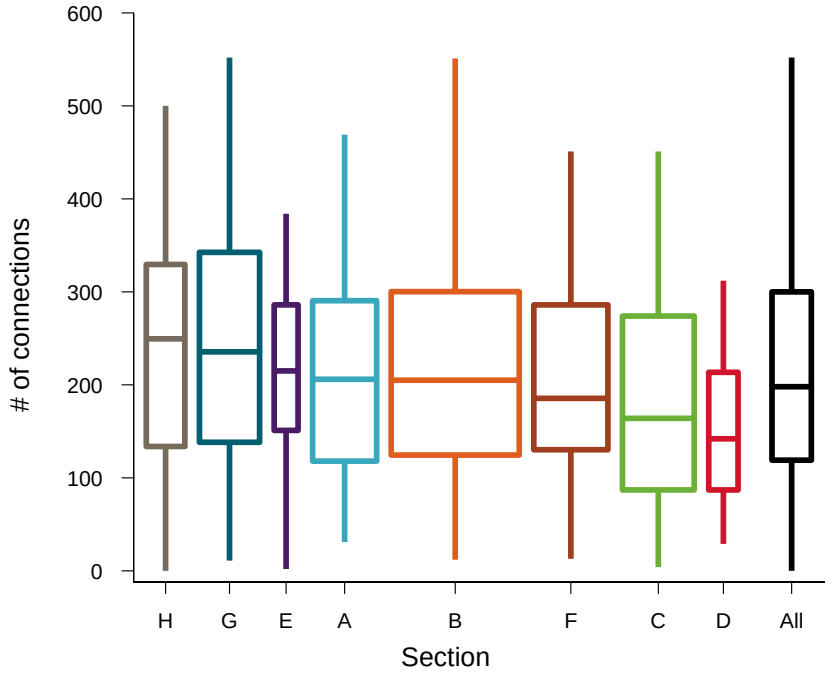


Figure 34: How connected the various subclasses are.

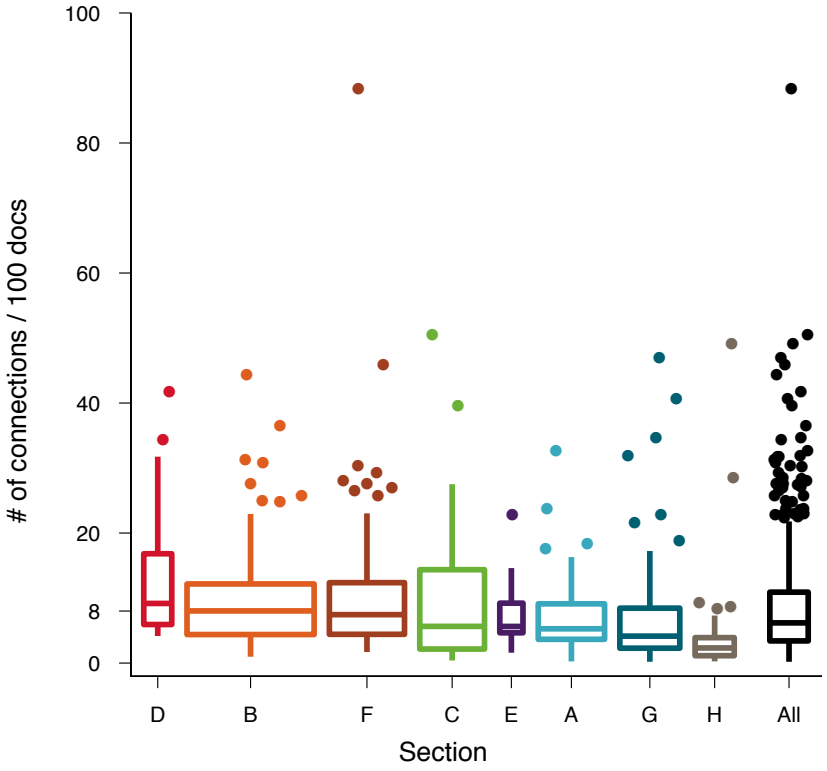


Figure 35: Subclass connections normalized to 100 documents.

matter not otherwise provided for in this section"). It is a filler subclass for patent documents that do not seem to fit into any other pre-defined subclass. Such subclasses have less than ten documents but occur in every section.

One of the highly connected subclasses is G01N. This is worth mentioning because it is also one of the largest subclasses, containing approximately 245 000 documents, making it responsible for 23% of the connections. If such a subclass receives special treatment in both feature selection and classification, an increase of the classification performance might be feasible.

From the five most connected subclasses three come from section B, from the ten most connected subclasses 6 come from section B. Because section B has the most subclasses, it is easily the most connected section. But due to the large number of subclasses in the section, its median is not as high, ranking it at only the fifth position.

The statistics for the entire corpus reveals an even distribution. There are no outliers, the IQR is between 120 and 300 connections per subclass. The median is 198, the mean is 212.

There is not a single outlier in either a single section or the entire corpus.

Figure 35 shows the same data but normalized to 100 documents per subclass. It shows only those subclasses with 100 or more documents. Except for one outlier in section F, all subclasses

have less than 50 connections per 100 documents. All medians are between five and ten connections. It also shows significantly more outliers than the non-normalized dataset, though the medians of all eight sections are within 10% of each other, similar to the non-normalized dataset shown in Figure 34.

3.5.3 Distribution of Connectivity between Subclasses

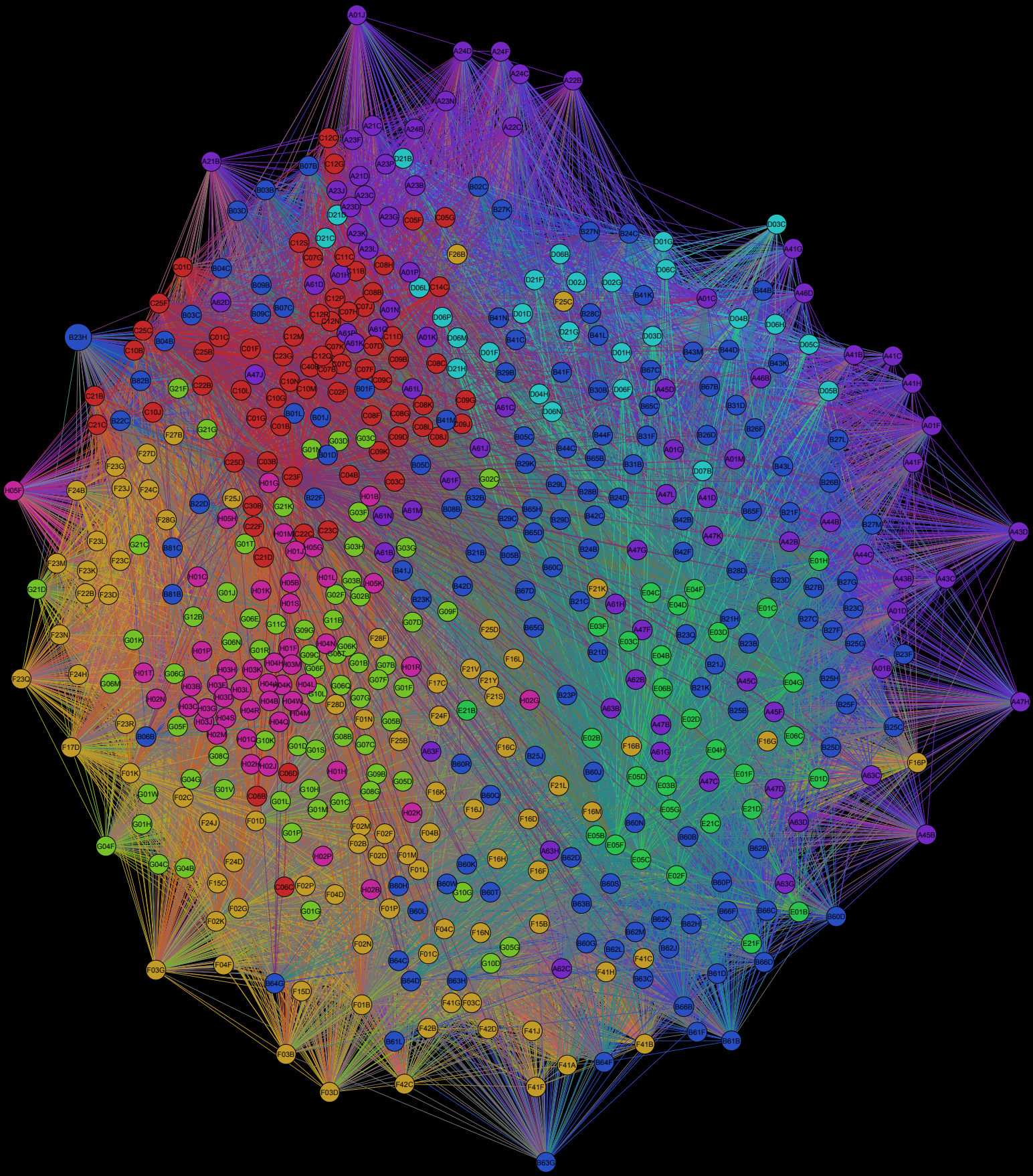
Figure 36 shows the connections of each subclass with every other subclass. The coloring indicates the section. The figure was created by using a layout. The network was imported as an undirected graph and the Fruchterman–Reingold algorithm (a force-based layout algorithm for drawing graphs by Fruchterman and Reingold [17]) was applied for ten seconds. The closer two subclasses are (i.e. the more connections between those two subclasses exist), the closer they are in the chart.

What becomes immediately apparent is the clustering of subclasses belonging to the same section. Such a clustering appears when the relationship between the subclasses of a section is stronger than the relationship that these subclasses form to subclasses from other sections.

Subclasses from section H (pink) are, except for a few outliers, all in the same spot. There is little overlapping because the Fruchterman–Reingold algorithm tries to minimize it. Almost all of the subclasses from section G (bright green; to the left) are next to subclasses from section H. This indicates an extraordinarily strong bond between the two sections and will be responsible for many shared tokens.

Another clustering happens with subclasses from section C (red). Some of those subclasses are bound to a few specific subclasses from section A (violet) — though most of the subclasses from section A have no relationship to subclasses from section C. In these cases the subclasses are almost always connected to subclasses from section A. Another cluster is section D (turquoise). While it is not as centralized as sections C and H, it bears almost no connections to any subclass from sections F, G and H. It is heavily connected to section B though.

The conclusion this figure delivers is a heavily connected corpus and subclasses staying within their section, shielded off from other sections by subclasses from their own section. Additionally each section has a few outliers breaking the overall picture of the section, for example subclass F16P (yellow; right).



Connections between each subclass. Color indicates sections. The Fruchterman–Reingold algorithm was applied for 20 seconds. Explanation can be found in the text.

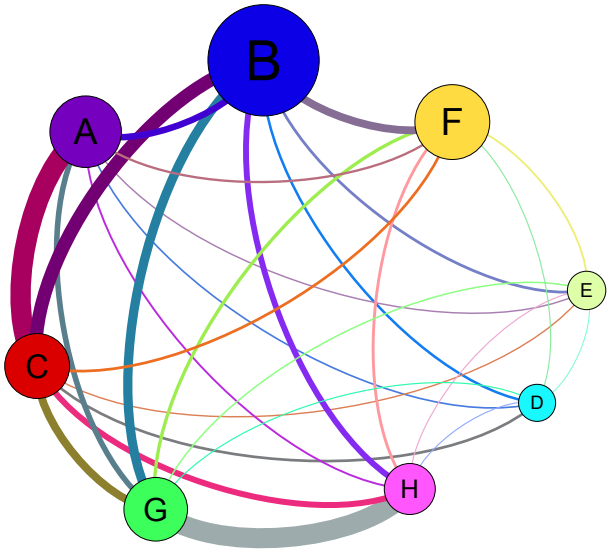


Figure 37: Connectivity of subclasses grouped by their sections.

3.5.4 Connectivity of Sections

Figure 37 shows the same data — connectivity between subclasses — but grouped by section, leading to a more compact view of the data. The node colors are the same as for Figure 36. Because it is an undirected graph, edge colors have no indication whatsoever. The thicker the edge the stronger the connection between the two nodes. What is not visible in the previous figure is that the section chart is a complete K_8 graph. Figure 37 confirms the strong connection between section A and C as well as between G and H.

Figure 38 shows the same data as Figure 37 but normalized to its relative importance. The y-axis shows eight groups, where each group has one section as its head, shown to the right. The x-axis shows the percent of documents that are connected between one section and the head. The green bar shows the section, which is responsible for most of the connections and the orange bar shows the section itself. The absence of the green bar in a group means the section of the group has more connections to itself than any other section. The percentage is normalized by the number of documents assigned to the head of the group, there is a larger connection between sections E and B than B and E. Section B has hundreds of thousands of documents whereas section E only has a few thousand, hence making the few hundred connections it has to section B more important. This characteristic can be classified as a directed graph. Half of the sections have the most connections to themselves. In section C one third of the documents are connected to at least one other subclass from section C.

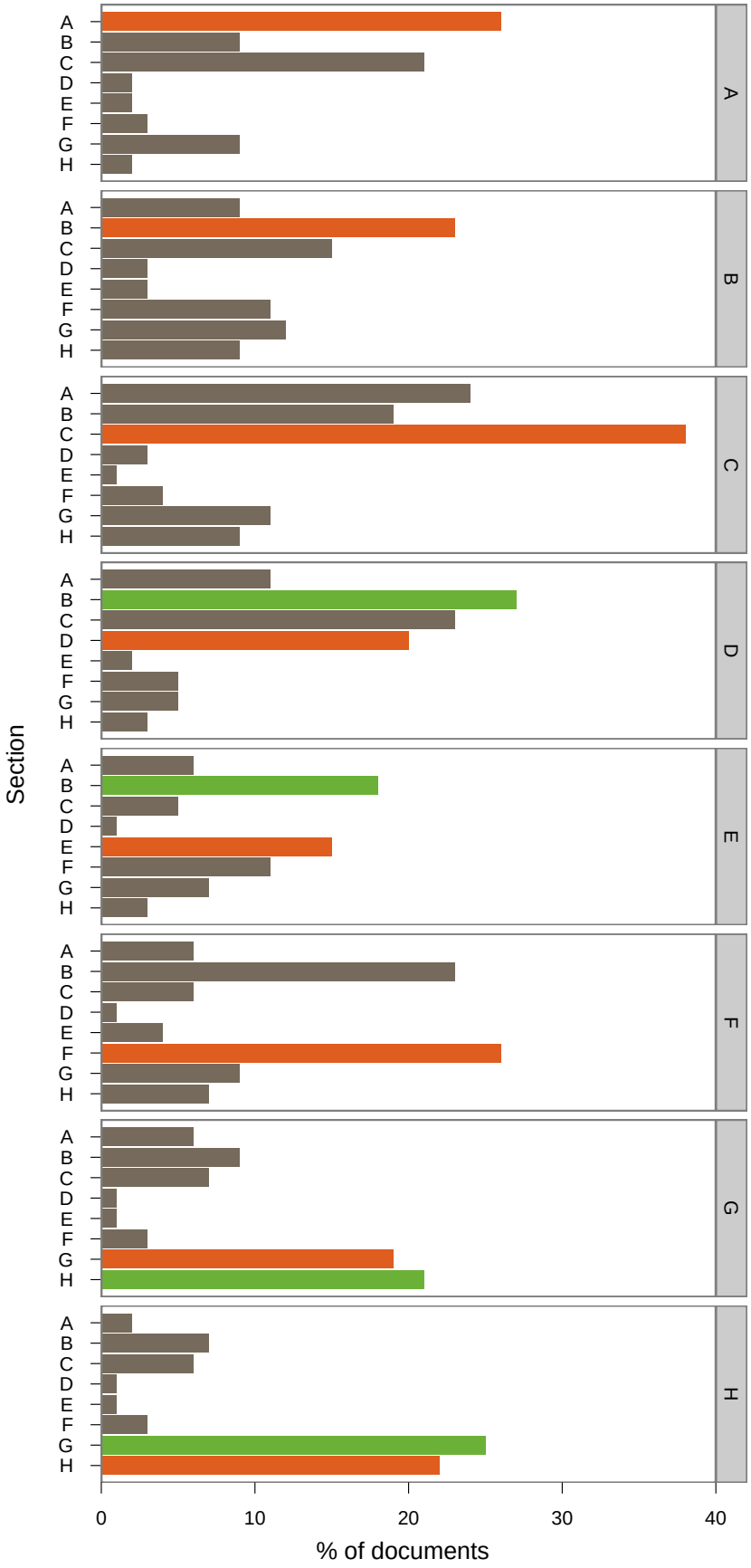


Figure 38: Connectivity of subclasses grouped by their sections. Each group uses the number of documents assigned to the section as 100%. X-axis shows the number of documents being connected in percent, y-axis shows the section groups with the head of the group to the right.

3.5.5 Connectivity between Subclasses and Sections

To see how the connectivity is distributed amongst the subclasses, Figure 39 shows a few hand-picked subclasses and how they are connected to the various sections. While the figure has significantly fewer subclasses than exist, the selected subclasses represent the other missing subclasses well. This can be easily checked because Section A.5 visualizes the connectivity for all subclasses. Subclass A01P is fascinating because it gives an almost 100% guarantee of it being connected to at least one other subclass from section A. It is also connected to section C but no connections at all to sections D, E, F, G or H.

Subclass B81C is connected to subclasses from B, G and H. This is a common sight for subclasses from section B.

A visible pattern with all subclasses seems to be that they are first and foremost connected to subclasses within their own section. Additional connections to subclasses outside their own section cannot be considered the norm. This confirms the previous two figures.

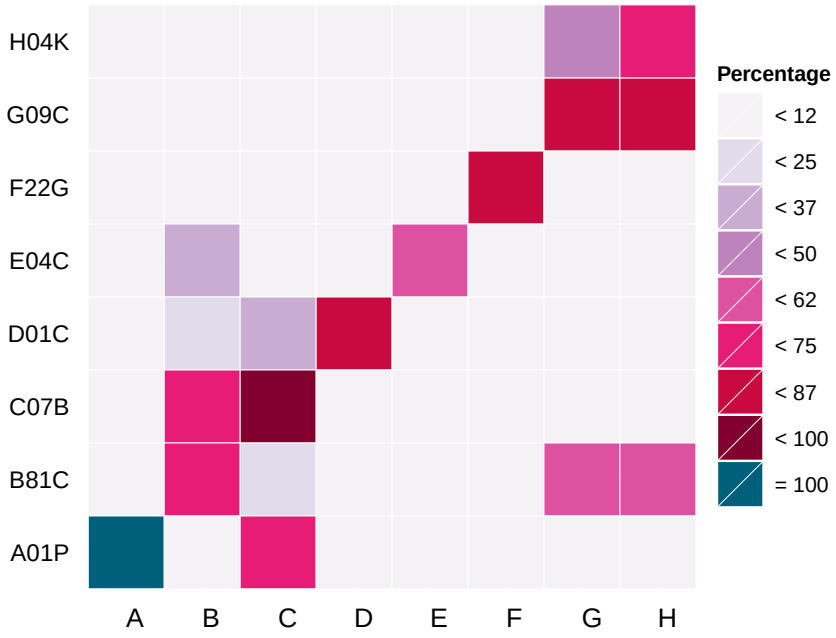


Figure 39: Connectivity for various subclasses. Connections to their own subclass are not counted. The darker the cell the more connections.

3.6 CONCLUSION

Section 3.1 shows the importance of the USPTO to my thesis. The number of patent examiners in conjunction with the number of filed patent applications shapes the priorities of the USPTO. With plenty of examiners, reclassification receives a higher priority. Since every application, which is misclassified by the USPTO, has the potential to throw off my algorithms and classifiers, reclassification is important. The number of patent examiners is determined by the annual budget, an increase in the annual budget potentially leads to an increase in examiners.

First a pre-examiner classifies an application, determining which patent examiner should process the application [21]. Therefore many of the applications in the backlog might not be correctly classified yet. Applying an automatic classification application as soon as the application is filed helps the human resources department of the USPTO as well as the management to determine which sections and groups are understaffed without relying on misclassified applications. Since fewer examiners are needed for the classification task, they are freed up and can work towards the reduction of the pendency of an application as well as the ever-growing number of pending applications.

Since 2006 the number of electronically filed patent applications has steadily increased. This results in a reduction of the number of digitalized applications leading to fewer character errors in the application that could potentially throw off my algorithms.

The origin of the patentees has an impact on the language used in the patent application. While large corporations like Toshiba or Canon have the financial means to hire English native speaking law firms to create their patent applications, individuals might not be able to afford them and simply hire translators or even worse, use machine translation services. The difference in the language has an immediate impact on my algorithms, as will be seen in the next chapter.

Sections 3.2 and 3.3 show the actual corpus used in my experiments, its characteristics and how the applications are distributed amongst the subclasses. Since they are unevenly distributed, the algorithms have to be adjusted, taking the distribution into consideration. Because patent classification is a multi-label task, Section 3.5 shows the connections between the subclasses. This can also be used as a foundation for future work by modeling the classifiers closer to the connectivity of the corpus.

Section 3.4 shows the large number of unique features. First, this means the task at hand has a high-dimensional feature space, requiring vast resources. Second, this characteristic not only justifies feature selection but makes it absolutely necessary.

4 FEATURE SELECTION WITH WORD ENTROPY FOR PATENTS

CONTENTS

This chapter presents the core and purpose of the thesis. Specifically the metrics and algorithms used to remove specific features from the index are explained. Additionally I explain how I incorporate the IPC structure in determining which features to keep and which to remove.

4.1 BACKGROUND

The thesis applies the theory of Montemurro and Zanette [33] on the domain of patents and tries to reduce the number of features used to train the classifier but still keeping the quality of the classification the same. All the necessary information needed to understand Montemurro's theory is explained in this section.

Advanced readers familiar with entropy, MI and IG can go directly to the specific Montemurro metric in Subsection 4.1.4.

When a feature is named as negative in the following chapters, a linguistic mindset is used. When a feature does not have a lot of semantic meaning, it is a negative feature. Furthermore it cannot be used as a keyword for distinguishing subclasses from each other. Examples are function words, such as "the", "each", ... A positive feature is the opposite, carrying a lot of information and occurring often (but not too often). Examples are "meat", "butchering", "radio transmitter". The word "the" is not particularly useful as it is the same in every subclass. The word "butchering" however does not occur in every subclass and if it does occur, it can be used as a keyword since it also carries further information (i.e. production of meat, killing, ...).

4.1.1 Entropy

Entropy was introduced by Claude E. Shannon in [42]. In general, entropy is used to measure the amount of information contained in a random variable X with possible values $\{x_1, \ldots, x_n\}$ [32] and is defined as:

$$H(p) = -\sum_{i=1}^{n} p(x_i) * \log_2(p(x_i)) \tag{4.1}$$

where

x_i	single word
X	all words
$p(x_i)$	probability of the occurrence of x_i
n	number of words in X (i.e. $\|X\|$)
$\sum_{x=1}^{n}$	sum over all possible x in X

Since the concept comes from information theory, the unit of measurement is bit (hence the logarithm to the base 2). Using any other base yields a linear scaling of the results. Since Montemurro and Zanette use the same base, for the rest of this chapter, an unadorned log should be read as log to the base 2 ($\log_2$).

The smaller the entropy $H(p)$, the easier it is to encode all the words. The smaller the value for a single item, the less random and important it is because it occurs often. This is the main hypothesis for using Montemurro's method as a feature selection filter. A simple example can illustrate the evaluation of the importance. If you transport a message, "The opposite of the trivial theorem is the deep theorem", over a network, the entropy provides an indication of how effectively you can encode the message (either on character or word level). It also shows how

the	−0.521
opposite	−0.332
of	−0.332
trivial	−0.332
theorem	−0.464
is	−0.332
deep	−0.332
H(p)	**2.46**

Table 13: Entropy for each token type for the message “The opposite of the trivial theorem is the deep theorem".

many bits are required to transport it. Table 13 shows the entropies for every token type from the message based on word level. It uses the number of occurrences for each word. For example “the" occurs 3 out of 10 times, making $p(x) = \frac{3}{10}$. Using this for $p(x) * \log_2(p(x))$ results in $p(x) * \log_2(p(x)) = -0.521$. In this specific case the entropy for the token type “the" is the lowest because it appears the most often. This is why encoding it with the shortest possible term (possible term: 1) would yield the highest results and the lowest memory requirement. Encoding the token type “trivial" would require more bits (possible term: 111) and therefore has a higher entropy because less information can be transported with a single bit. The total is positive because the sum in $H(p)$ is negated.

In NLP entropy is used (among other things) to detect token types of lesser quality in a text. Such token types are features of lesser quality and do not provide any additional information about the text. If they are kept and fed into a classifier as features, class separability would not increase (or worse: decrease). The classification task would require more resources (both time and memory) and might not perform as accurately as possible.

4.1.2 Mutual Information

MI [40] is defined in information theory as a quantity between two random variables that measures the mutual dependence of the two variables. Since it is used in information theory, the most common unit of measurement of MI is the bit, hence a logarithm to the base 2 is used.

The mathematically definition of MI is

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p_1(x) p_2(y)} \right)$$

where

X, Y	discrete random variables
$p(x,y)$	joint probability distribution function of X and Y
$p_1(x)$, $p_2(y)$	marginal probability distribution functions of X, Y.

In recent years MI is more and more used in machine learning research, especially in combination with feature selection. Most of the current research focuses on using MI to find out how relevant a feature is within a category or text. This can be seen in Peng et al. [36] and Liu [30] about feature selection and how to incorporate MI in a single metric, which can be optimized by different non-machine learning algorithms (i.e. genetic algorithms, Ant Colony System (ACO)).

4.1.3 Information Gain

IG is another popular measure in machine learning and is sometimes called Kullback-Leibler divergence [27]. It measures the difference between two distributions by using their entropies. As such it is defined as

$$IG(Y|X) = H(Y) - H(Y|X) \tag{4.2}$$

where

Y, X	discrete random variables
$H(Y)$	entropy of Y
$H(Y\|X)$	average specific conditional entropy of Y

IG is often used in decision trees to determine whether or not a split of a feature inside the tree into two features resulted in a gain of information. This is done by calculating the entropy of the original feature and substracting the entropy of the two newly split features.

4.1.4 Montemurro Metric

In [33], Montemurro and Zanette used a quasi-entropy metric to figure out the distribution of words within a text (in their article

they used books as references), trying to see if words change between chapters.

This is achieved by splitting the entire corpus into an arbitrary number of segments. Those segments are then used independently of each other, i.e. the order of the segments has no impact on the information value the metric assigns to a feature. One of the criticisms of the approach is the destruction of the coherence of the text due to the segmentation of the corpus. Such a segmentation interrupts the flow of the corpus. As I explain later, I do not apply the metric on a single document but rather concatenate the patent documents in different ways and apply the metric on the concatenated documents. Since there is no natural cohesion between individual independent patent documents, coherence in the concatenated corpus cannot be destroyed to begin with.

The number of words within a single segment is used in form of a parameter and determines the number of segments the corpus is split into. The user of the algorithm has to set it prior to the execution. From here onwards this parameter is named s.

The mathematical definition of the Montemurro metric is given by

$$\Delta I(s) = \sum_{w=1}^{K} p(w) * [\hat{H}(J|w) - H(J|w)] \tag{4.3}$$

where

J	all parts of the corpus
s	size of each equally large part
K	entire set of words
w	single word
n	number of occurrences of w in J
N	length of entire text
$\sum_{w=1}^{K}$	sums over all words
$p(w)$	given by n/N

$H(J|w)$ is the entropy of word w by summing over all the parts j contained in J. It is given by

$$H(J|w) = -\sum_{j=1}^{P} \frac{n_j}{n} * \log_2(\frac{n_j}{n}) \tag{4.4}$$

where

J, w, n	defined in Eq. 4.3
P	number of parts in J
j	single part of J
n_j	number of occurrences of w in j
$\sum_{j=1}^{P}$	sum runs over all parts j in J

Meanwhile, $\hat{H}(J|w)$ is the entropy of the word w in part J. For the calculation a random shuffling of all the words in the text is used and the entropy is built by averaging all possible random shufflings of the regular text.

However, Montemurro and Zanette [33] suggest another way of calculating the random entropy without having to use any kind of randomization. This is done by calculating

$$\hat{H}(J|w) = -P \sum_{m=1}^{min\{n,N/P\}} p(m) \frac{m}{n} \log_2 \frac{m}{n} \tag{4.5}$$

where

J, w, n	defined in Eq. 4.3
P	defined in Eq. 4.4
m	number of parts w occurs in

The marginal probability $p(m)$ is calculated as

$$p(m) = \frac{\binom{n}{m}\binom{N-n}{N/P-m}}{\binom{N}{N/P}}. \tag{4.6}$$

where

N, n	defined in Eq. 4.3
P	defined in Eq. 4.4
m	defined in Eq. 4.5

Both formulas are used to calculate the entropy by generating the entropy for a single part and multiplying it with the number of parts, which is fixed and known a priori. As can be seen, the information of words is additive over the whole corpus. Being additive helps making the computation of the metric concurrent and therefore making it significantly faster.

The metric returns a single value for each token type. Differently to MI, the Montemurro metric not only uses the frequency of a token type in the corpus but also the distribution. This results in words having a higher information value when they occur only in certain parts of the corpus. If a token type is in every part of the corpus, the metric will rate it as a lower quality feature, even if it occurs only once per part. An example for often occurring, lowly rated words are articles.

4.1.4.1 *Differences between Montemurro metric and Mutual Information*

The difference between Montemurro's metric and the normal MI method is the split into segments. The MI between one feature and a subclass or the entire corpus is purely calculated based

on frequencies. It calculates the amount of information a feature contributes to the subclass. Montemurro's metric has several independent segments. All the segment values of a feature are then summed up to get the overall importance of the feature in the corpus. This gives additional information about the distribution of the feature in the corpus, information MI is unable to pick up.

Because of the segmentation MI requires less resources to compute than the Montemurro metric.

4.2 IMPLEMENTATION

For the experiments, a Java[1] console-based application was written. Most of the application is custom made, except for the classifiers. The classifiers are publicly available and are written in C, C++ and Java.

All of the SVM and L2-regularized classifiers supported by my framework are shown in Table 15.

For performance reasons COLT[2] (a high performance scientific and technical computing library written in Java) was used. This has the distinct advantage of less memory usage due to the proper use of primitive types instead of wrapper classes (e.g. `int` $\rightarrow$ `Integer`, `double` $\rightarrow$ `Double`) and better performance, especially on retrieval and sorting tasks.

For parallelizing the application Java's built-in concurrency model was used. Threads spanned over the physical cores (one thread per core as suggested by Goetz et al. [18] in their book about Java's concurrency model) and provided an almost collision free parallelization process on certain tasks, for example training the classifiers. However, the emphasis does not lie on a perfect concurrency model but rather achieving a better performance with the least amount of coding effort.

Name	Version	Maintainer	Type
LIBLINEAR	1.6	MLGroup NTU	L2-regularized
LIBLINEAR Java	1.5	B. Waldvogel	L2-regularized
LIBSVM	2.91	MLGroup NTU	SVM
LaSVM	1.1	L. Bottou	SVM
SVM^{light}	6.02	T. Joachims	SVM
LIBSVM Java	2.91	MLGroup NTU	SVM

Table 15: List of all supported classifiers, their latest incorporated update and the current maintainer of the software.

[1] `http://java.com/en/` — visited on 22 Aug, 2011

[2] `http://acs.lbl.gov/software/colt/index.html` – visited on 22 Aug, 2011

4.3 USAGE OF MONTEMURRO METRIC

Montemurro and Zanette [33] ranked the features on regular English fiction books (e.g. Moby Dick). The actual metric was not further used to do any kind of feature selection and was not oriented to a specific task. Since it is now used for a multi-label classification task, two procedures have been developed on how to use the metric.

4.3.1 Subclass Level

The first is computed as follows:

1. one subclass is selected.
2. all of its documents are selected.
3. the first document is split up into parts, where s is the parameter defining the size of each part.
4. the entropy is calculated for every token in each part and summed up.
5. if there is overhead (remaining tokens in the document), it is temporarily cached. Example: If the length of a specific document is 4199 and s is 100, an overhead of 99 is found and added to the temporary cache.
6. the next document is analyzed in the exact same way as the first. At the end of the document, the overhead is added to the temporary cache. If the cache is now larger than s, the cache is processed and reduced.
7. this is done until every document has been processed.
8. repeat until every subclass has been processed.

The result of this procedure is a Montemurro Metric Value (MMV) for each token type for each subclass, therefore from now on it is also called subclass-level. Hence 631 is the maximum number of entropies a token type can have.

4.3.2 Corpus Level

The second procedure is similar to the first but instead of running it on one artificial corpus per subclass, it combines all the subclasses to a single large artificial corpus. From this follows all documents from one subclass are merged into a single document. This is done for every subclass. Once all subclasses consist of a single document, they are alphabetically merged into one

big artificial corpus. This means subclass A01P is merged before B60B. B60B in turn is merged before D06Q and so forth. All the subclass documents from one section are neighbors in the final document.

The situation gets worse considering patent categorization is a multi-label problem. Therefore a single patent document could theoretically be part of the new artificial document 631 times (the number of unique subclasses), since it could be labeled with each subclass ID. The overhead of each document is handled the same way it is handled in the first procedure. If a subclass has an overhead, it is prefixed to the first document of the next subclass. The result of the second procedure is a single MMV for each feature.

Calculating the entropy for the randomized text is done as explained in Subsection 4.1.4, again on subclass level. Since this procedure evolves around the idea of artificially creating documents that span over the whole subclass, the binomial of large numbers has to be calculated. Since a precise computation of the binomial requires extensive memory and CPU resources, an approximation of the binomial calculation is done. Stirling's approximation (presented in Feller [15] with a proof of its accuracy) is chosen and the $\log_{10}$ of said approximation is further used to calculate the entropy. Since it is proven that Stirling's approximation is very close to the actual optimal value, no real loss in accuracy is expected for the MI.

4.3.3 Results

Applying the metric on subclass level to the actual corpus yields mixed results. Unfortunately some negative features are not recognized as such. Table 16 shows the best and worst ten token types according to the metric for subclass G01N (which resides in section "Physics"). I ran it twice, once with $s = 1000$ shown on the left side and once with $s = 10000$ shown on the right side. The difference is significant. The union of the ten best token types for both runs is an empty set!

Column MMV shows the actual value of the metric. It can be seen as the confidence level. Especially around the neutral zero point, the results are mixed. There are false positives and negatives, the exact number is difficult to determine because of the large number of features that would have to be manually looked through. This makes it difficult to determine a threshold for removing the feature from the corpus. It also shows that negative features have a higher confidence level than their positive counterparts, especially the lower s gets.

Table 17 shows the best and worst ten token types according to the metric for subclass A22C. The difference between the two

	s = 1000		s = 10 000	
	Token Type	MMV	**Token Type**	MMV
Best	iats	4.878e-07	cys	1.81e-03
	lirion	4.876e-07	alteration	1.806e-03
	farmland	4.864e-07	round	1.80e-03
	prkc	4.864e-07	colored	1.799e-03
	prohibitin	4.863e-07	stimulate	1.797e-03
	megsin	4.860e-07	absorbing	1.794e-03
	cvhpp	4.859e-07	centrifugation	1.793e-03
	cert	4.857e-07	covalently	1.790e-03
	upps	4.851e-07	conservative	1.788e-03
	mgp	4.849e-07	minor	1.786e-03
Worst	cells	-2.411e-02	identity	-1.709e-02
	protein	-1.585e-02	polypeptide	-2.216e-02
	dna	-1.393e-02	encoding	-2.740e-02
	seq	-1.373e-02	probe	-3.018e-02
	amino	-1.360e-02	antibody	-4.313e-02
	binding	-1.34e-02	protein	-4.431e-02
	nucleic	-1.288e-02	sequences	-4.567e-02
	antibody	-1.132e-02	proteins	-5.006e-02
	polypeptide	-1.030e-02	nucleic	-5.504e-02
	proteins	-9.204e-03	dna	-7.595e-02

Table 16: MMV for subclass **G01N** (Physics).

subclasses is nicely shown by what the Montemurro metric finds to be important. Looking at the worst token types reveals the problem with the Montemurro metric for feature selection. It classifies certain tokens, e.g. meat, fish or sausage, with a low value (below zero). While such features might not be important within the subclass, they are great features for the classifier. The reason for this is the low probability of the word “meat" occurring in a subclass from section “Physics". Therefore it would be a great feature to distinguish a subclass from section “Physics" from a subclass related to food processing.

It is also interesting to see that there are fewer (if any) token types in subclass A22C that seem to be random character compositions as seen in G01N. Three of the top ten token types in G01N are “iats" (acronym), “cvhpp" (protein of genes) and

	s = 1000		s = 10 000	
	Token Type	MMV	**Token Type**	MMV
Best	loaf	1.904e-03	write	3.247e-05
	directing	1.895e-03	vibration	3.245e-05
	transportation	1.893e-03	transponder	3.233e-05
	brushes	1.890e-03	dosage	3.219e-05
	effecting	1.876e-03	evacuation	3.214e-05
	cabinet	1.875e-03	collars	3.203e-05
	beams	1.872e-03	cooler	3.197e-05
	shelf	1.870e-03	despooging	3.191e-05
	expose	1.869e-03	shim	3.187e-05
	antimicrobial	1.862e-03	breeder	3.187e-05
Worst	bracket	-1.197e-02	cellulose	-3.746e-02
	cooked	-1.328e-02	blade	-3.882e-02
	sausages	-2.215e-02	poultry	-4.472e-02
	fish	-3.130e-02	carcass	-5.427e-02
	blade	-4.161e-02	sausage	-6.785e-02
	conveyor	-4.342e-02	food	-8.080e-02
	poultry	-4.466e-02	conveyor	-9.818e-02
	casing	-6.806e-02	fish	-1.002e-01
	food	-8.018e-02	casing	-2.196e-01
	meat	-1.878e-01	meat	-2.553e-01

Table 17: MMV for subclass **A22C** (Human Necessities).

“cert" (protein production) which compared to “breeder", “write" and “dosage" seem to be nonsense-words.

4.3.4 Correlation to Zipf's Law

The correlation of the metric based on Montemurro and Zanette and CF is shown in Figure 40. The CF of a word is the number of occurrences in the collection. For both metrics their values were calculated on the entire corpus. Those values were used to sort the words and use the sorted list to retrieve the ranks for each word. For the Montemurro metric the higher the value for a word, the higher its rank. For CF Zipf's law [47] was used to rank the words. On the x-axis of the figure the ranks of the CF are

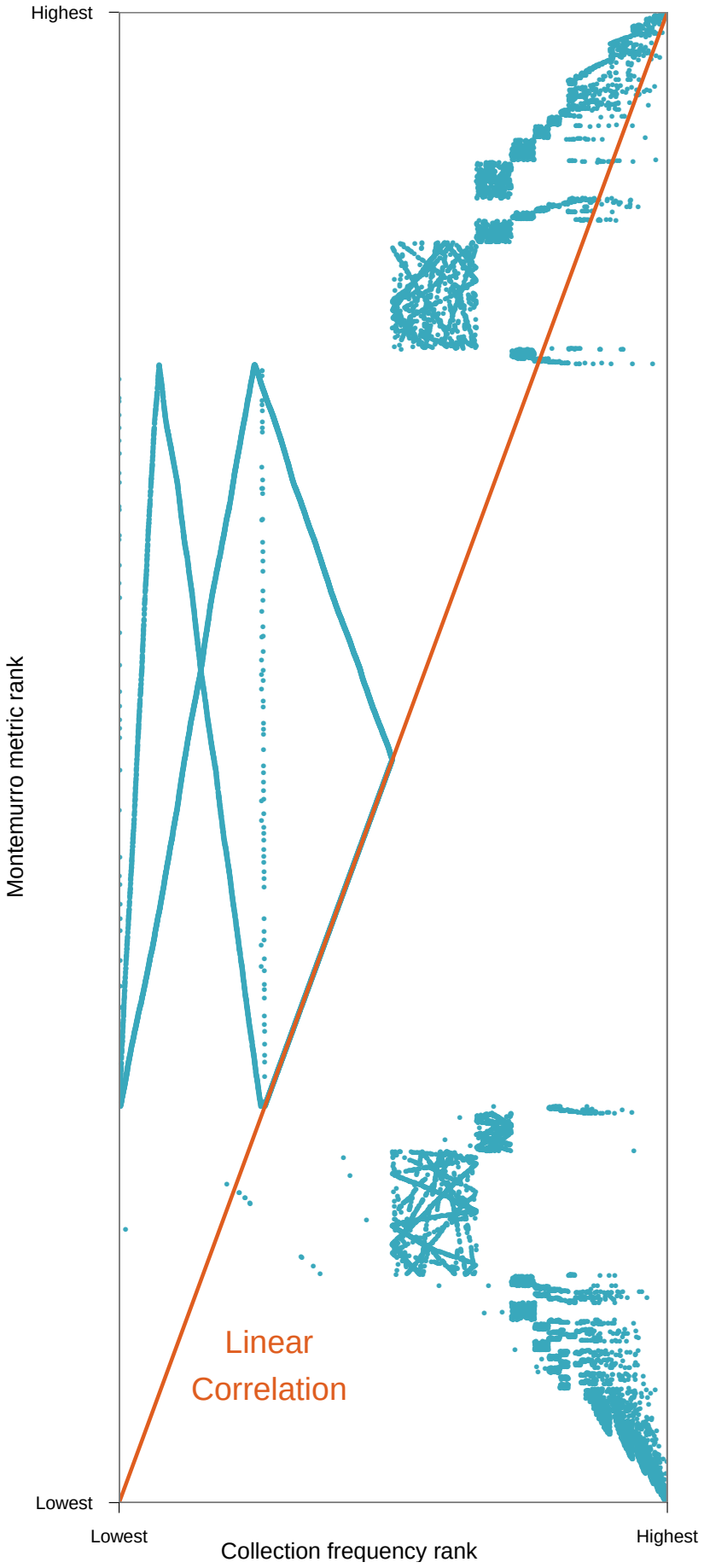


Figure 40: Rank of every word for both the Montemurro metric and the CF. Each dot represents a word.

drawn. The y-axis shows the Montemurro metric ranks. In the lower left corner are the lowest ranks for both metrics, the further up the dot is located, the higher the Montemurro metric rank. The further to the right the dot is located, the higher the CF rank for the word is. The solid line shows the linear correlation. If all the blue dots are on this line or at least very close to it, there will be a linear correlation between CF and the Montemurro metric algorithm.

The visualization shows the lack of a correlation between the two metrics. There seems to be no pattern. Inferring the value/rank of a word for the Montemurro metric by looking at the CF value/rank of a word is impossible.

4.3.5 Actual Metric Values

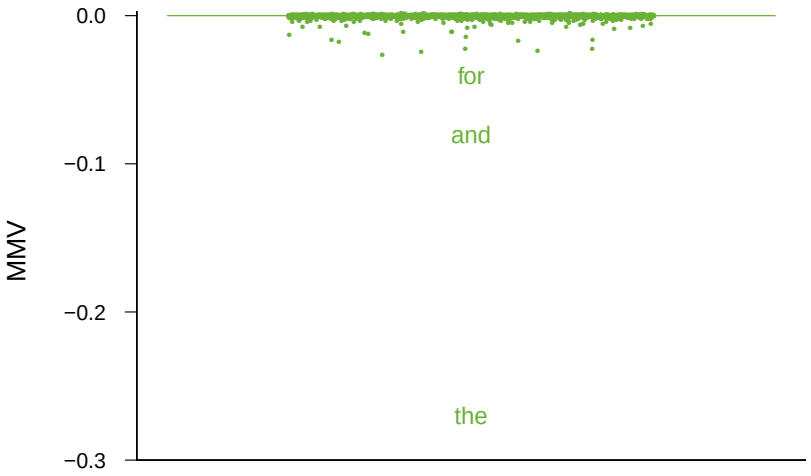
Figure 41 shows the distribution of the actual MMV of the entire USPTO corpus on corpus level. It zooms from a 100% coverage, including all the words present in the corpus, into an IQR representation. In between the inner 80% of the words are shown.

Figure 41a shows all the words in a box plot. The y-axis shows the actual MMV. The three most negative words are all function words. What is interesting is the rather large gap between the two words “the" and “and". The gap between them is larger than the entire range of MMV for all the other words. Despite the fact that a box plot is drawn, it basically is just a line because the IQR is so close to the median. Nonetheless there are quite a few outliers (tens of thousands actually).

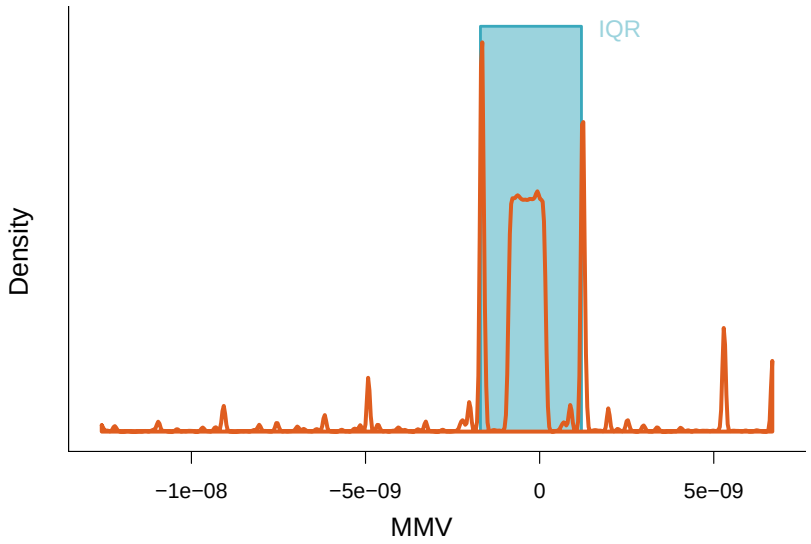
Since the IQR and both the 2^{nd} as well as the 98^{th} percentile are very close together we zoom into the data. Instead of a full 100% coverage of all the words Figure 41b presents a density function showing the inner 80%. The y-axis shows the density, the x-axis shows the specific MMV. By displaying only the inner 80% (10% from the top and bottom are removed), we cut off all the outliers on both sides of the spectrum. This leads to a simpler, more compact view of the actual MMV.

Not surprisingly the most used MMV are around zero. This goes hand in hand with Figure 41a. There are fewer words above than below zero. Most values have a low density. 30% of the words cover more unique MMV than the IQR, which is shown by the blue area.

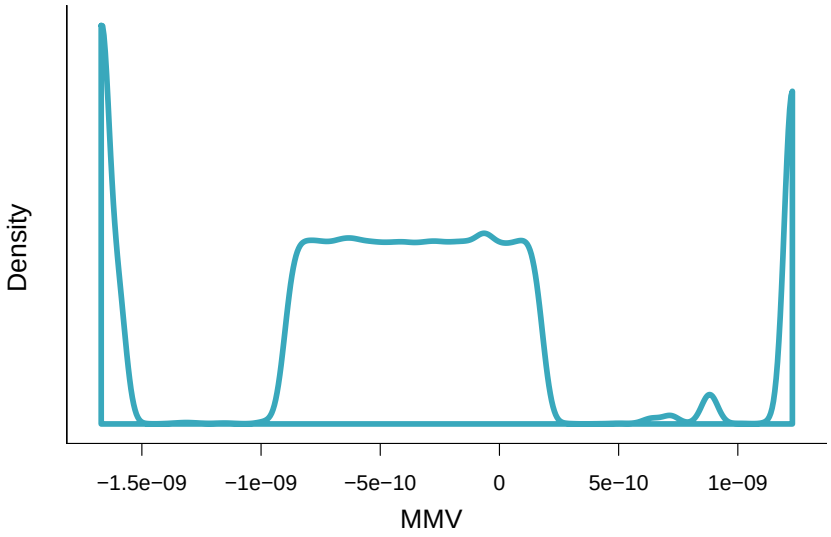
Figure 41c shows the IQR. There are not many values used between $-1x10^{-09}$ and $-1.5x10^{-09}$, the peak in the middle between $-1.0x10^{-09}$ and $1x10^{-11}$ contains the median and most of the values of the IQR. As can be seen the values are extremely small making it difficult to fine tune any filtering algorithm using this metric. There are tens of thousands of features having a value like $-5x10^{-10}$ while the word “the" has a value of -0.25. Such a



(a) Box plot with 100% coverage.



(b) Density function with inner 80% of the words.



(c) Density function of IQR

Figure 41: MMV for all the features and how the features are distributed amongst the small range of possible values.

discrepancy would either favor the word “the" or would take tens of thousands of features in because it can’t distinguish between them. Such a behavior would very quickly put a limit on the number of words that can be removed with a feature selection algorithm based on the Montemurro metric.

4.4 MONTEMURRO METRIC SURVEY

To test the hypothesis of how well Montemurro’s metric is able to identify the most and least important features in a patent corpus (from a linguistic standpoint), an online survey was conducted. There was no prior requirement for any person to participate (all non-anonymous participants are listed in Section A.3). A total of 25 people voted on at least one subclass, 80 subclasses have been voted on. These 25 people were either students, professors or professionals in the field they voted on. Unfortunately this means over 550 subclasses received no vote at all. Even worse the entire section B received no vote. For this reason the survey should not be seen as a complete, empirical study but rather give a first glance at how well the Montemurro metric performs. Due to the highly specialized fields involved in patent documents it is difficult to get competent people to participate in such a survey. A future master thesis could solely concentrate on finishing such a survey.

During the sign up, a user selected one section. Randomly selected subclasses from this one section were presented to him. This was done because showing a subclass related to for example physics is useless if the user has no knowledge about physics. Because of this the option to skip a subclass altogether was also available. This option was used 147 times from 19 users.

For each subclass 30 words from the subclass were selected and shown to the user one by one. 15 out of those 30 were rated as the most negative by the Montemurro metric (when applied on subclass level — the entire USPTO corpus was used), the remaining 15 were rated as the most positive, a stop word list was applied to remove the worst features like “the" or “each". The participant had to either agree or disagree with the word being related to the subclass.

Figure 42 shows a screenshot of the online survey, specifically the voting page for a user. The blue area shows the currently to-judge word, the orange category area shows the name of the currently judged subclass.

I present the results of the survey in form of a confusion matrix in Tables 18, 19 and 20. Said matrix was introduced in Section 2.6, explaining what true positives, false positives, false negatives and true negatives are. This applies to all three tables but with

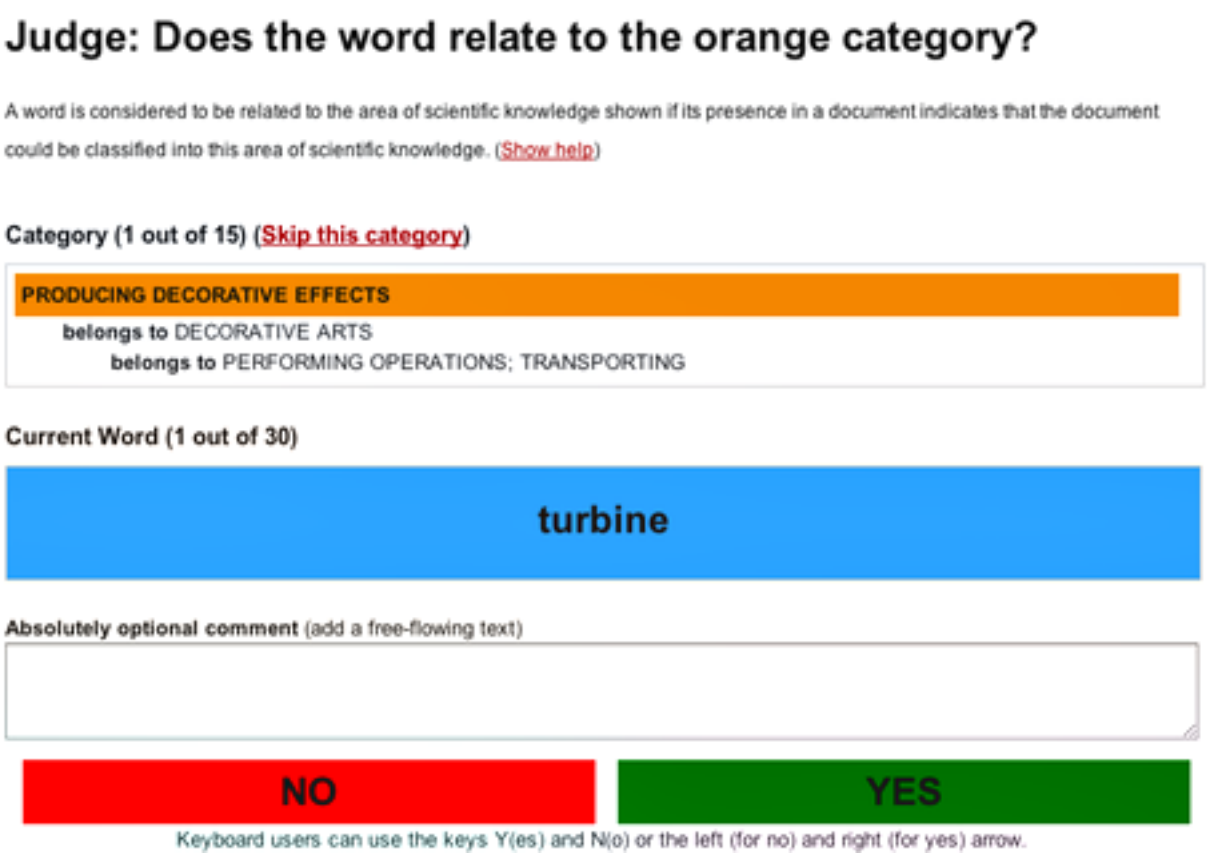


Figure 42: Screenshot of online survey.

columns and rows switched. The columns of the tables show the prediction, which results from my algorithm. The rows named "Vote" reference the actual data, which results from the votes of the users. While a basic confusion matrix shows the four metrics in absolute numbers, I present percentages. And due to the 50/50 split of positive and negative words, each column sums up to exactly 50%. Furthermore for these specific tables the sum of both true positive and true negative shows the total percentage of agreement between my algorithm and the users.

Table 18 shows the agreement of the participants with the Montemurro metric on subclass level. This means the Montemurro metric was applied to a single subclass instead of the entire corpus. On average 64% of the time the participants agreed with the Montemurro metric when both the user and the metric judged it as a positive feature, 72% when it was a negative one. A disagreement on positive features was happening 28% of the time and 36% of the time on negative features. This translates into an agreement 68% of the time, 32% of the time there was a disagreement. It does not shift enough to either false negatives or false positives to really determine a trend.

As mentioned in the previous section when Montemurro is applied on subclass level it identifies words as negative features be-

		Word	
		Positive	Negative
		%	%
Vote	Positive	32	14
	Negative	18	36

Table 18: Agreement and disagreement of the survey amongst all subclasses on subclass level.

		Word	
		Positive	Negative
		%	%
Vote	Positive	46	12
	Negative	4	38

Table 19: Agreement and disagreement of the survey amongst all subclasses on class level influenced by corpus level information.

cause they occur often and evenly distributed within the subclass. Therefore we will compare the votes against the Montemurro metric on both corpus and class level. Table 19 shows the average agreements between the participants and the Montemurro metric on subclass level with corpus level information. This means a negative feature in a subclass is transformed into a positive if the corpus level information indicates that it is indeed a positive one. Transforming the other way around is unfortunately impossible because of the lack of data. One such feature is “Meat". It is rated as negative on subclass level for specific subclasses (e.g. subclass “Butchering") but participants voted it as positive. On corpus level it is a positive feature as well since it does not occur very often. When it occurs, it is highly centralized. This would turn the feature into a true positive. The table immediately shows the difference for true positives. This is because of the corpus level information, which does not have such a strong impact on true negatives and false positives. The overall agreement is now 84%, an increase of 16 percentage points.

As mentioned before there were many subclasses with zero or a single vote. When looking at the three subclasses with three votes each, the agreement rate increases, even when compared solely to subclass level information. This can be observed in Table 20, which does not use corpus level information. It is almost as good as the Montemurro class level classification influenced by Montemurro corpus information. Whether this is because a single vote is not strong enough and might have a few misclassifications by the participant himself or because the three subclasses with the

		Word	
		Positive	Negative
		%	%
Vote	Positive	45	8
	Negative	5	42

Table 20: Agreement and disagreement of the survey amongst the three most voted on subclasses without corpus level information.

highest number of votes are easier is unclear. This would have to be investigated in future survey works as well, both by increasing the quality of the examiners and by increasing the number of words tested per subclass. Testing each subclass would clearly be beneficial as well.

4.5 FILTERING WITH MONTEMURRO–ENTROPY

This section presents my three feature selection algorithms. All three of them are based on Montemurro and Zanette's metric and are two step algorithms. First, the metric evaluates all the features in an adapted corpus. Second, the evaluation is used to determine which feature to keep.

This section does not present experimental results. Those can be found in Chapter 5.

4.5.1 Word-Entropy Based Filter 1

Algorithm Word-Entropy Based Filter (WEBF) is proposed. Currently there are two versions, this subsection presents Word-Entropy Based Filter 1 (WEBF1). The algorithm uses the metric based on subclass level, i.e. creating one artificial corpus per subclass by combining all the documents belonging to that subclass and applying the Montemurro metric onto this newly created corpus.

WEBF1 creates a single index for all of the binary classification tasks. Algorithm 1 shows how the algorithm works. It compares each feature with all the other features and if the difference is too small (i.e. falls below parameter threshold α), it removes one of the two features. This should remove all redundant features because they should be very similar and hence their sum should fall below threshold α. Similarity stems from two features being used in the same subclasses. If this is the case, the difference is small. If they also have almost the same Montemurro values, their difference decreases to zero, making them almost identical. If they are this similar, their sum of subclass differences might

fall below threshold α. The higher the threshold α, the bigger the difference has to be for both to survive the comparison.

The algorithm, including the computation of the metric, has a total of two parameters, the threshold α used in the algorithm and the size of each part s used during the computation of the metric.

4.5.1.1 *Algorithm*

Algorithm 1 WEBF1 (Featureset F, Threshold α, MMV on subclasses $M(C_1, \ldots, C_N)$)

```
1:  insert all features from F into S
2:  F_i ← FirstFeature(F)
3:  repeat
4:    F_j ← NextFeature(F,F_i)
5:    repeat
6:      diff ← Σ_{k∈C}^{|C|} abs(M_{k_{F_i}} − M_{k_{F_j}})
7:      if diff < α then
8:        remove F_j from S
9:      end if
10:     F_j ← NextFeature(F,F_j)
11:   until F_i == NULL
12:   F_i ← NextFeature(F,F_i)
13: until F_i == NULL
14: return S
```

Line

4 uses NextFeature to receive the next feature after F_i in S, which in the very first loop would be the second feature of F. The method NextFeature will not return a feature which is no longer in S.

5-11 inner loop comparing every feature to F_i.

6 computes the sum of the absolute difference between each MMV for F_i and F_j over every subclass k in C, expressed as $M_{k_{F_i}}$. If a feature is not in a subclass, the value 0 is used as a substitution. Hence if both features are not present in the same subclass, the subclass has no meaning to the difference.

7/8 determines if the difference is below α and if so, line 8 removes the current feature of F_j from S. There is no chance of F_j ever being inserted back into S once it has been removed.

10 stores the next feature after F_j in F_j.

12 stores the next feature after F_i in F_i.

4.5.1.2 *Demonstration*

The comparison of two features can be visualized by using a histogram. Figure 43 shows a comparison between two features that are highly different. First, they occur in many subclasses. Second there are almost 300 subclasses where they don't occur at the same time. In such a case the difference is high and it is almost always a feature pair that will be kept.

The second case arises when two features are similar. This means their histograms look almost the same. The underlying theory here is that feature A is redundant when feature B is present. Since this does not add any information to the classifier, it is unnecessary to keep both since it would just increase the resources required later (e.g. during classification, filtering keywords, POS-taggers, ...). Figure 44 shows a histogram of two similar features. The difference between these two features will certainly fall below threshold α and therefore be removed. Once it is removed from subset S (as seen in Algorithm 1), it will never be added to it again. Therefore this feature can be skipped for every inspected feature coming after it.

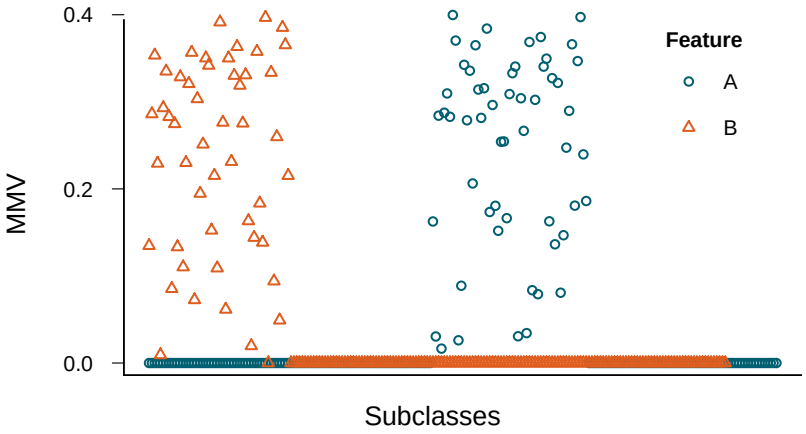


Figure 43: Histogram of two different features.

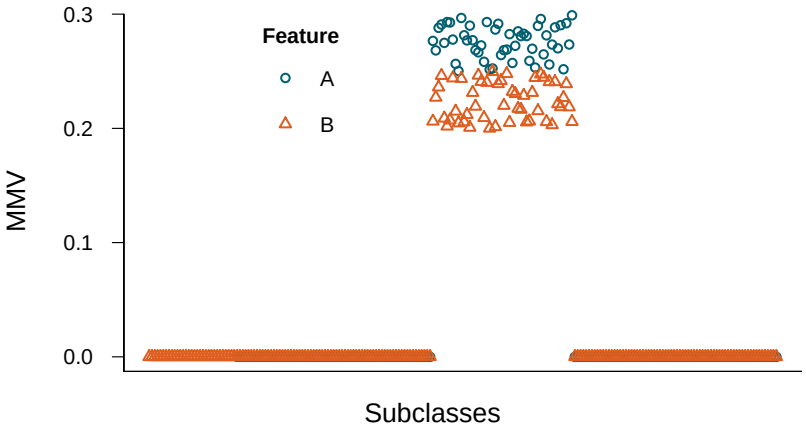


Figure 44: Histogram of two similar features.

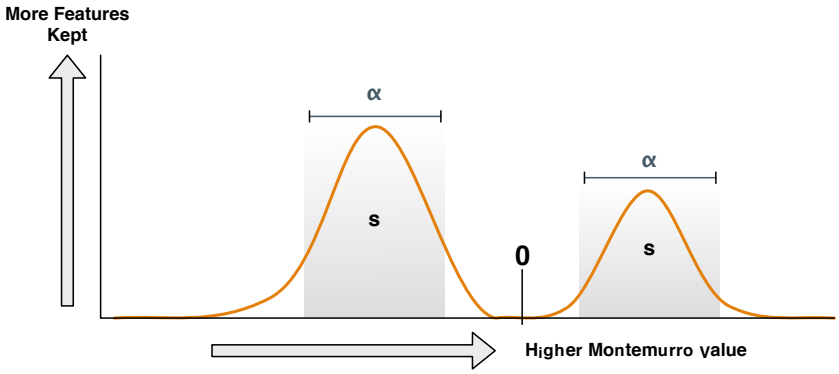


Figure 45: Workings of WEBF1.

This method does not emphasize the removal of negative features. It aims to find a balance between removing all negative features and just enough so each document has enough features during the training of the classifier. Since negative features have a higher probability of occurring, WEBF1 keeps some of them to ensure the removal of more features around or above zero.

Since some negative features are left in, one cannot use a stop word list since this would most likely remove words that are considered negative features. A stop word list based on the Montemurro metric will therefore be presented independently in Subsection 4.5.3.

How the two parameters, α and s, interact can be seen in Figure 45. The y-axis shows how many features are kept, the x-axis shows MMV for a feature. MMV gets higher to the right. The right top corner would mean the highest rated features according to Montemurro would be kept the most. As can be seen, features are not selected equally amongst the different Montemurro values. There are two significant curves in the function line, one is between the most negative and the most neutral features (according to the Montemurro value). The second is between the most neutral and the most positive features. The actual width of the two curves is defined by α. The lower the value of α, the wider, higher and smoother the curve. In general (i.e. if the α setting selects less than 10% of the features) the first curve contains more features than the second, as shown in the figure. The selected features by this algorithm depend on parameter s. This parameter fine tunes the actual Montemurro values, putting one feature above or below another one. It therefore can shift a feature outside of the curve.

The reason why the left curve takes more features is because these features occur in more subclasses than the positive features and their sum is therefore larger. This means features have a higher probability of being kept the more subclasses they are present in. This is done on purpose to keep the negative features

and to see if the negative features selected by Montemurro have enough classification potential to distinguish subclasses from each other.

Since it is unknown if the balance between positive and negative features is really needed, a second version named Word-Entropy Based Filter 2 (WEBF2) is introduced in the next subsection. It will tackle this problem through normalization. Normalization should work especially well when two features occur in different subclasses or their number of subclasses differs.

4.5.2 Word-Entropy Based Filter 2

WEBF2 aims at balancing the injustice, which occurs when two features are compared and they occur in a different number of subclasses, by normalizing the sum.

Algorithm 2 shows this approach. The algorithm is the same as in Algorithm 1 with the only differences in line 7 and 8. There the variable diff is being normalized by the number of unique subclasses the two features are in. For example, if feature A is in the subclasses $\{0,1,2,3\}$ and feature B is present in the subclasses $\{1,3,4,5\}$, the number of unique subclasses would be 6 since the UNION of those two sets is $\{0,1,2,3,4,5\}$. When using this new approach, it is necessary to change the threshold α. Being normalized, the values of diff are quantized and are within a smaller range compared to WEBF1.

Algorithm 2 WEBF2 (Featureset F, Threshold α, MMV on subclasses $M(C_1,\ldots,C_N)$)

1: insert all features from F into S
2: $F_i \leftarrow$ FirstFeature(F)
3: **repeat**
4: $F_j \leftarrow$ NextFeature(F,F_i)
5: **repeat**
6: diff $\leftarrow \sum_{k\in C}^{|C|} abs(M_{k_{F_i}} - M_{k_{F_j}})$
7: count $\leftarrow |UNION(subclasses(F_i), subclasses(F_j))|$
8: diff $\leftarrow$ diff / count
9: **if** diff $< \alpha$ **then**
10: remove F_j from S
11: **end if**
12: $F_j \leftarrow$ NextFeature(F,F_j)
13: **until** F_i == NULL
14: $F_i \leftarrow$ NextFeature(F,F_i)
15: **until** F_i == NULL
16: **return** S

Chapter 5 shows the differences in the number of selected features. The difference is not as much as one might think, still

the best performing experiments for WEBF1 and WEBF2 share approximately 80% of the features. Therefore the Figure 45 also applies for WEBF2, although more features are kept with WEBF2, these features stem from around the zero point, which would create a third bump in Figure 45. This third bump will get larger and, if enough features are selected, will get higher than the other two.

4.5.3 Word-Entropy Based Stop Word List

In addition to WEBF1 and WEBF2, a simple ranking based feature selection algorithm named Word-Entropy Based Stop wOrd List (WEBSOL) was developed. WEBSOL is based on the whole corpus. The way to compute the metric, which was used to rank the features, is described in Section 4.3. Since each feature now has only one value, the metric can be used the same way other metrics (TF-IDF, MI, IG, ...) are used. Therefore we sort all the features by their MMV. Based on the sorted list we can select an exact number of features, which is impossible with WEBF1. The lowest β features are removed, hence parameter β sets how many features are removed. Since the list is already sorted, the feature selection algorithm WEBSOL is a simple ranker.

Because the only filtering is done by ranking, it could be used as a stop word list instead of a frequency based stop word list (e.g. TF).

4.5.3.1 *Demonstration*

This behavior and the meaning of the parameters is shown in Figure 46. β signals the number of features kept by computing $N - \beta$, where N is the total number of features. Since the features are ranked by their Montemurro value, features with a lower value are removed first. This gives features with a higher value a better chance of survival. Parameter s determines what features are kept since s changes the actual Montemurro value. Because keeping a feature is not dependent on the actual Montemurro value (i.e. a feature is kept based on its rank and not because it is below a certain threshold), there is a steep increase in the number of features kept, indicated by the 90° angle of the function line.

The biggest difference to both WEBF1 and WEBF2 is the removal of only negative features. Positive features occur less often than negative features. This automatically means WEBSOL cannot remove as many features as WEBF1 because the more features WEBSOL removes, the less often the remaining words occur. This could result in documents with no words left making them impossible to classify.

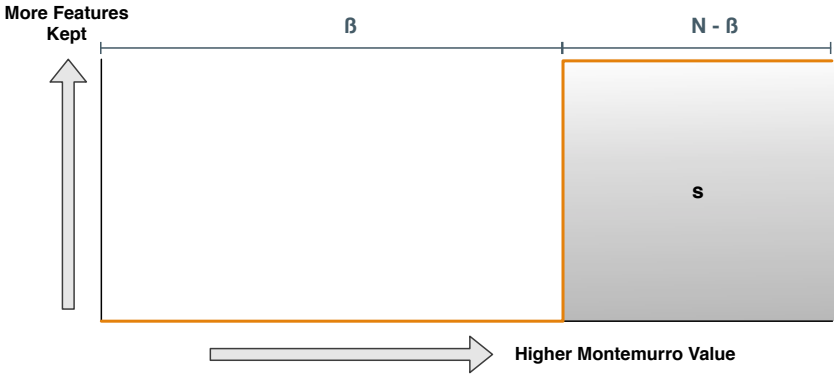


Figure 46: Workings of WEBSOL. Parameter β determines how many features are removed, s which features are kept. N is the total number of features in the corpus.

4.5.3.2 *Algorithm*

Algorithm 3 shows the implemented procedure. No line-by-line analysis is given because the algorithm is self-explanatory.

Algorithm 3 WEBSOL (Featureset F, Threshold β, MMV on corpus M)

1: Sort F by values from M in ascending order
2: Remove the first β features
3: Store remaining features in S
4: **return** S

4.5.4 Differences between WEBF and Montemurro Metric

While all three proposed algorithms use the Montemurro approach of evaluating features, they provide different methods of applying it to a collection of documents. In their article Montemurro and Zanette apply their metric only to a single book, I apply it to millions of single documents. To the best of my knowledge Montemurro and Zanette have not tried this yet.

Table 22 illustrates further differences between my proposed algorithms and the metric by Montemurro and Zanette.

	Montemurro metric	Proposed algorithms
Purpose	Identifying important words in a book	Using the metric to reduce the number of features
Corpus	Single book	Introduces technique for applying the metric and WEBF to millions of patent documents
Document Length	Avg. 120 000 words per book	Avg. 6 000 words per document. Concatenated in the billions
Document Type	Single topic per book	Multiple, unrelated topics
Feature Selection	Never been used for feature selection	New algorithm using the Montemurro metric (WEBF1, WEBF2, WEBSOL)
Categories	Different categories are not taken into account	Takes subclasses and their sections into account

Table 22: Differences between Montemurro metric and proposed algorithms.

5 EXPERIMENTS

CONTENTS

This chapter presents the various experiments surveying the effect of WEBF1, WEBF2 and WEBSOL on the classification performance and the number of features. Furthermore the impact of the parameters on the actual performance of the feature selection algorithms is explored.

5.1 CORPUS

The corpus for the experiments is based on the USPTO corpus. Due to performance issues the experiments were run on smaller subsets of the corpus. Instead of using all of the roughly 3.5

million patent documents, subsets with approximately 100 000 and 200 000 documents were built.

A third subset with 100 000 was built. This dataset is balanced, containing exactly 50 subclasses. Each subclass has 2000 documents assigned to it.

The connectivity, which was explained in Section 3.5, is not kept. Finding a subset of a corpus with both the same ratio and the same connectivity of the subclasses turns out to be a hard problem. Therefore the constraint of the connectivity between subclasses was removed.

The algorithm for finding the subset with the same ratio of samples per subclass:

1. The user sets the desired number of documents.
2. The user sets the percentage of documents a subclass should take from the original corpus.
3. Sort the subclasses by their number of documents in ascending order.
4. Split the subclasses into bins, where each bin has the same number of subclasses.
5. Set the first bin, containing the subclasses with the most documents, as the current bin.
6. Randomly select a subclass from the current bin.
7. Select the documents based on the percentage from step 2.
8. Set the next bin as the current bin.
9. Go to step 4 until enough documents are chosen.
10. If the number of selected documents is more than 20% bigger than the number set in step 1, the ratio from step 2 is reduced by 10% and the algorithm starts at step 4 again. Otherwise the algorithm is finished.

The first two steps are performed by the user and can be regarded as the initial input, the remaining eight steps are performed by the algorithm.

This does not select the exact number of documents because the ratio of documents per subclass is kept in the sub-sampled corpus. If subclass A has three times more documents than subclass B in the original corpus, it will have three times more documents in the new sub sampled corpus if both are selected. Applying this algorithm to the corpus yields the results from Table 23.

As mentioned earlier, the third corpus, b100k, is balanced. Although patent corpora are imbalanced in real life, this corpus

	#			Viz.
	Docs	**Features**	**Subclasses**	**Figure**
	k	k		
100k	135	1037	49	47
200k	225	973	75	48
b100k	100	897	50	49

Table 23: All subsets with references to their distribution visualization.

Section	**Corpus**		
	100k	**200k**	**b100k**
A	6	9	7
B	10	17	9
C	8	11	8
D	2	3	1
E	2	6	1
F	8	15	4
G	11	8	11
H	2	6	9
Total	49	75	50

Table 24: Number of subclasses in sections for each sub sampled corpus.

would ensure that the algorithm runs on balanced corpora and that the algorithm does not prefer one corpus characteristic over the other. This happens when the algorithm would tailor the subset of features to prefer larger subclasses since their size makes them more important for the macro-averaged F1. This can obviously not happen in a balanced corpus hence the inclusion of this corpus.

Table 24 shows the number of subclasses for each section in each of the three sub sampled corpora.

Figure 49 shows the number of subclasses per section for the b100k corpus. Sections D and E have fewer subclasses than they should have because both sections contain many subclasses with fewer than 2000 documents. A new ratio between the sections was computed based on only the valid subclasses (those with more than 2000 documents assigned to them). For each section subclasses were randomly selected from the valid subclasses.

Figures 47 and 48 show the number of documents in the 100k and 200k corpus respectively.

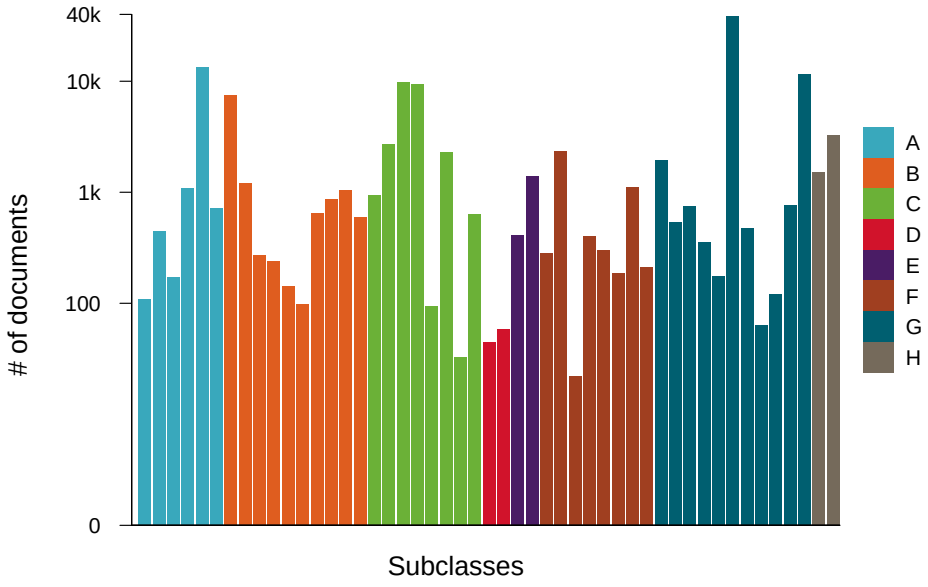


Figure 47: Documents per subclass - 100k.

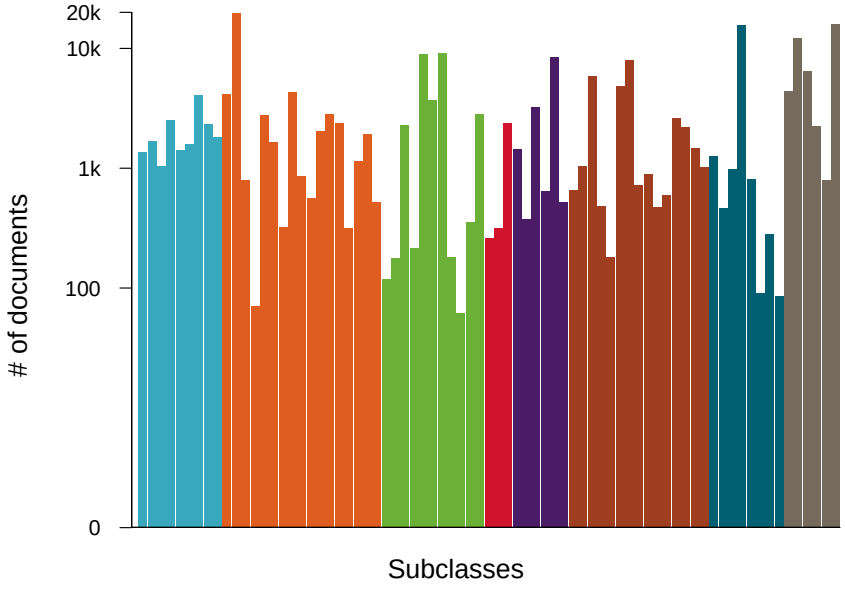


Figure 48: Documents per subclass - 200k.

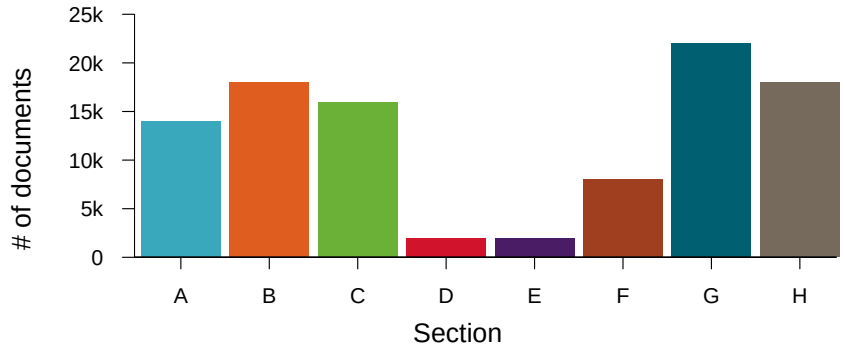


Figure 49: Documents per section - b100k.

5.2 EXPERIMENTAL ENVIRONMENT

	Type	Description
Hardware	Machines	2 IBM x3950
	CPU	32 Cores (4 Quad core Intel XEON@2.93 per node)
	RAM	256GB
	Disk	IBM DS4800, SATA RAID-5
Software	OS	Fedora Core 11
	Programming	Java 1.6, Ruby 1.9.1
	Libraries	Java Colt, Java Google Collection, Weka
	Classifiers	SVM^{light}, LIBLINEAR

Table 25: Details of the MDC, provided by the IRF.

All the experiments were run on the MDC, a server provided by the IRF. Table 25 shows the details of the MDC and the used software.

All experiments have been parallelized and used the memory extensively. Most of the experiments utilize ten cores.

Due to the many documents in the corpus, one terabyte of hard disk space has been used. This was mostly used to serialize the several sub corpora (deserialization being faster and easier to parallelize than parsing the raw data) and to create training and test files for the classifiers. Instead of reading everything from the original MAREC each time an experiment was started, the three corpora were parsed once and the data in the form of the programming language's data structure was written in plain text files onto the disk (also known as serialization). Now every time an experiment was started, the text file was read and the MAREC data in form of the programming language's data structure was extracted (also known as deserialization).

The following setup was used for the experiments:

preprocessing De-capitalization, removal of non-alphabetic characters such as numbers or braces. XML-tags 'Figure' and 'Table' were removed. For more details, please have a look at Subsection 2.5.1.

For the baseline, the 2000 most frequently used words are removed.

evaluation	Split ratio for training/test: 80/20. For validation repeated random sub-sampling is chosen. Each experiment is repeated ten times. The quality of the categorization was evaluated by using recall, precision and F1. Due to the lack of balance in the corpus the focus lies on macro-averages calculated at subclass-level.
hardware	10 cores and 75 GB memory were utilized.
classification algorithm	SVM^{light} was used for the SVM implementation. A proper introduction to SVM and SVM^{light} can be found in Subsection 2.4.1. SVM^{light} determines the cost parameter C on its own by calculating $[avg.\ x * x]^{-1}$. This default behavior was used for both the baseline and the feature selection algorithm. LIBLINEAR was the L2-regularized linear classifier used, its introduction can be found in Subsection 2.4.2. The cost parameter C for LIBLINEAR was the only overridden parameter and was evaluated with a grid search. One grid search was run for the baseline and one for the feature selection algorithm.
c settings for liblinear	baseline: $C = 60$. fewer features: $C = 120$.
training method	One-vs.-rest binary classifiers with TF*IDF as term weighting. Trained on the subclass level of the IPC.

Stemming was tried on the baseline, specifically the Snowball stemmer. Not only was it absolutely unsuccessful at improving the baseline, it even made it worse.

5.3 EXPERIMENTAL RESULTS AND BASELINE

The following experiments explore the effects of the Montemurro metric described in Subsection 4.1.2 and the filter algorithms WEBF1, WEBF2 and WEBSOL. Unless stated otherwise results were achieved by using LIBLINEAR as classifier.

Tables 26 and 27 show the baselines for all corpora for LIBLINEAR and SVM^{light} respectively, which is the base for all the

	F1	Precision	Recall
	%	%	%
100k	82.5	94.2	73.3
200k	76.2	92.7	76.3
b100k	84.0	84.8	69.1

Table 26: Macro-averaged metrics for the baseline with LIBLINEAR.

	F1	Precision	Recall
	%	%	%
100k	66.8	75.9	59.7
200k	66.2	78.3	57.3
b100k	79.1	89.8	70.7

Table 27: Macro-averaged metrics for the baseline with SVM^{light}.

classification performances of WEBF1, WEBF2 and WEBSOL. It uses all the available features in the respective corpora. This shows how well the classifiers perform on the raw data, when no interference takes place. For our experiments, we use the same settings (the corpus, the features, the classifiers, the classifier settings). Therefore any change in classification performance that occurs can be attributed to the feature selection algorithms.

Every analysis is done using macro averages for recall, precision and F1. All baselines have the same characteristics concerning higher precision than recall and the ratio between precision and recall is approximately the same as well.

The feature selection algorithms we compare the three algorithms against are TF-IDF, MI and IG. All the features in the corpus are ranked by the TF-IDF/MI/IG measure and the top N features are kept. This has the additional advantage of being able to compare the exact same number of features, otherwise impossible because one cannot determine the exact number of features for WEBF1 and WEBF2 a priori.

As mentioned in Subsection 2.8.1, many visualizations of the experiments use a so-called box plot.

The only difference to box plots from prior sections is that the individual box plots of the sections have the same width. Since some sections have fewer subclasses in the sub sampled corpora, their widths would be too small and they would lack outliers, whiskers and a large IQR. This would make it easy to overlook an entire section.

The rules for the table layouts in this chapter are the same as explained in Subsection 2.8.2.

In this chapter each subsection consists of four different areas explaining four different points about the subsection:

introduces gives references to the tables or figures introduced in the subsection and, if necessary, gives references to cross-references chapters and sections.

motivation gives the reason behind the decision to include this subsection at all and what will be presented in it.

legend explains the introduced chart and how to interpret it. It also lists the classifier and corpora used in the chart.

technical analysis offers an interpretation of the newly introduced tables and figures. If possible, a reason why this happens and why my algorithm behaves the way it does is given.

In some subsections a fifth area is shown:

disclaimer some subsections present data that, while they are truthful, might not represent reality accurately. This area explains explicitly what might be misunderstood about the data and what you have to be cautious about.

The areas are marked by margin notes. This is valid for all three Sections (5.4, 5.5, 5.6).

5.4 WORD-ENTROPY BASED FILTER 1

This section evaluate WEBF1 by investigating the impact of the features on the classification performance, the two parameters α and s and the two different classifiers. Furthermore it analyzes the impact of the fewer features from WEBF1 on the CPU runtime and the impact of the feature skipping mechanism of WEBF1 on the number of feature comparisons.

5.4.1 Classification Analysis

This subsection shows the classification performance. The baseline is compared to WEBF1 on basically three different measures: F1, precision and recall. All three are macro- and not micro-averaged.

Since both SVM^{light} and LIBLINEAR perform similarly, most of the analysis is done with LIBLINEAR. As will be shown later in Subsection 5.4.4, LIBLINEAR performs significantly faster than SVM^{light} and training the three other feature selection algorithms (MI, IG, TF) with SVM^{light} was not feasible due to time constraints. Since this particular implementation of an SVM has more parameters than LIBLINEAR, a grid search for the classifier settings for each of the feature selection algorithms would require too many resources.

5.4.1.1 *Best Experiment Selection*

		Features		F1		
		#	%	Base	WEBF1	Diff
		k	%	%	%	PP
LIB	**100k**	11.8	1.1	82.5	84.3	+1.8
	200k	33.6	3.4	84.0	85.7	+1.6
	b100k	57.3	6.3	76.2	76.3	+0.1
SVM	**100k**	12.7	1.2	66.8	67.8	+1.0
	200k	33.8	3.5	79.1	79.9	+0.8
	b100k	17.5	2.0	66.2	66.7	+0.5

Table 28: F1 comparison between WEBF1 and baseline.

		Precision			Recall		
		B	W	Diff	B	W	Diff
		%	%	PP	%	%	PP
LIB	**100k**	94.2	92.9	−1.3	73.3	77.2	+3.9
	200k	92.7	90.9	−1.3	76.3	81	+4.7
	b100k	84.8	81.4	−3.4	69.1	71.8	+2.7
SVM	**100k**	75.9	76.1	+0.2	59.7	61.2	+1.5
	200k	89.8	89.9	+0.1	70.7	71.9	+1.2
	b100k	78.3	77.8	−0.5	57.3	58.4	+1.1

Table 29: Recall and precision comparison between WEBF1 (W) and baseline (B).

	LIB		SVM	
	s	**α**	**s**	**α**
100k	1000	0.0001	1000	0.000 01
b100k	15 000	0.000 001	1000	0.000 01
200k	5000	0.000 01	1000	0.000 001

Table 30: Best combination for the two parameters of WEBF1.

Introduces Tables 28, 29 and 30.

Motivation The best classification performances and their parameter settings for WEBF1 for all three corpora are shown.

Legend **Table 28**: column “Features" shows the relative size of the features from the first column. Macro-averaged F1 in percent is shown for LIBLINEAR and SVM^{light} for all three corpora. Column “Diff" shows the difference between WEBF1 and the baseline. A plus sign denotes an improvement by WEBF1 over the baseline.

Table 29: shows macro-averaged recall and precision for all three corpora.

Table 30: shows the two parameters α and s of WEBF1.

Technical Analysis The overall improvement in macro-averaged F1 with WEBF1 to the baseline is relatively small. Depending on the corpus 94% to 99% of the features can be removed. SVM^{light} removes more than LIBLINEAR without sacrificing classification performance on the b100k corpus. The difference between LIBLINEAR and SVM^{light} is small and could be attributed to parameter settings of the classifiers. The only non-marginal difference can be seen in terms of the number of selected features in the b100k corpus.

WEBF1 increases recall, precision drops below or close to the baseline.

5.4.1.2 *Method Comparison*

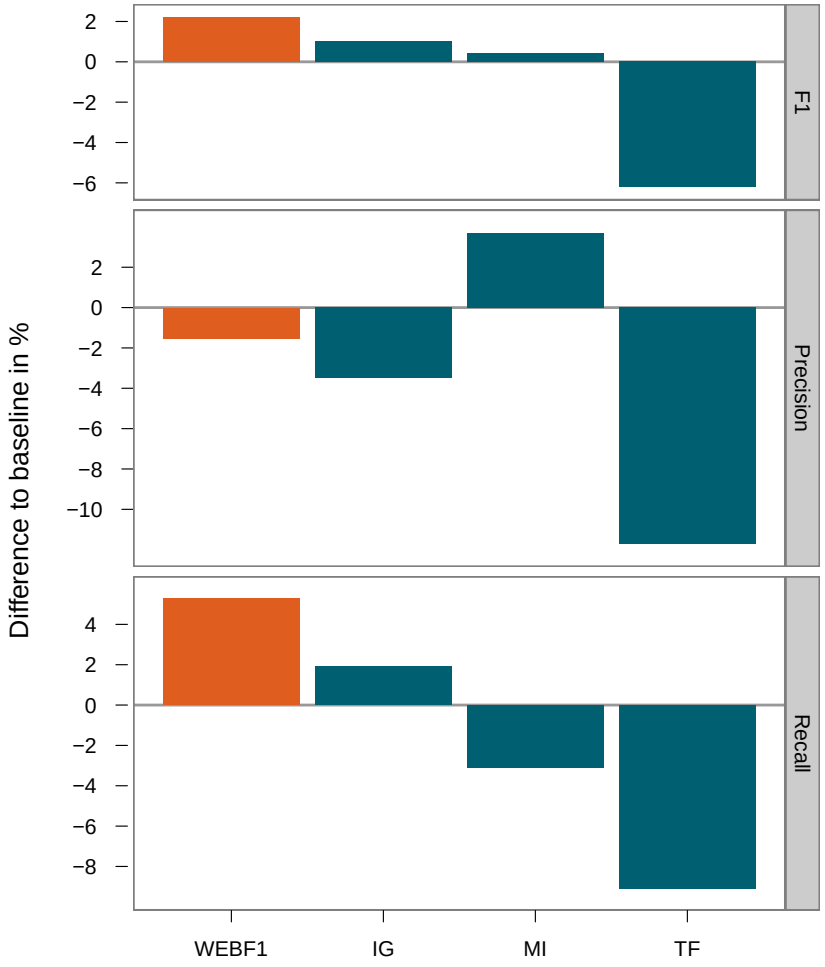


Figure 50: Comparison of WEBF1 to other methods.

Figure 50. *Introduces*

To see how well WEBF1 performs it is compared to three other feature selection algorithms and the baseline. *Motivation*

The y-axis shows the difference between a method and the baseline in percent grouped by the three measures. The x-axis shows the four methods, WEBF1, MI (MI), IG (IG), TF-IDF (TF). The data comes from the 100k corpus with LIBLINEAR . For each of the methods F1, precision and recall comes from the best performing experiment. *Legend*

While MI is the only method increasing precision, WEBF1 is close to the baseline and better than the other two methods. WEBF1 increases recall and therefore outperforms all the methods on F1, including the baseline. While it is feasible to increase precision with WEBF1 above the baseline and close to MI, recall drops below TF-IDF. *Technical Analysis*

5.4.1.3 *Features Removed*

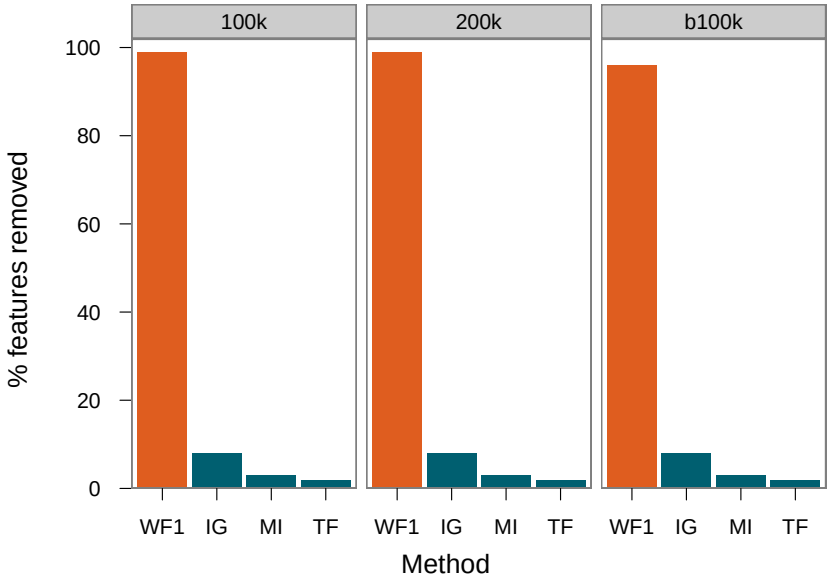


Figure 51: Comparison of the number of removed features.

Introduces Figure 51.

Motivation After comparing WEBF1 to other feature selection methods for its classification performance, we compare how many features each of those methods are removing for their best classification result.

Legend The y-axis shows the percentage of features removed by the various methods compared to the baseline. The x-axis shows the four methods, WEBF1 (WF1), MI (MI), IG (IG), TF-IDF (TF), grouped by the three corpora. The experiment chosen for each method is the experiment with the best classification result.

Technical Analysis WEBF1 removes approximately 99% of the features from the baseline in the 100k corpus, 97% in the 200k corpus and 94% in the b100k corpus. The three other methods all remove less than 10% of the features, with TF-IDF removing only 2-3%. TF-IDF is also the only of the four methods that cannot classify the corpus better than the baseline.

5.4.1.4 *Performance with same Number of Features*

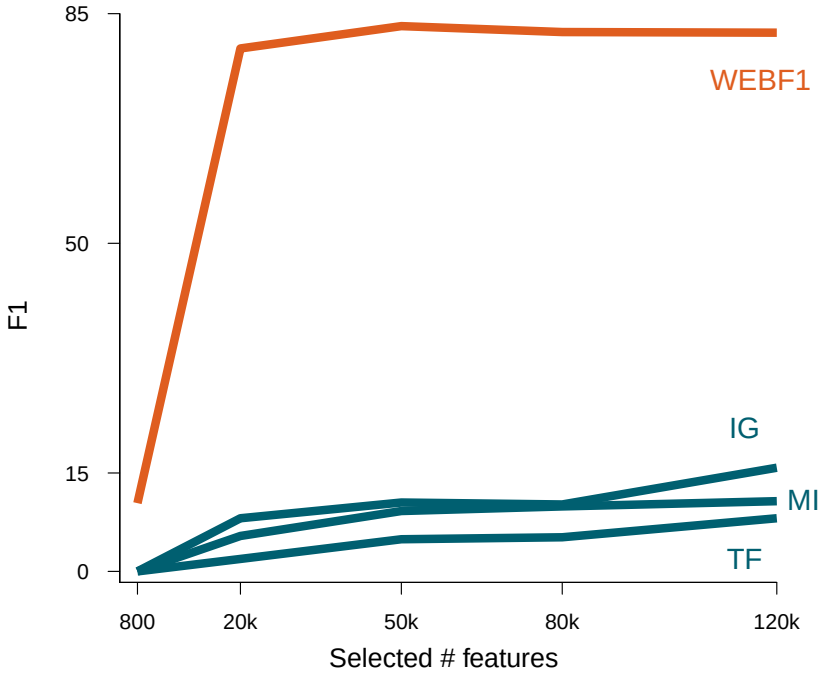


Figure 52: Compares F1 when all are set to the same number of features.

Figure 52. *Introduces*

As previously shown, the feature selection methods choose a different set of features and the size of this set varies as well. In some use-cases it is beneficial to set a specific number as the target for the size of the set. How the methods perform compared to WEBF1 is analyzed. *Motivation*

The 100k corpus is used. Unfortunately WEBF1 cannot select more features easily, hence there is no comparison over 120 000 features. None of the five numbers of selected features represents the optimal number of features for WEBF1 though. *Legend*

The figure shows that the three other methods are not capable of classifying anything when 800 features are selected. Selecting more features does not help WEBF1 to classify the corpus better, since the best result stems from approximately 12 000 selected features. The other methods get better the more features they select, although even with 120 000 features they perform at less than 50%. Being able to classify a corpus as well or better than the baseline with as few as 1% or 2% of the features is a useful capability that seems to be unique to WEBF1. *Technical Analysis*

5.4.1.5 *Classifier Impact*

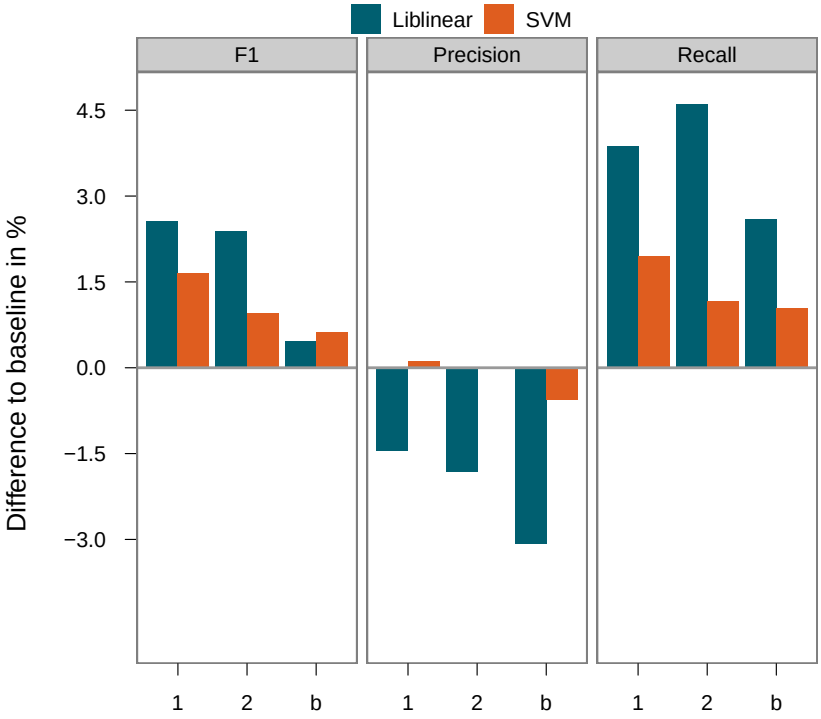


Figure 53: Classifier impact on the performance of WEBF1.

Introduces Figure 53.

Motivation The figure shows the difference between LIBLINEAR and SVM^{light} on the classification performance.

Legend The y-axis shows the difference between WEBF1 and the baseline in percent. A positive value indicates that WEBF1 performs better than the baseline. The x-axis shows the three corpora 100k (1), 200k (2) and b100k (b) grouped by the measures.

Technical Analysis As it was shown in the previous subsection, precision generally drops for WEBF1 compared to the baseline. Though with SVM^{light} the change is within $\pm$ 0.5%, whereas with LIBLINEAR the drop is approximately 3%. The same characteristic can be observed with recall, although in this case WEBF1 performs better than the baseline. SVM^{light} does not improve the baseline as much as LIBLINEAR, at best by 2%, and in addition is always lower than LIBLINEAR. While LIBLINEAR and SVM^{light} do not produce the same improvements to the baseline, the difference is not enough to warrant concern. This might be a simple case of imperfectly chosen parameter settings.

There seems to be no difference for WEBF1 as it does not favor one over the other classifier.

5.4.1.6 *Classifier Impact on Subclass*

	Docs	
	#	%
100k	1100	0.9
200k	2600	1.3

Table 31: Document statistics for subclass F25D.

		LIB			SVM		
		Base	WF1	Diff	Base	WF1	Diff
		%	%	PP	%	%	PP
100k	**F1**	89.5	89.3	+1.0	89.3	90.7	+1.4
	P	95.9	95.4	−1.0	95.4	94.2	−1.2
	R	83.9	83.9	+3.2	83.9	87.5	+3.6
200k	**F1**	94.8	90.2	−0.1	90.2	94.7	+4.5
	P	96.1	96.0	−0.1	96.0	95.2	−0.8
	R	93.5	85.1	+0.6	85.1	93.9	+8.8

Table 32: Performance of subclass F25D on both corpora and classifiers.

Introduces

Tables 31 and 32.

Motivation

We will analyze the impact of the two classifiers on a specific subclass. The absolute classification performance is not being analyzed as this thesis concentrates on the feature selection.

Legend

Subclass F25D is chosen because it represents approximately 1% of the documents. Since 30% of the subclasses contain 1% of the documents, F25D is not an outlier. The subclass is not available in b100k.

Table 31: displays the absolute and relative number of the documents of the subclass in the two corpora.

Table 32: shows F1, precision (P) and recall (R) for LIBLINEAR and SVM^{light}. The baseline (Base) is compared to WEBF1 (WF1) in each corpus. The third column of each classifier shows the difference between the baseline and WEBF1 in PP.

Technical Analysis

F1 is close to the baseline and in all four cases, recall increases. This is coherent with what the macro-averaged measures show in both corpora. There the classification performance is within ±1% of the baseline. This indicates that for subclass F25D WEBF1 can identify the less distinguishing features and remove them without any consequences on the classification performance.

5.4.1.7 *Classification Performance on Subclasses*

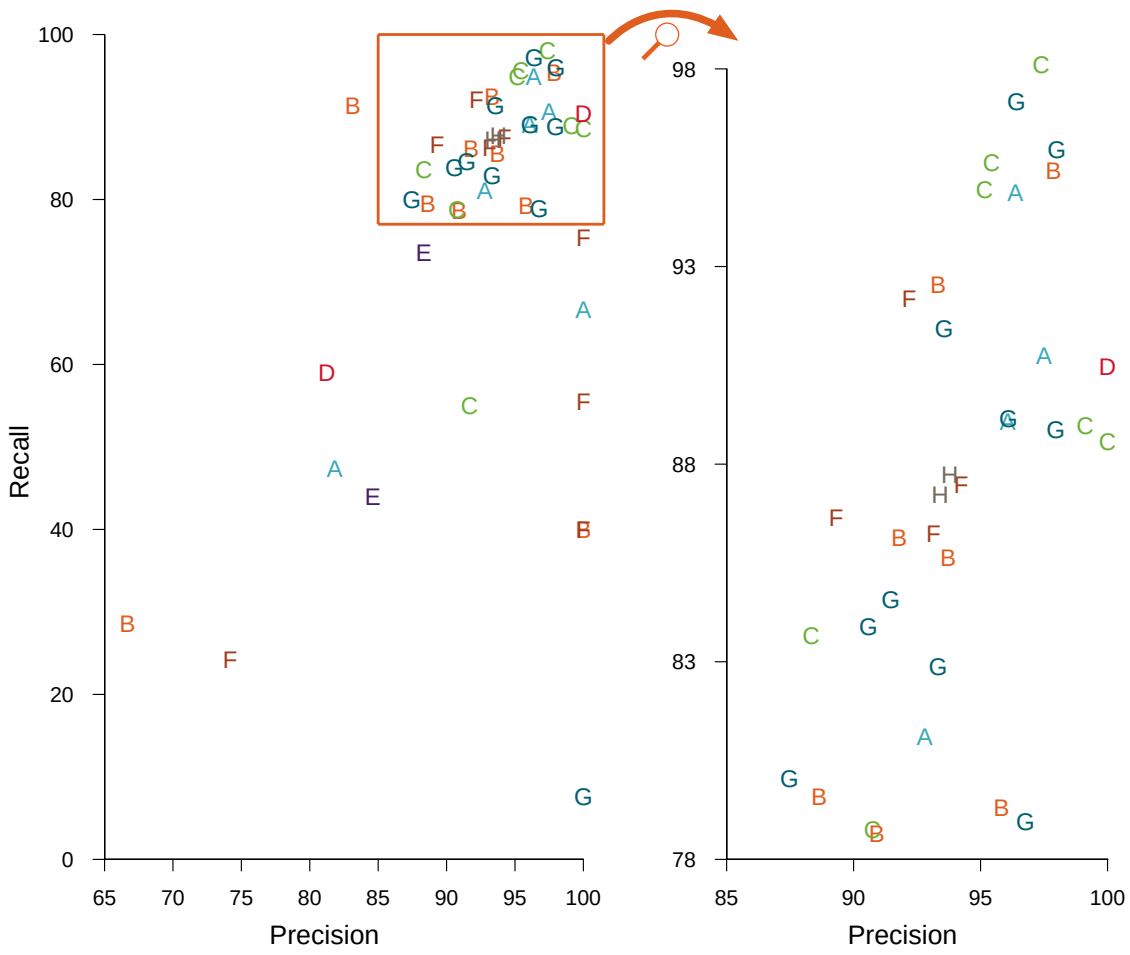


Figure 54: Precision and recall for all subclasses.

Introduces Figure 54.

Motivation While macro-averaged F1, precision or recall gives a big-picture view of the classification performance, there might be outliers far away from the average.

Legend The y-axis shows recall in percent, the x-axis shows precision in percent. The 100k corpus with LIBLINEAR is used. Each character represents a single subclass, the colour and the character denote the section the subclass belongs to. The experiment chosen is the experiment with the highest macro-averaged F1. The figure on the right shows the top right corner of the left figure.

Technical Analysis It confirms that the corpus favors precision. There are also two major facts visualized by the figure. First, the top right corner is highly concentrated, hosting 37 out of 49 subclasses — resulting in 76% of the subclasses. Second, there are eight subclasses performing with exactly or close to 100% precision. Three of those eight are within the five lowest recall values. The only chance to improve F1 is by increasing recall for specific subclasses.

5.4.1.8 *Precision and Recall on Subclasses*

Introduces

Figure 55.

Motivation

The figure gives an overview of how the subclasses react to WEBF1 in all three corpora. F1, precision and recall are compared to the baseline.

Legend

The entire figure compares two variables, the three corpora at the top and F1, precision and recall on the right. The y-axis shows the difference to the baseline in each corpus in percent. Each one of the three rows shows a different measure. Each dot represents a single subclass. The color and shape of the dot shows whether or not WEBF1 is better than the baseline. Their horizontal position has no meaning. The size of the dots has no meaning as well but it helps illustrating lone dots at the top. LIBLINEAR is used.

Technical Analysis

As mentioned before, WEBF1 improves recall in all three corpora when looking at the macro-averaged measures. This is now confirmed on subclass-level. The increase ranges from 1% to approximately 50% per subclass. Only four subclasses reduce their recall value in the 200k corpus, three in both the 100k and b100k corpus.

Precision is the opposite. All but one (!) subclasses in the b100k corpus reduce their precision, 43 out of 49 subclasses reduce in the 100k corpus, 67 out of 75 in the 200k corpus. The range is between 1% and 25%, being significantly smaller than for recall.

This leads to an increase in F1 in around 75% of the subclasses.

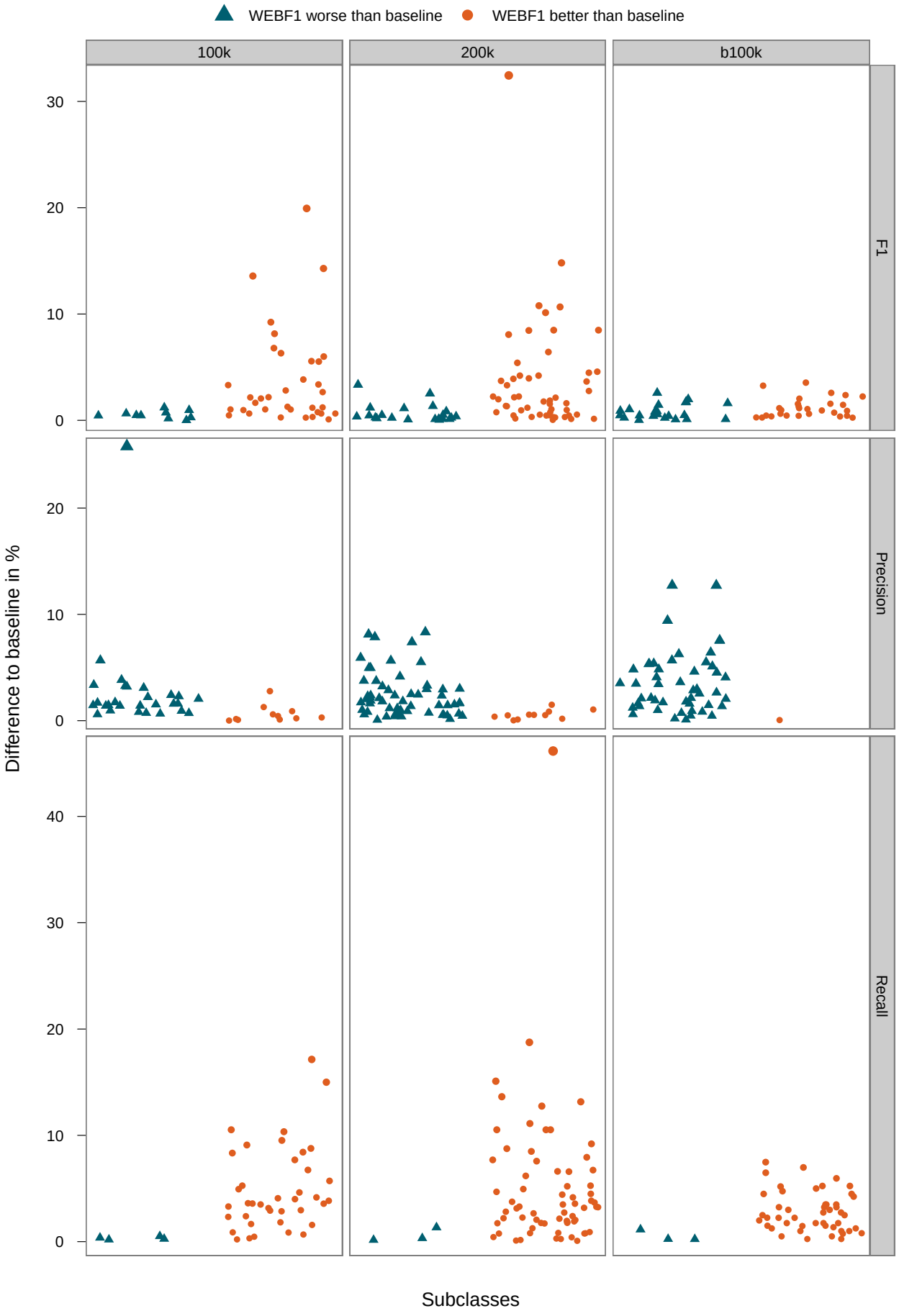


Figure 55: Classification comparison between WEBF1 and baseline on subclasses.

5.4.1.9 *Precision and Recall grouped by Section*

Introduces

Figures 56 and 57.

Motivation

These figures show the classification performance of all the subclasses grouped by their section. This shows whether or not a section might be responsible for a large increase or decrease of a specific measure.

Legend

Figure 56: y-axis shows recall in percent, the x-axis displays precision in percent. Each item represents the macro-averaged value of a section by averaging all the subclasses of that section. The experiment used was the experiment with the highest macro-averaged F1 value for the 100k corpus.

Figure 57: same holds for this figure. Its y-axis shows the macro-averaged F1 values for a subclass and its x-axis subclasses grouped by sections.

Both: LIBLINEAR is used.

Technical Analysis

None of the sections are responsible for a significant drop in precision as all of them are within 7% of each other. Although section E has the lowest precision average, its impact on the macro-averaged value is not significant enough because of its relatively small number of documents. The range of recall values is quite big, going from 57% to 90%. Section E is again the worst performing section but the same logic applies. The section with the largest impact on the overall recall is section F. It is the third largest section but more than 10% below the largest subclass. The reason for an increase in macro-averaged recall is section G, the section with the most documents and subclasses, performs at approximately 80%.

This is slightly surprising because section G has a subclass sporting an F1 of only 16%. But since the remaining subclasses perform at approximately 86% or higher, the entire section is responsible for a slight increase in macro-averaged F1 — and as we will see later, also for a significant portion of the feature reduction.

Another interesting characteristic displayed by Figure 57 is the lack of outliers above the median. No subclass pushes the median up. On the other hand, there are a total of five outliers below the median. Too few to drag the median of a section down.

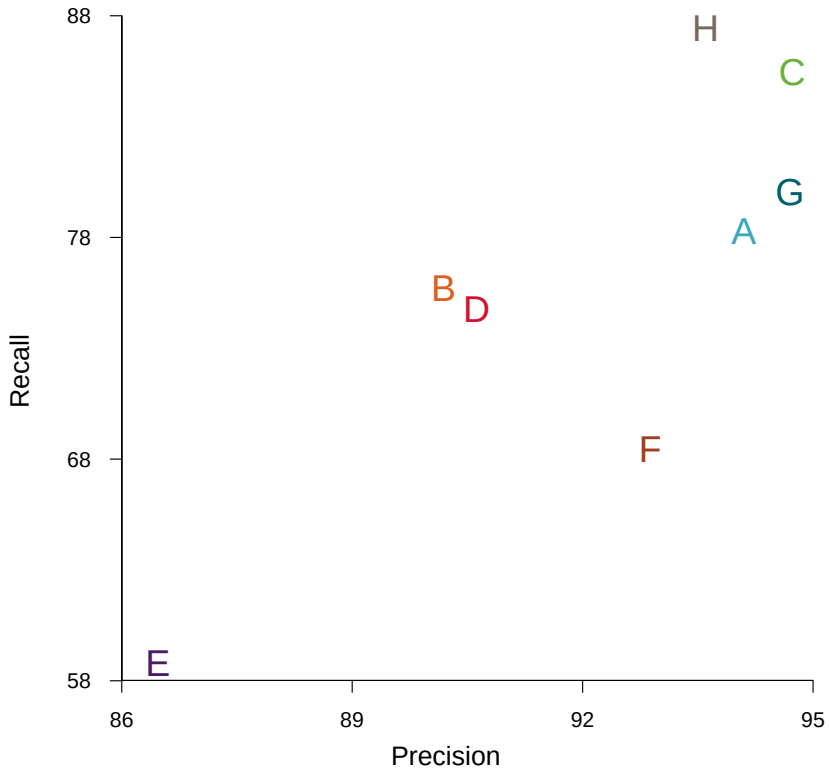


Figure 56: Precision and recall for macro-averaged sections.

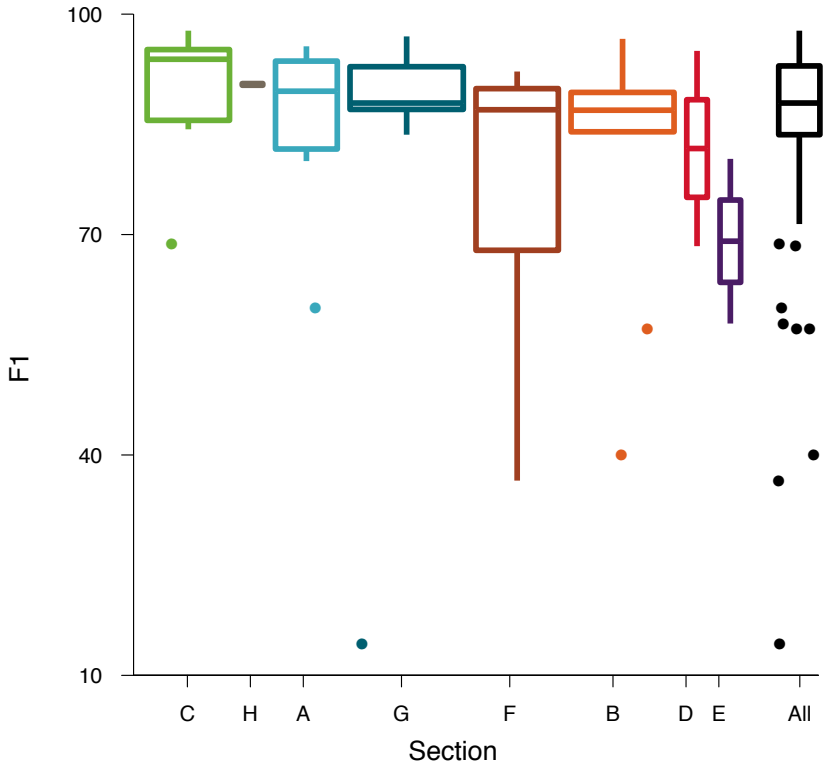


Figure 57: F1 for every subclass grouped by sections.

5.4.1.10 *Performance Analysis by Size*

Introduces

Figures 58 and 59.

Motivation

Whether or not the number of documents of a subclass influences its classification performance is an important information to have when the performance of specific subclasses needs to be improved. This will be analyzed in the following two figures.

Legend

Both: The y-axis shows the value of the three measures in percent. The x-axis shows the five bins grouped by F1, precision and recall in descending order by the medians. All 49 subclasses from the 100k corpus are split into five bins. Each bin contains ten subclasses (except for bin 5, containing subclasses with the most documents, which has only nine). Subclasses are sorted by the number of documents they are assigned to. The first ten subclasses with the fewest documents are in bin 1. Bin 2 contains the subclasses which have the 11th fewest documents up to the subclass with the 20th fewest documents. This proceeds till all subclasses are in a bin. In the 200k corpus, there are 75 subclasses split into 5 bins. Each bin contains exactly 15 subclasses. The method of the splitting is the same as for the 100k corpus. Both corpora use LIBLINEAR .

Figure 58: shows the 100k corpus.

Figure 59: shows the 200k corpus.

Technical Analysis

In both corpora the medians of the F1 values for bins 1 to 4 are within 3% of each other. In the 100k corpus only the median of bin 5 is approximately 10% higher than the one of bin 1. In the 200k corpus it is similar, although the median is only 7% higher. The medians of precision in the 200k corpus are very close together, only 3% separate the highest from the lowest. In the 100k corpus the highest median is found in bin 1. This is the only time that bin 5 is not the highest in the three measures for both corpora. Recall in the 100k corpus is approximately 23% higher in bin 5 than in bin 1. The differences for recall in the 200k corpus are not as high, although bin 5 is again the best performing bin, being 7% higher than bin 1 and 2. Generally the bin size cannot be used to estimate the classification performance beforehand. The medians of the bins 1 to 4 are always very close together.

Bin 5 is the exception as it almost always outperforms the other bins. It seems that only if the number of documents in a subclass is extraordinary high (i.e. over 20 000), it will indeed help predetermining the classification performance. One subclass having a few more documents than another subclass is not enough.

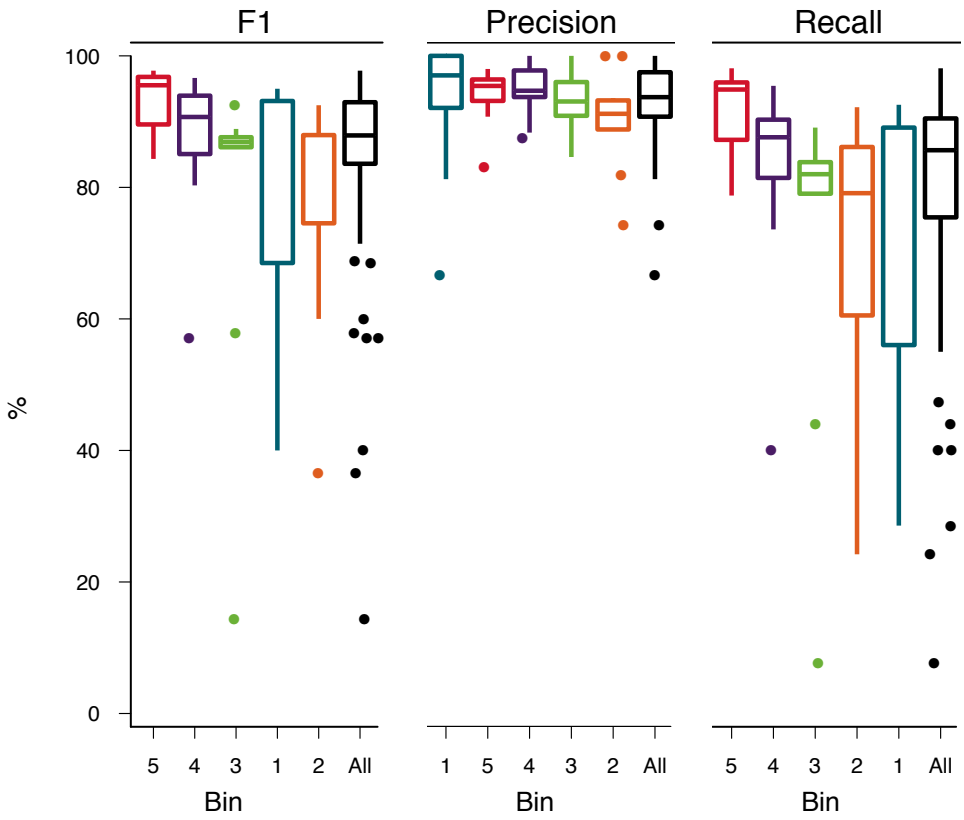


Figure 58: 100k.

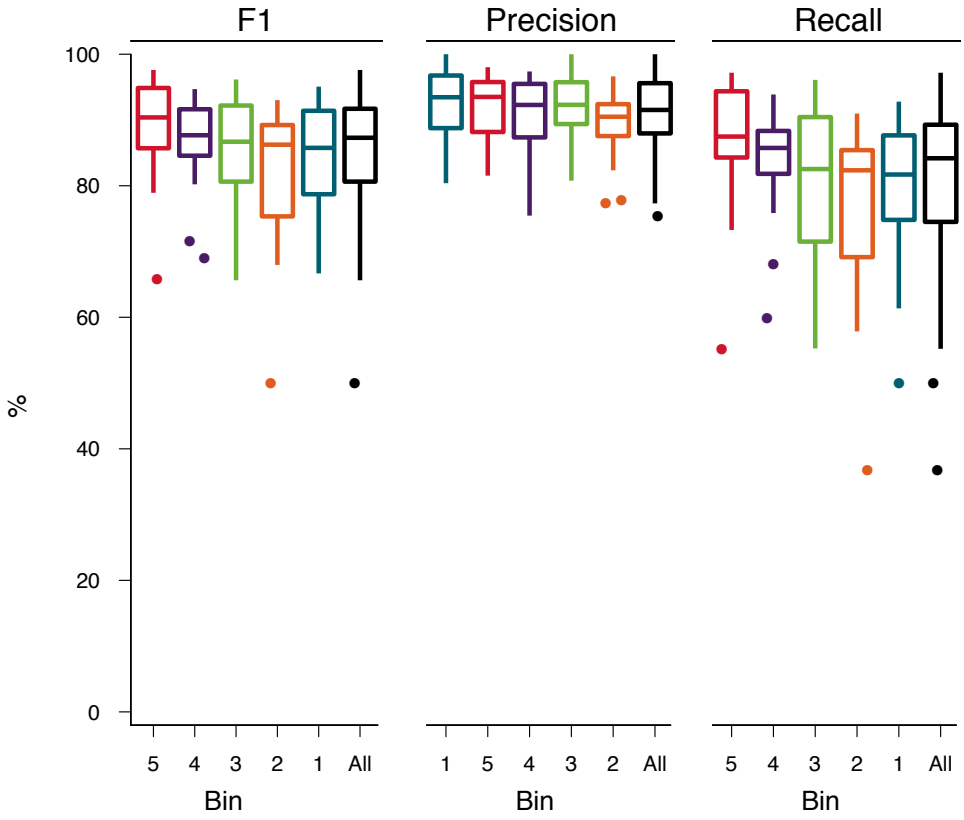


Figure 59: 200k.

5.4.1.11 *Micro-average Results*

		WEBF1	Base	Diff
		%	%	PP
F1	100k	89.8	92	−2.2
	200k	89.9	90	+0.1
Precision	100k	95.2	96.3	−1.1
	200k	92.2	93.8	−1.6
Recall	100k	84.9	89.1	−4.2
	200k	87.8	86.5	+1.3

Table 33: Micro-average comparison between WEBF1 and baseline.

Table 33. *Introduces*

While this thesis concentrates on macro-averages, this table shows the best micro-averaged classification performance. This gives the reader a rudimentary understanding of how WEBF1 performs when all subclasses are prioritized the same. This is also the only presentation of micro-averaged results in the entire WEBF1 section. *Motivation*

The difference column shows PP between WEBF1 and the baseline. A positive value indicates that WEBF1 is better than the baseline. LIBLINEAR is used but SVM^{light} behaves similarly. Only 100k and 200k are shown. b100k is neglected since for this corpus micro- and macro-averages are the same. Why? b100k has exactly 2000 documents assigned to each of the 50 subclasses. Hence each subclass weighs in with exactly 2%. This holds for both micro- and macro-averages. *Legend*

The absolute values for F1, precision and recall are higher than for macro-averages for both WEBF1 and the baseline. The main difference to the macro-averaged results is the loss in performance of the 100k corpus. While it improves macro-averaged recall, it loses around 4% for micro-averaged recall. A possible reason for such a behavior might be the before mentioned highly uneven distribution of the 100k corpus. *Technical Analysis*

In the 200k corpus recall improves, though still not as much as with macro-averages. In both corpora precision remains a weakness of WEBF1. A reason for this might be because the values are already high and increasing it further is more difficult. Since a classifier can hardly classify a corpus perfectly, increasing a precision baseline yielding 96% is beyond the capabilities of WEBF1.

5.4.2 Parameter Analysis

5.4.2.1 *Impact of s on Macro-F1*

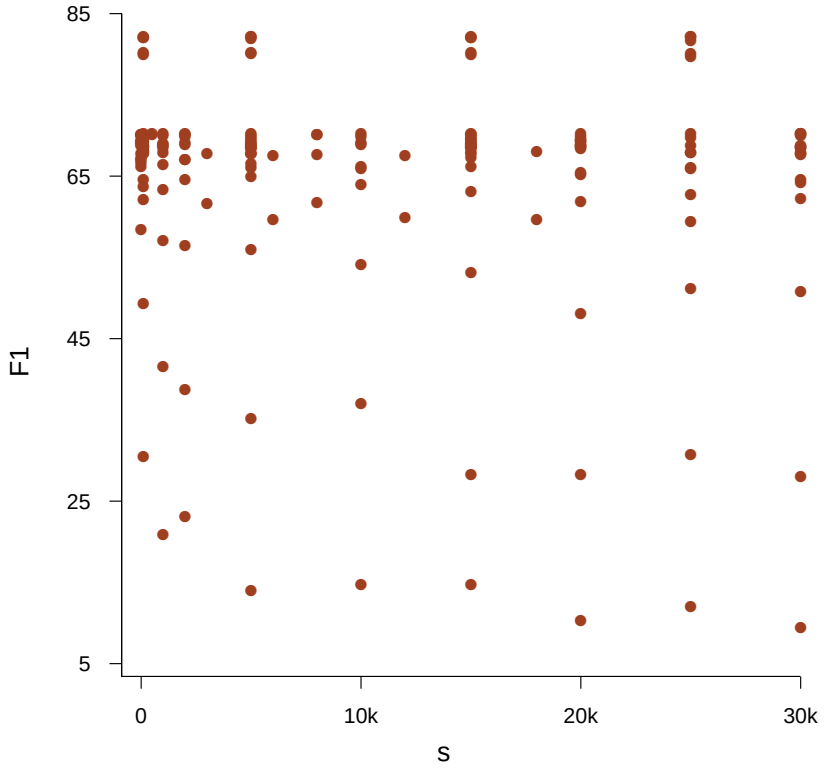


Figure 60: Impact of parameter s on macro-averaged F1 for WEBF1. Each dot represents a different experiment.

Introduces Figure 60.

Motivation Parameter s is one of two parameters provided by WEBF1. The very wide range of possible values for parameter s will make a grid search necessary and if one has to test all of these the grid search will be severe. Therefore the impact of s on the macro-averaged F1 is of high importance.

Legend The x-axis shows the parameter s set to values between 100 and 30 000 on the 100k corpus. The y-axis shows the macro-averaged F1 value for each experiment using LIBLINEAR . A dot represents a single experiment. Each experiment has a fixed set of parameters for the classifier and only the two parameters of WEBF1, s and α, change. For each setting of s (e.g. $s = 10000$) the only parameter changing is α.

Technical Analysis The figure clearly shows that parameter s has little to do with the classification performance of an experiment, at least when measured by macro-averaged F1. If s is fixed to a certain value (e.g. $s = 10000$), there are tens of experiments with different F1 values. Some experiments are as low as 10% and as high as 83%.

5.4.2.2 *Impact of alpha on Macro-F1*

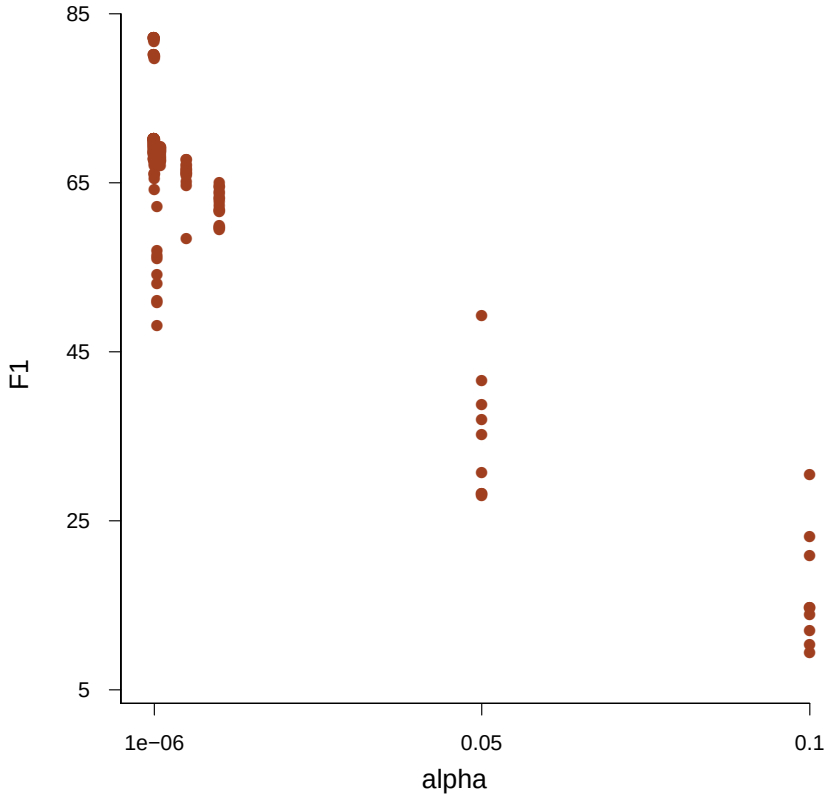


Figure 61: Impact of parameter α on macro-averaged F1 for WEBF1. Each dot represents a different experiment.

Figure 61. *Introduces*

This analyzes the impact of α on the macro-averaged F1. *Motivation*

The x-axis shows parameter α from $1x10^{-06}$ to 0.1. A dot represents a single experiment. The y-axis shows the macro-averaged F1 value for every experiment. As before, the set of parameters is fixed, only two parameters change for a dot, α and s. This means for a specific α value in the figure, only s changes. The data comes from the 100k corpus with LIBLINEAR . *Legend*

The lower the parameter α, the more features are kept and the better the classification performance becomes. This parameter is solely responsible for achieving good classification results. Once α is set to a reasonable value (e.g. 0.000 05), F1 increases and the fine-tuning begins. It is done by finding the best possible value for s. s changes a lot on subclass level, more than fine-tuning α would. This reduces the need for an intense grid search, as one finds a well performing value for α and only for this single value a search for s is done. Otherwise every combination of s and α would have to be tried. *Technical Analysis*

5.4.2.3 *Impact of s on Subclass-Level*

Introduces Figure 62.

Motivation In the previous subsection it was shown that the impact of parameter s on macro-averaged F1 is not big. Therefore we are going to check the differences on subclass level. This is especially important if the performance of specific subclasses has to be improved.

Legend Rows show the measures for every subclass. Columns show the different s settings. The first shows $s = 1000$, the second shows $s = 25000$, both on the 100k corpus. The y-axis shows the difference to the baseline. Their horizontal position is not important. The baseline is the best performing experiment for $s = 5000$. Each dot represents a subclass. The colors show whether or not the s-setting is better than the baseline. For each of the three s-settings the highest classification performing experiment was chosen. All three measures — F1, precision and recall — are taken from the same experiment.

Technical Analysis The changes in precision are overall smaller, ranging from -5% to 10%, whereas in recall they range from -10% to 17%. Changes and their magnitudes do not appear to have anything to do with the number of documents assigned to a subclass. Subclasses with thousands of documents are as likely to change as subclasses with only a few hundred. Why there is such a large span of changes for recall is unknown at this point.

Furthermore 41 out of 49 subclasses do not share the same change for both $s = 1000$ and $s = 25000$. While a subclass might change by -3% for $s = 1000$, the same subclass might change by 8% for $s = 25000$. While this has little or no impact on the macro-averaged measures, it is interesting for those who deploy an ensemble classification approach. Here each subclass might be trained by a different index, which stems from WEBF1 but with different settings for s.

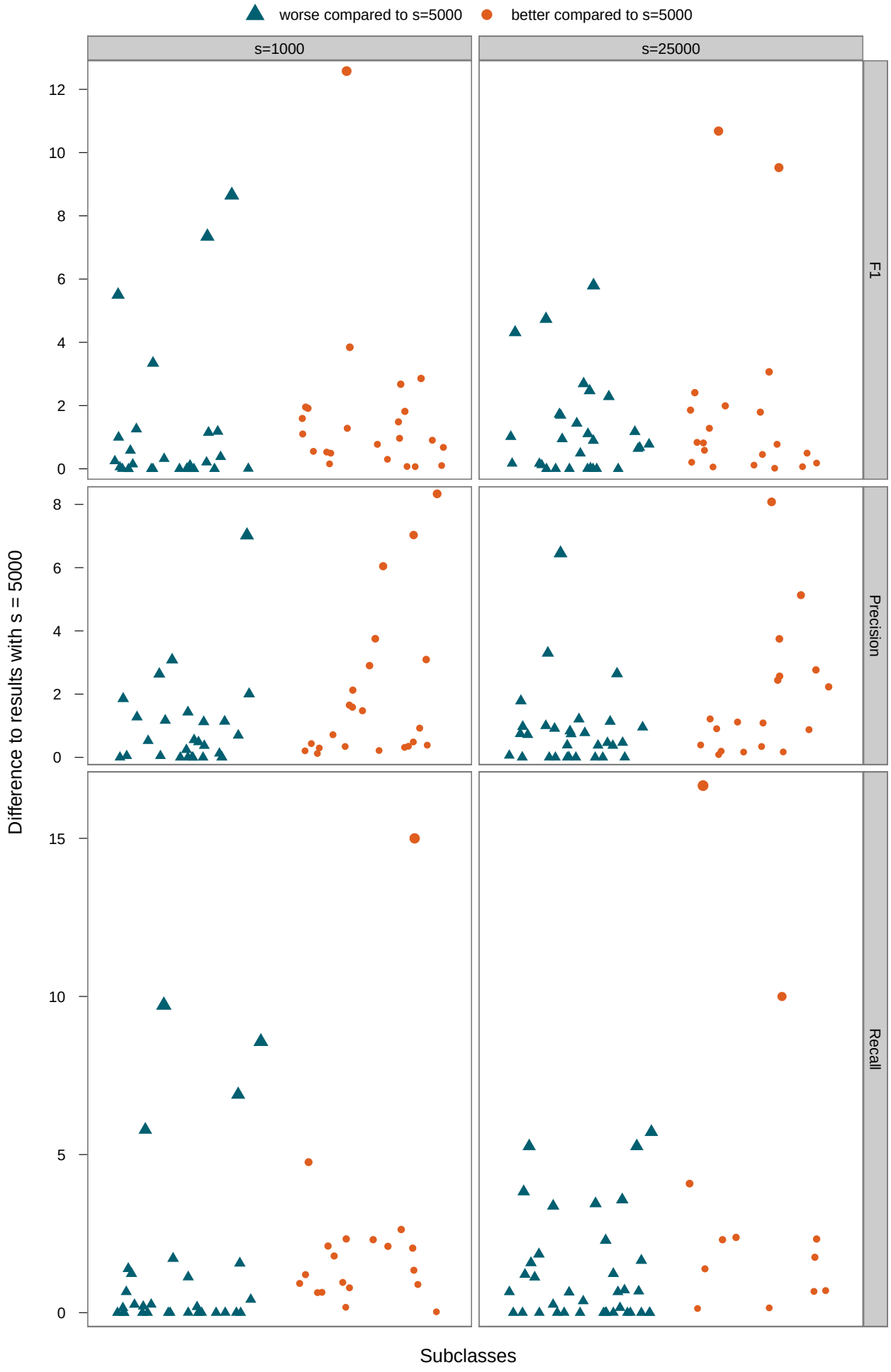


Figure 62: Classification performance values for three different settings for parameter s on the corpus 100k.

5.4.2.4 *Impact of s on the Number of Features*

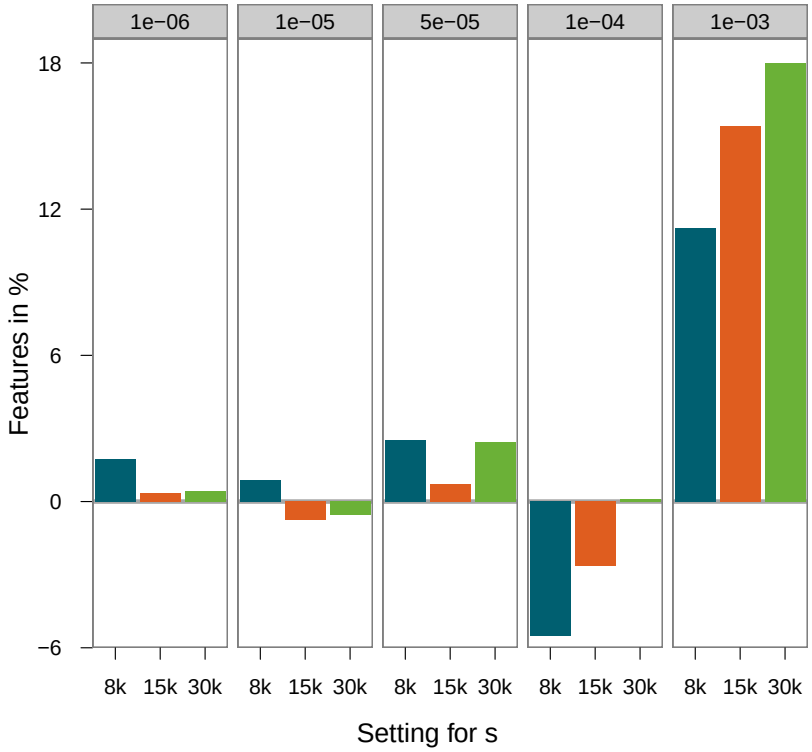


Figure 63: Percentage variance of the number of features from s.

Introduces Figure 63.

Motivation While parameter s does not have a high impact on the macro-averaged F1, this section investigates the impact of s on the number of selected features.

Legend The highest classification performing experiment for $s = 100$ is chosen as the baseline for each of the five α settings shown at the top of the x-axis. The y-axis shows the difference in percent in the number of selected features between the baseline and three different s settings grouped by five different thresholds. A negative value means the baseline selected more features. The 100k corpus is used.

Technical Analysis The quintessence is: The smaller the threshold, the less variance parameter s introduces on the number of selected features. With a threshold of 0.001, the difference between the baseline and any other s setting is at least 12%. In addition the difference between the other three s settings is also higher than any other difference for the smallest tested threshold. Hence it is impossible to determine beforehand which s setting results in the most selected features for a specific threshold setting.

5.4.3 Feature Analysis

5.4.3.1 *Selected Features Showcasing*

Alkanolamides, alloys, another, antibody, antidiabetic
Bacon, barbers, basculature, bubbled
Cardiovasular, chrysogenum, compute, corn
Decorations, dente, detergents, dialkanol, dopamine, dynamic
Each, erect, exploded, explosion, extinction
Formulation, fortis, **fourth**
Glycol, **guided**, gradually, grippable
Holo, hybridized, hydrolyzed
Incubating, intellectual, iwans
Laparoscopy, **large**, leaf, lower
Magnesium, meat, **meet**, multiplex
Naked, neovasculature, noiva, **north**
Object, opaque, operations, organics
Particular, patch, patency, payment, precedents, producers
Quadrants, questionable, quiesce
Radiolabels, retriggerable, retroreflection
Savings, sexually, slope, sloped, smears, subassemblies
Topographies, trophic, transflector, trapezoid
Unmethylated, unsaturations, **unused**
Viewers, virtualized, volumes, vout
Waveband, wedgingly, weights, wobbles

Table 34: Small portion of the tokens on the 100k corpus.

Introduces

Table 34, references Figure 52.

Motivation

Showing all 11 852 selected features of an experiment is infeasible. A small partition is shown and analyzed.

Legend

Tokens in **bold** are outliers, linguistically judged as words that carry little information. The data comes from the 100k corpus and an experiment that selected 11 852 tokens.

Technical Analysis

WEBF1 keeps only one to four percent of the features. This is the reason why WEBF1 cannot remove those outliers in **bold** as otherwise some documents would be completely empty. While they might not be discriminative enough to be good enough on their own, they at least give the classifier a chance of classifying. Classifying a completely empty document is impossible. This was shown in Figure 52.

5.4.3.2 *Selected Features on Subclass-Level*

	C07K			G06F		
	F1	**UF**	**# F**	**F1**	**UF**	**# F**
	%		k	%		k
Base	84	339 000	453	81	291 000	398
WEBF1	84	**28**	35	82	**9**	37
Diff %	0	-99.99	-92.22	+1	-99.99	-90.64

Table 35: Performance of two subclasses for the baseline and WEBF1.

Introduces Figures 64 and 65, Table 35.

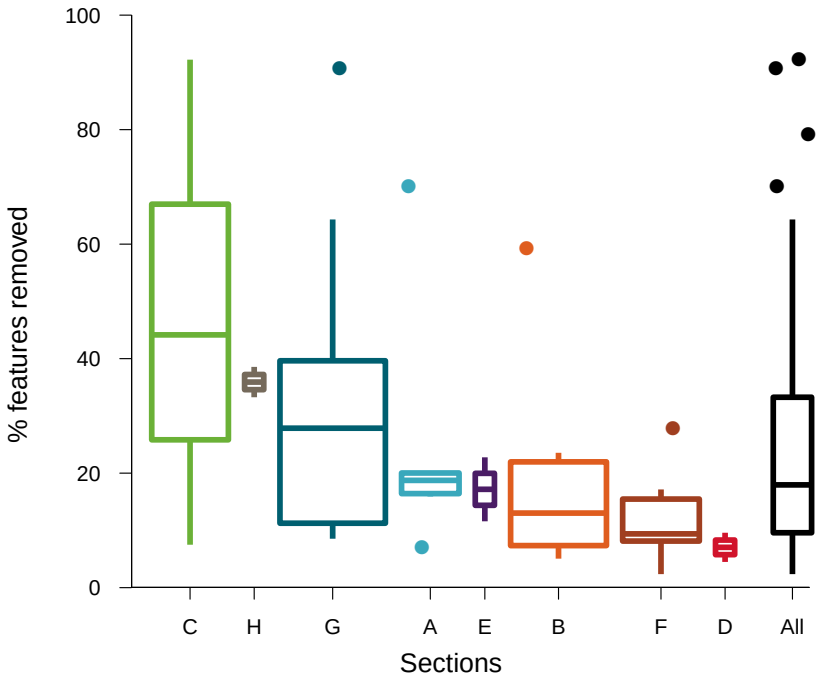
Motivation While a total reduction of 95% or more sounds impressive, it is interesting to know which subclasses are responsible for such a significant reduction.

Legend **Figure 64**: Shows the drop of the number features in percent on the y-axis and the subclasses on the x-axis grouped by section.

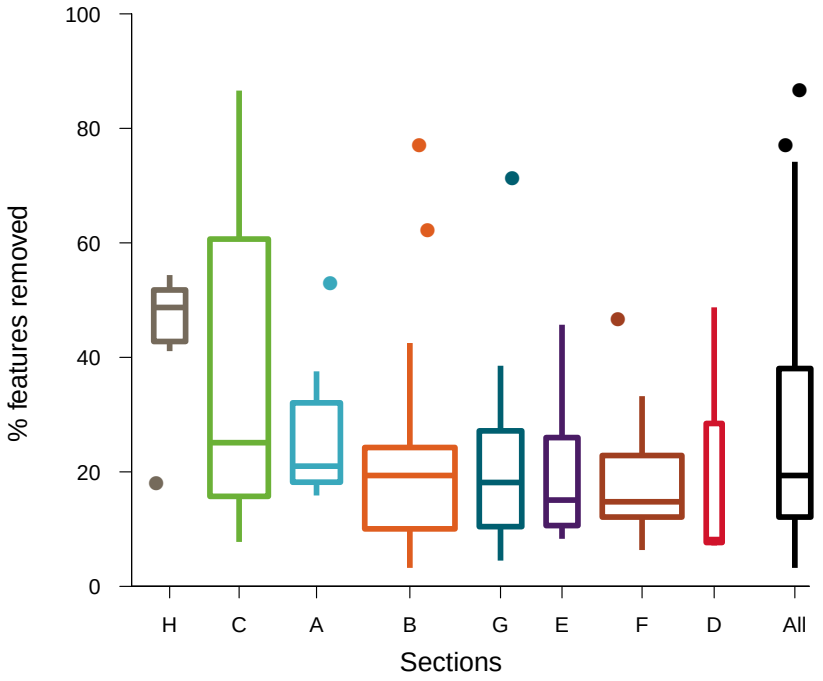
Figure 65: The y-axis shows the number of features in a subclass, the x-axis shows the subclasses colored by their sections and in descending order by the relative drop. Each line represents one subclass, the top of the line shows the number of features for the baseline, the bottom the number of features for WEBF1.

Table: The table shows the top two outliers for the 100k corpus, subclasses C07K and G06F. Column UF shows the number of corpus-wide unique features in the subclass, the third column shows the total number of features in the subclass.

Technical Analysis Most of the reduction stems from subclasses that have plenty of features in the baseline, most notably those which have more than 25 000 features. This is true in both corpora and seems intuitive. It is very difficult to remove 99% of the features from a subclass that contains only 4 000 features. This would lead to many empty documents and therefore the classification would be impossible. The two top outliers posses approximately 300 000 unique features and are therefore responsible for roughly 57% of all the unique features. WEBF1 is capable of removing 99.99% of the unique features in both subclasses without sacrificing the classification performance. When such a large partition of the unique features is removed, overfitting is a concern. Despite F1 remaining close to the baseline, overfitting cannot be completely dismissed. Even if it stems from only a handful of subclasses.



(a) 100k



(b) 200k

Figure 64: Percentage of selected features compared to baseline.

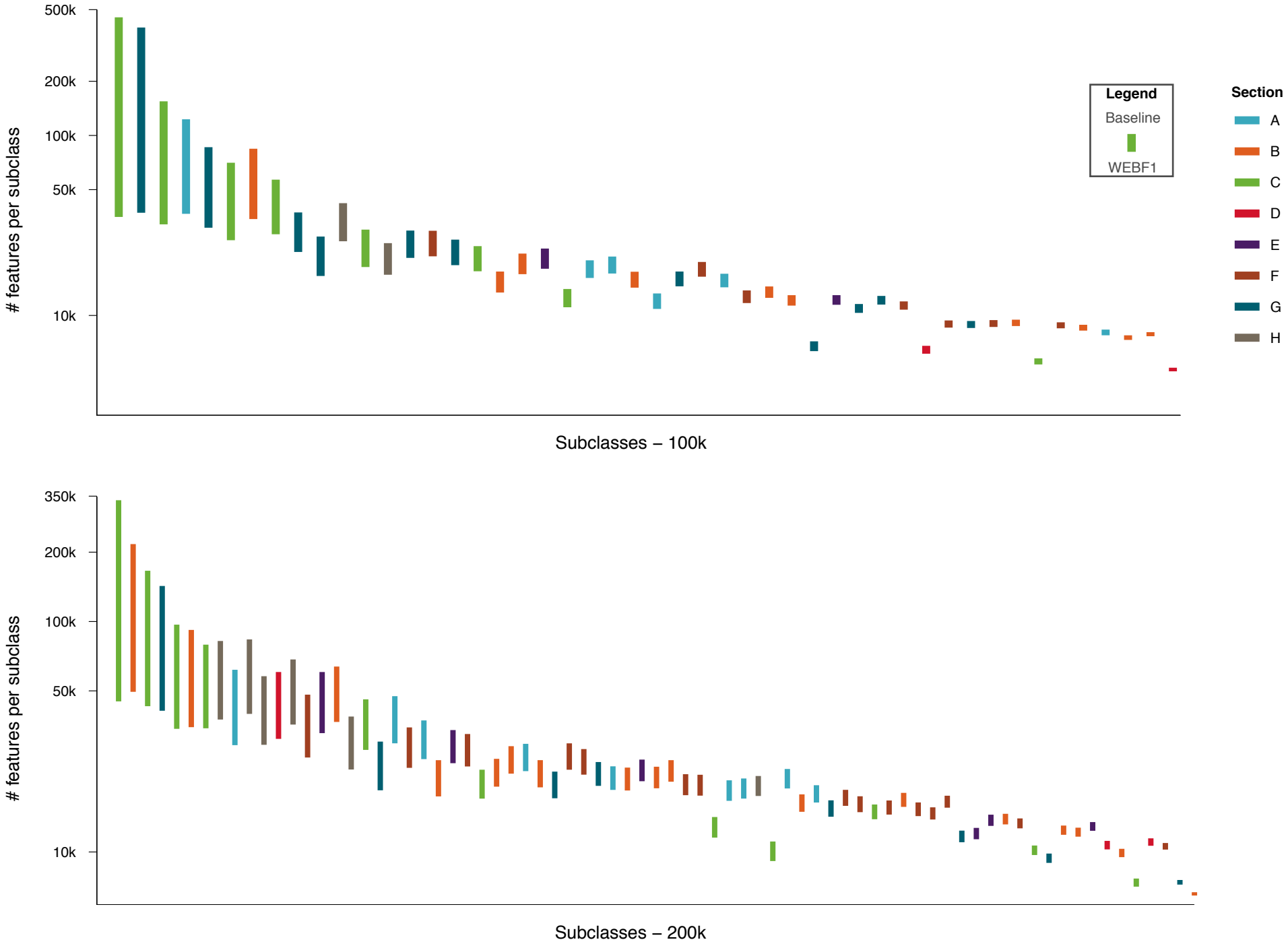


Figure 65: Number of features per subclass for the baseline and WEBF1. In descending order by relative reduction.

5.4.3.3 *Unique Features per Subclass*

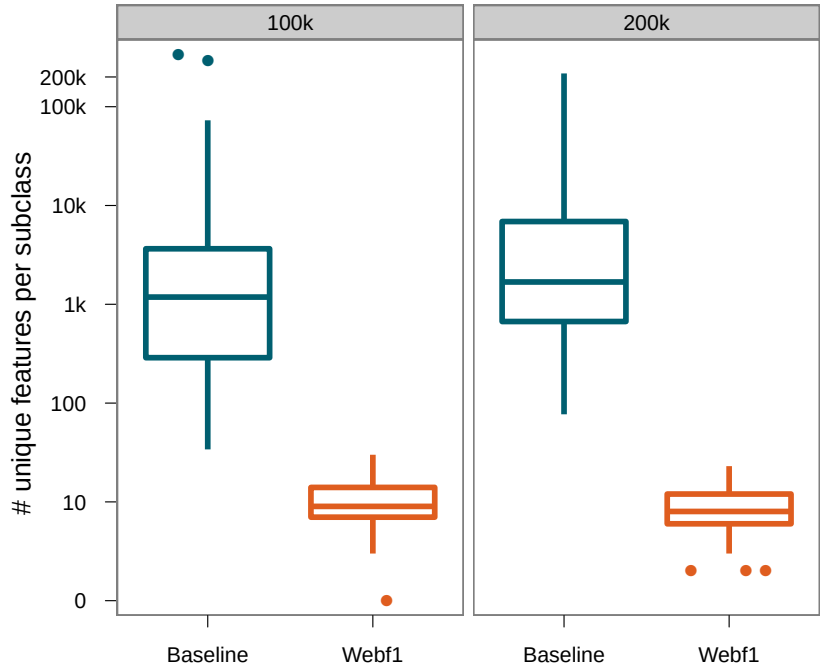


Figure 66: Unique features per subclass.

Figure 66. *Introduces*

Corpus-wide unique features should in theory be important to distinguish subclasses from each other. The figure shows the change in the number of unique features between the baseline and WEBF1. *Motivation*

The y-axis shows the number of unique features in a subclass grouped by the corpus. The underlying data comes from the best performing classification run of both the baseline and WEBF1. This means that the classification performance of the run used to show WEBF1 in the two figures is higher than the baseline. *Legend*

On the 100k corpus the median of the unique features per subclass gets reduced to nine. In the baseline it is approximately 2700, a reduction of 99.7%. The subclass with the highest number of unique features gets reduced from over 350 000 to 30. On the 200k corpus the median of the unique features per subclass is reduced from 1200 to seven. The highest outlier is reduced from over 200 000 to 26 unique features. Despite the reduction of the unique features to basically zero, the classification performance is the same. This might indicate that a linear classifier cannot use those features to distinguish subclasses from each other, which is plausible when those unique features do not occur often. *Technical Analysis*

5.4.3.4 *Unique Features per Document*

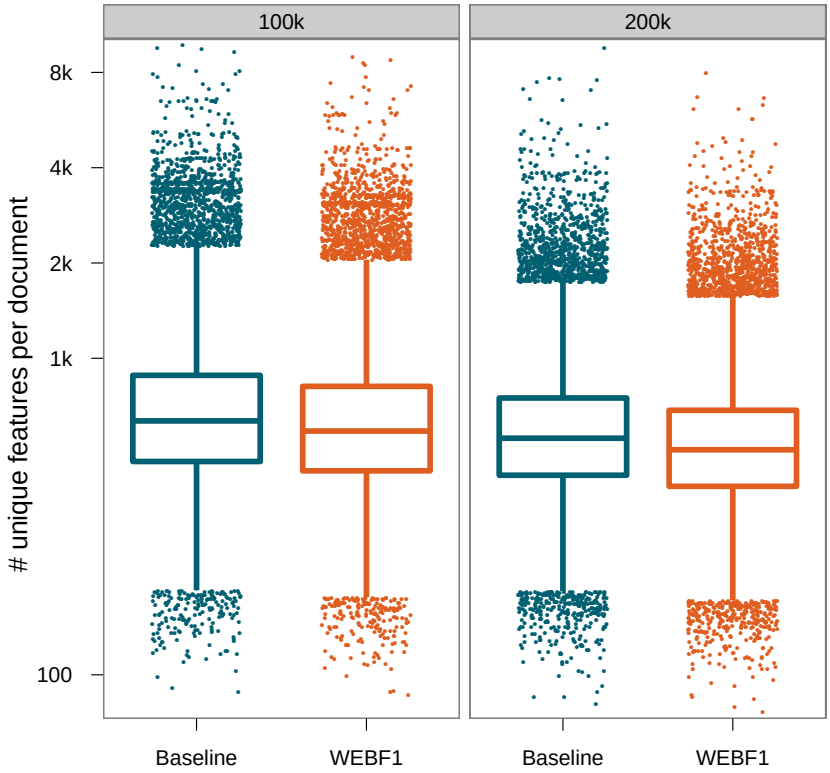


Figure 67: Unique features per document.

Introduces Figure 67.

Motivation The number of unique features in a document is important for various reasons. Among others, the fewer unique features in a document, the less resources are needed to train (smaller files being written to the disk/read from the disk, less memory used to keep documents in the RAM, ...).

Legend The y-axis shows the number of unique features per document. The x-axis shows the method grouped by the corpus.

Technical Analysis On the 100k corpus the median of the number of unique features per document is reduced from 783 in the baseline to 749 with WEBF1 and from 638 to 622 on the 200k corpus. This indicates that the removed unique features are not occurring in that many documents.

5.4.3.5 *Sparseness*

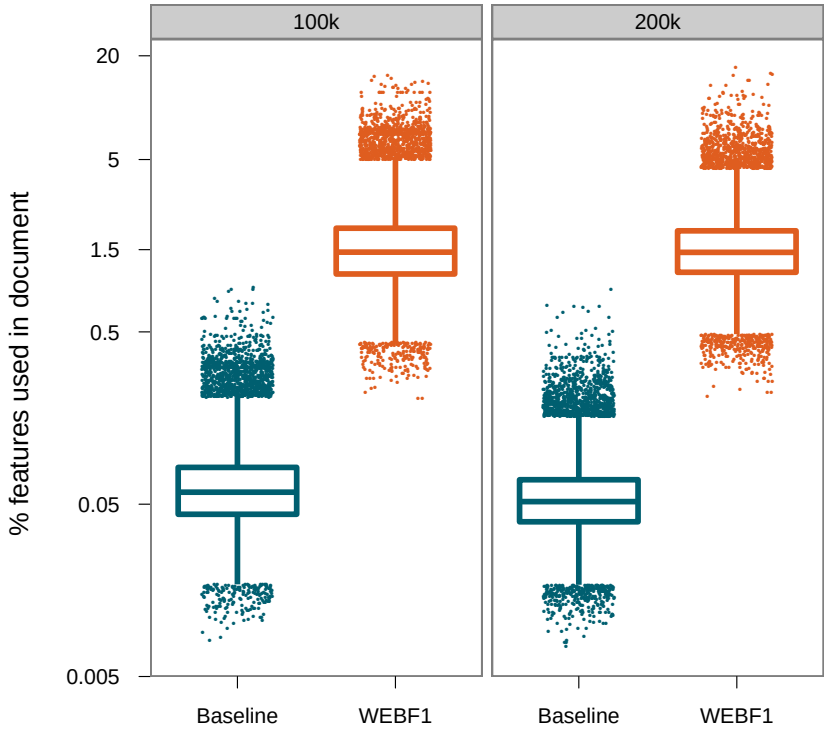


Figure 68: Sparseness of a document.

Figure 68. *Introduces*

Sparseness is important because many algorithms that perform well on dense data structures are not necessarily performing as well on sparse data structures. In addition with sparse data, applying statistical language modeling is more difficult, as one might be forced to use smoothing with low frequency words. *Motivation*

The y-axis shows the percentage of unique features in a document compared to the total number of unique features in the corpus. The x-axis shows the method grouped by the corpus. *Legend*

Sparseness decreases in both corpora. The median of the baseline in both corpora is approximately 0.05%. In the 100k corpus WEBF1 increases the sparseness to 1.7% and in the 200k it increases to 1.64%. Outliers increase from 1.7% to 16.4% in the 100k corpus and from 0.8% to 17.3%. This means there are documents that contain approximately every sixth word of the corpus, which is one step closer to solving the sparse data problem. Even the other documents increase by a factor of approximately 35. *Technical Analysis*

There is not a single outlier with WEBF1 as low as the median of the baseline.

5.4.4 Runtime Analysis

5.4.4.1 *CPU Runtime*

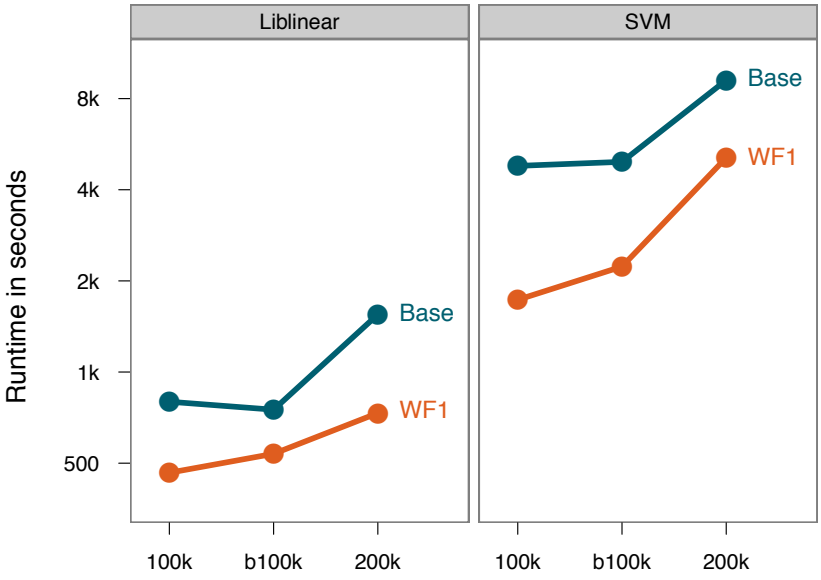


Figure 69: Comparison of runtime between classifiers.

Introduces Figure 69.

Motivation The reduction of features removes words from a document. Shorter documents and fewer overall features should shorten the runtime of the classifier during the training phase. This analysis shows how much time is saved by doing feature selection prior to the training phase.

Legend The x-axis shows the three corpora, the y-axis shows the CPU-runtime of the classification process in seconds. The classification process also includes reading and parsing the training and test files. The baseline (Base) and WEBF1 (WF1) are shown. Runtime results are measured with 10 cores on the MDC. For WEBF1 the highest classification performing experiment was chosen.

Technical Analysis In the baseline LIBLINEAR is approximately five times faster than SVM^{light}. The 200k corpus takes about twice as much time as the 100k corpus. This gap increases with WEBF1, especially when used in conjunction with SVM^{light}. After applying WEBF1, the runtime is approximately reduced to half of what it is with the baseline. The difference between 100k and 200k increases from 200% to 350% after WEBF1 is applied, clearly showing the benefit of using WEBF1.

5.4.4.2 *Feature Decline*

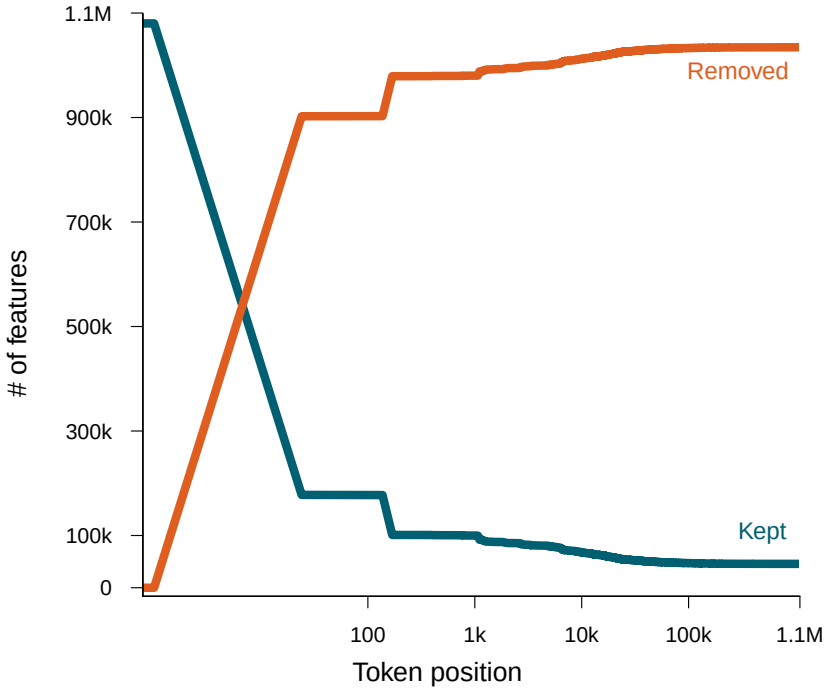


Figure 70: Decline in the number of features kept during WEBF1.

Figure 70. *Introduces*

As shown in Algorithm 1, once a feature is removed during the WEBF1 process, it is skipped later on in the feature pair comparisons. Therefore the number of features in each feature pair comparison is directly related to the CPU runtime and an analyzes of it is in order. This should not be confounded with the number of feature comparisons, which will be shown later. *Motivation*

The x-axis shows the token positions. The list of features is sorted by the order of extraction. Consequently each feature is mapped to a single token position. The y-axis shows the number of features kept for a specific token position. The data comes from the 100k corpus and from one of the best performing experiments. *Legend*

After approximately the 100 000th feature, the number of kept features is stable and only an additional 5 000 features is removed. The decline is steep in the beginning, removing hundreds of thousands of features within the first 1 000 features. This indicates often occurring, completely redundant features in the beginning, functional words for example. *Technical Analysis*

5.4.4.3 *Feature Comparisons*

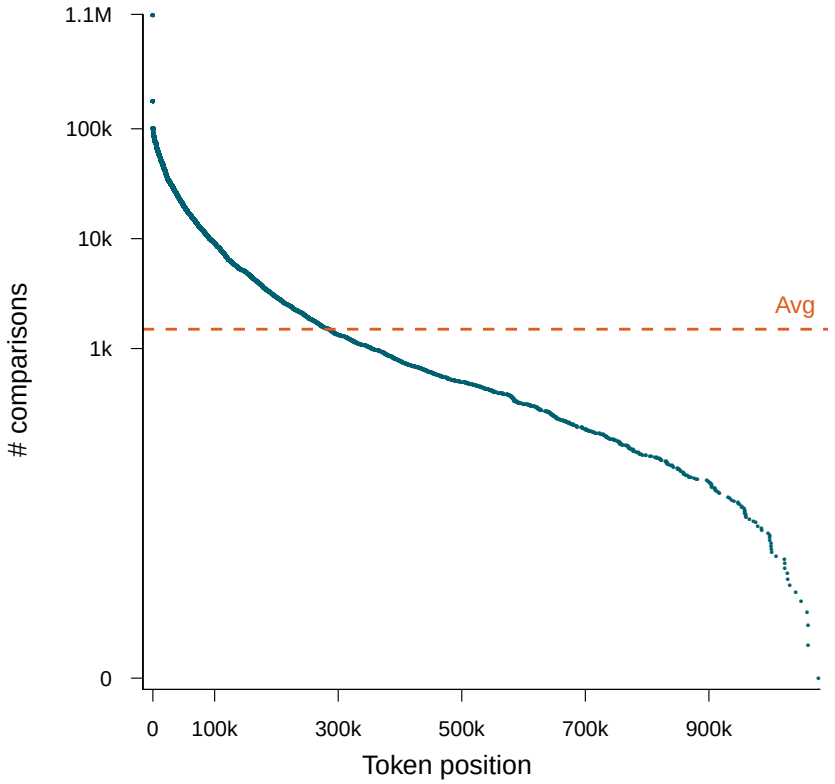


Figure 71: Number of comparisons for every feature.

Introduces Figure 71.

Motivation As previously mentioned, the number of feature comparisons is important for the runtime. With 1 million features, a comparison of all the feature pairs would result in approximately 500 billion comparisons.

Legend The x-axis is the token position, defined in Subsection 5.4.4.2. The y-axis is the number of feature pair comparisons for a particular token position. The 100k corpus is used and the chosen experiment is the experiment with the highest classification performance. The dashed line is the average number of comparisons per feature.

Technical Analysis The first one hundred features are compared to almost all of the features. After those, the decline of the number of comparisons is steep. The last 800 000 features are compared to only a maximum of 6 000. The white tail symbolizes the skipping of features. This seems intuitive since the features are ordered by their extraction points. Features, which are extracted last, only occur in a few documents and subclasses. Their MMV within a subclass might be high (though highly unlikely) but being in only a few documents/subclasses makes it more likely that the sum of differences falls below α.

5.4.4.4 *Algorithm CPU Runtime*

This subsection should show the actual CPU runtime of the algorithm. Since the actual runtime depends on the implementation and hardware resources, it is hardly useful on its own. Therefore it is skipped in this section and will be presented in the form of a comparison to WEBF2 in Subsection 5.5.4.2 and to WEBSOL in Subsection 5.6.5.2.

5.5 WORD-ENTROPY BASED FILTER 2

In this section the results of the experiments using WEBF2 are shown. A comparison to WEBF1 will also be provided. As explained before, the number of subclasses have no impact on the threshold since the final value is divided by the number of subclasses the feature is contained in. Because of this, smaller values for α are used and therefore cannot be directly compared to the α-values in Subsection 5.4. Parameter s is the same for both of the algorithms.

5.5.1 Classification Performance

Macro-averaged F1, precision and recall are used for all the classification performance comparisons. This subsection shows only LIBLINEAR for WEBF2, SVM^{light} behaves similarly.

5.5.1.1 *Best Parameter Settings*

	α	s
100k	$1x10^{-7}$	1000
b100k	$1x10^{-7}$	100
200k	$1x10^{-7}$	15 000

Table 36: Best parameter settings for WEBF2.

Introduces Table 36, references Table 30.

Motivation The parameter settings for the best classification performance with WEBF2 for all three corpora are shown.

Legend The second column shows parameter α. While s is the same for WEBF1 and WEBF2, WEBF2 has smaller α values (due to the normalization process) and should not be compared to the values in Table 30.

Technical Analysis Contrary to WEBF1, WEBF2 has the same value for α for all three corpora. This should not come as a surprise since α is more quantized with WEBF2 than it is with WEBF1. While even smaller values have been tested and therefore more features were selected, $1x10^{-7}$ is still the best. Same as with WEBF1, values for s seem arbitrary.

5.5.1.2 *Best Run*

		Base	WEBF2	Diff
		%	%	PP
100k	F1	82.5	84.3	+1.8
	P	94.3	92.9	−1.4
	R	73.3	77.2	+3.9
b100k	F1	76.2	76.4	+0.2
	P	84.9	81.8	−2.9
	R	69.2	71.8	+2.4
200k	F1	84.0	85.7	+1.7
	P	92.8	90.9	−1.9
	R	76.8	81.0	+4.2

Table 37: Comparison between WEBF2 and baseline.

	Features		F1		
	#	%	WEBF1	WEBF2	Diff
	k	%	%	%	PP
100k	11.2	1.1	84.3	84.3	0.0
b100k	33.8	3.8	76.3	76.4	+0.1
200k	57.2	5.9	85.7	85.7	0.0

Table 38: F1 comparison between WEBF1 and WEBF2.

Tables 37 and 38. *Introduces*

Shows the best experimental run for WEBF2 and the difference to the baseline and WEBF1. *Motivation*

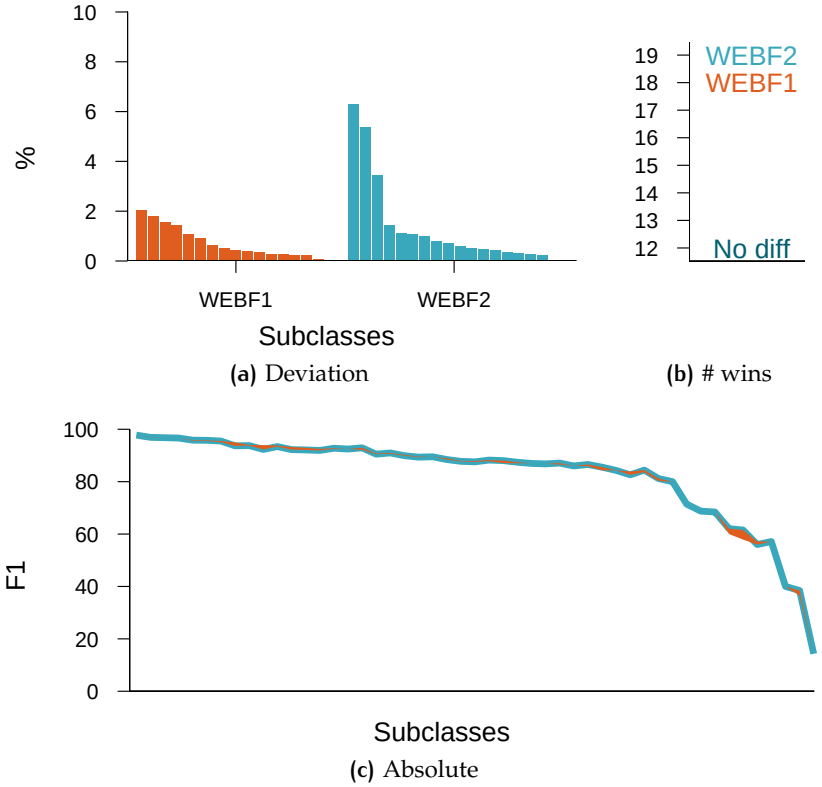
Table 37: Precision (P), recall (R) and F1 are shown in the rows for the three corpora. *Legend*

Table 38: F1 comparison between WEBF1 and WEBF2. Shows the number of features kept by WEBF2.

For **both** tables the difference is shown in PP.

The classification performance of WEBF2 is very similar to WEBF1. It is small enough to put them both on the same level by finding the best parameter settings for α and s or fine-tuning the classifier parameters more. The difference to the baseline is therefore as small as it is for WEBF1. The normalization process of WEBF2 does not seem to change the macro-averaged classification measures and it is unclear when to use WEBF2 over WEBF1 and vice versa. Whether or not WEBF2 would help improve certain subclasses is investigated in the next subsection. *Technical Analysis*

5.5.1.3 *Comparison WEBF1 and WEBF2*



(a) Deviation **(b)** # wins

(c) Absolute

Figure 72: F1.

Introduces Figures 72, 73 and 74.

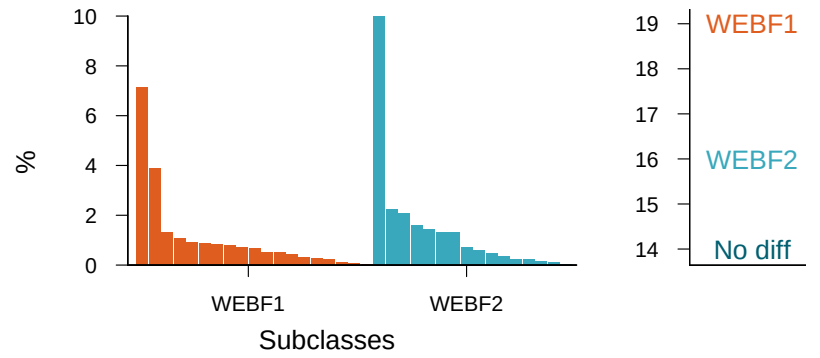
Motivation It compares F1, precision and recall on subclass level between WEBF1 and WEBF2 on the 100k corpus.

Legend **Figure 72**: **a)** shows the increase of a method over the other relative to the worse performing method in percent. The x-axis shows both methods and each bar represents a single subclass, in which the method performs better than the other. **b)** shows the number of wins per method and the number of equal subclasses. **c)** shows all subclasses in descending order. Blue lines shows WEBF2, the orange areas show the difference to WEBF1.

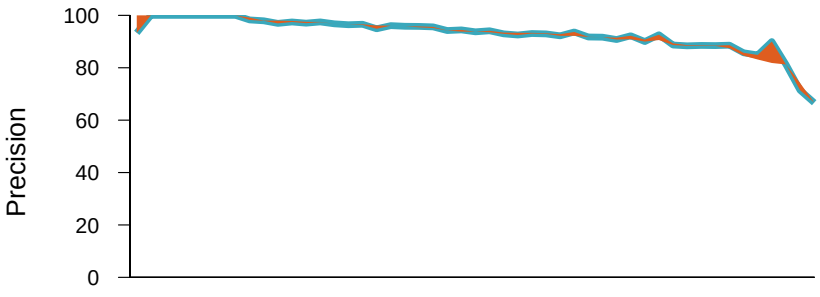
Figures 73 and 74 show precision and recall respectively.

Technical Analysis WEBF1 and WEBF2 are very close, even on subclass level. WEBF2 improves the classification performance of WEBF1 in approximately 1/3 of the subclasses. In another third WEBF1 takes the crown while in the remaining third, both perform equally. The biggest change is 10%. The better a subclass performs, the closer the methods are. At least for F1, the change in classification performance is too little to warrant preferring WEBF2 over WEBF1.

Generally the same characteristics can be observed with recall and precision. Except under precision, WEBF2 is slightly better than WEBF1 in the worse performing subclasses.

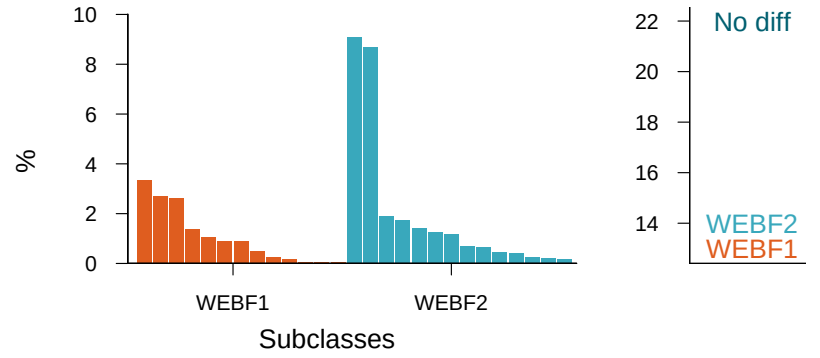


(a) Deviation (b) # wins

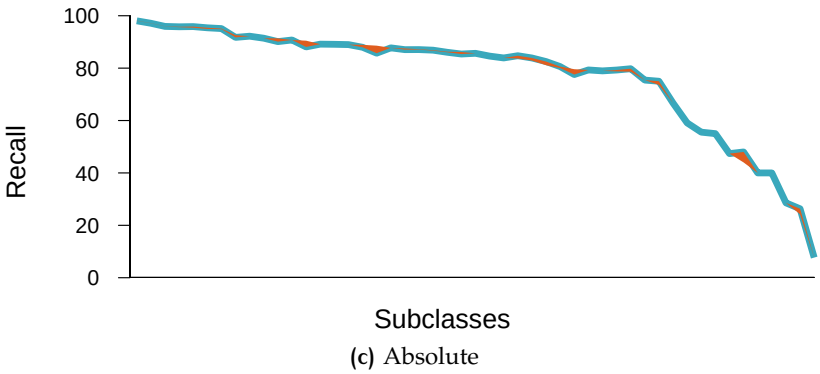


(c) Absolute

Figure 73: Precision.



(a) Deviation (b) # wins



(c) Absolute

Figure 74: Recall.

5.5.2 Parameter Analysis

Since parameter s is exactly the same for both WEBF1 and WEBF2, an analysis can be neglected. Parameter α changes because of the normalization but it is still solely responsible for the number of features that are removed and kept and therefore for how well WEBF2 performs. Same as with WEBF1, shown in Subsection 5.4.2, parameter s is used for fine-tuning and changing specific subclasses. Because of this the analysis of the impact of parameter α for WEBF2 is neglected as well. The most important aspects of the two parameters have been mentioned in the paragraphs explaining the best parameter combinations.

5.5.3 Feature Analysis

Since WEBF1 and WEBF2 do not select the same number of features, a comparison of the absolute values of the number of selected features is not as revealing and useful, the only feature analysis presented is for the relative number of unique features.

5.5.3.1 *Unique Feature Comparison to WEBF1*

Introduces Figure 75.

Motivation I analyze the difference in the number of unique selected features for every subclass.

Legend The x-axis shows the percentage of unique features in a subclass in ascending order by the top method. The y-axis shows every subclass. A bar reflects a single subclass. The color of a bar shows which of the two methods has selected more unique features per subclass. The top of the bar shows the percentage of unique features the method with a higher percentage of unique features has selected. The bottom of the bar shows the percentage of the method selecting a lower percentage of unique features. The length of the bar visualizes the drop between the two methods. The 100k corpus is used. The figure shows the difference between the experiments with the best classification performance for the respective method.

Technical Analysis It is quite visible that in most cases WEBF2 has a lower percentage of unique features per subclass. In only twelve out of 49 subclasses WEBF2 keeps more than WEBF1. In eleven cases, the difference is less than 15%, in one it is even lower than 1%. Only in subclass B01D, WEBF2 keeps about 30% more than WEBF1. On the other hand, there are subclasses in which WEBF1 selects 45% more unique features. WEBF1 clearly selects a higher percentage of unique features but, as previously shown, this does not translate into a significantly better classification performance. With the exception of one subclass, WEBF2 wins only in subclasses where both methods select fewer features.

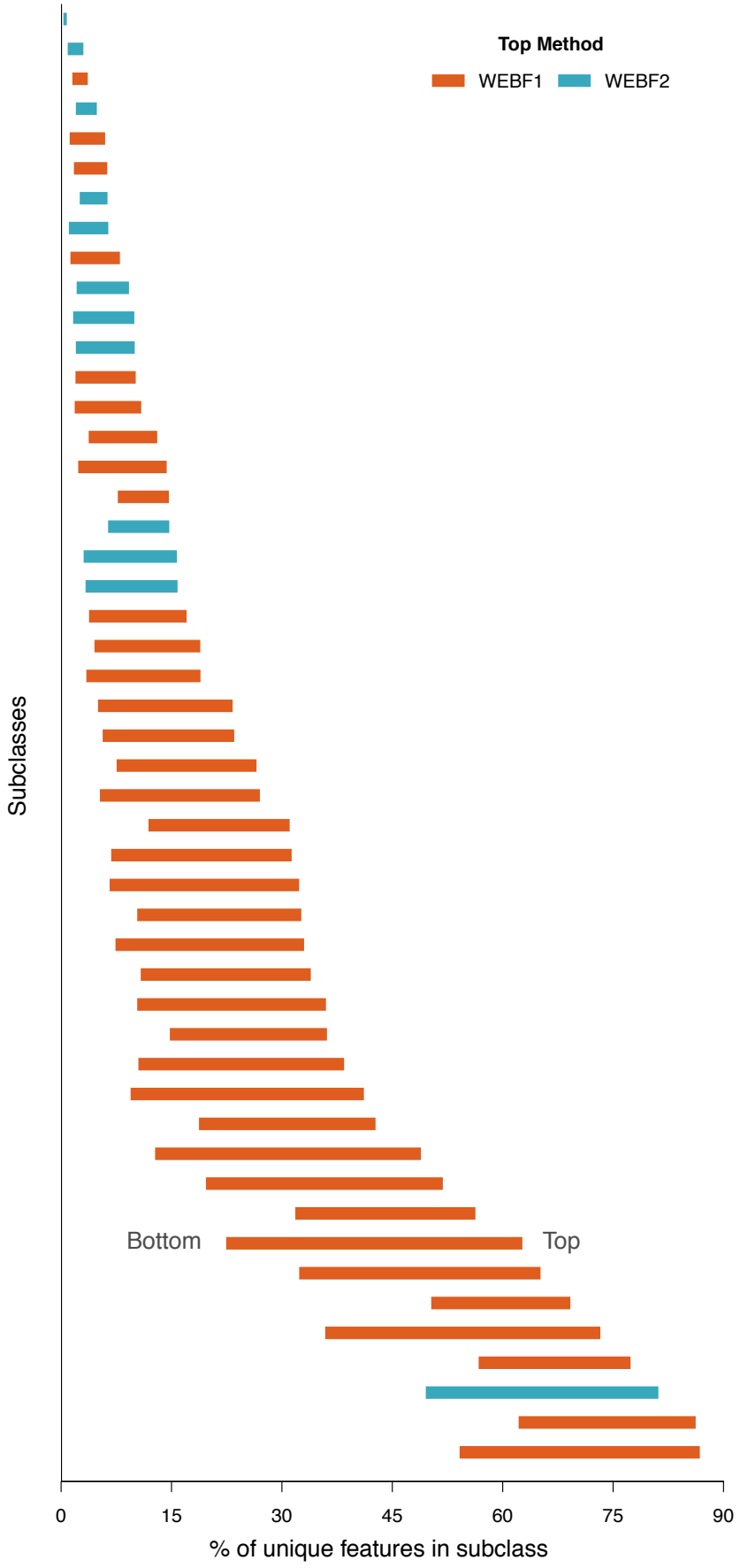


Figure 75: Comparison between WEBF1 and WEBF2 on 100k. In ascending order by top method.

5.5.4 Runtime Analysis

5.5.4.1 *Classifier CPU Runtime*

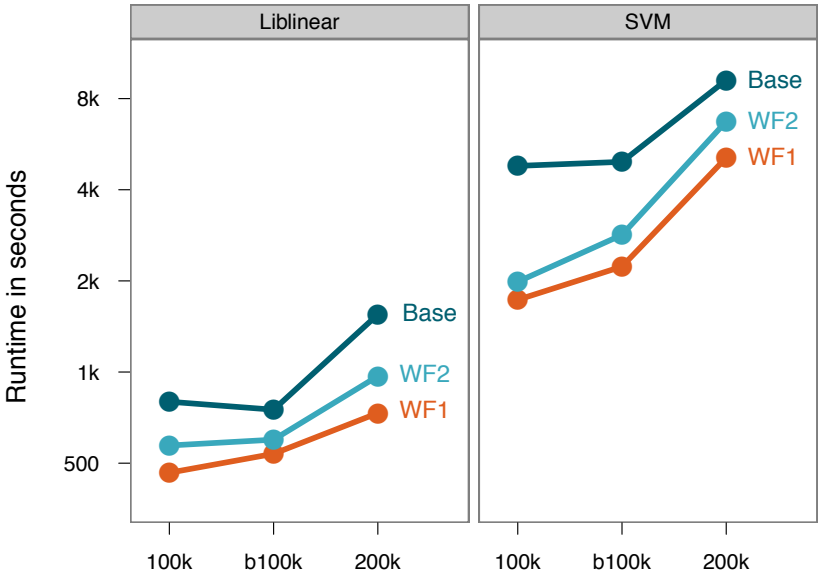


Figure 76: CPU runtime of the classifiers.

Introduces Figure 76.

Motivation While both WEBF1 and WEBF2 select almost the same number of features, the classifiers might solve the classification task in a different time. This section analyzes the difference in the CPU runtime of the training phase of the classifiers.

Legend The y-axis shows the runtime in seconds using $\log_{10}$. The x-axis shows the three corpora. Both classifiers used 10 cores. The baseline (Base), WEBF1 (WF1) and WEBF2 (WF2) are presented in the figure. The best performing experiments for all three methods are shown.

Technical Analysis Surprisingly there is quite a large gap between WEBF1 and WEBF2. While WEBF2 is still faster than the baseline, it is slower than WEBF1 in all of the six cases. The difference between WEBF1 and WEBF2 ranges from 20% to 30%. The difference between WEBF2 and the baseline is up to 60%.

5.5.4.2 *Algorithm CPU Runtime*

WEBF1	503
WEBF2	811
Diff	208

Table 39: Algorithm CPU runtime comparison between WEBF1 and WEBF2 in seconds for best performing classification experiment.

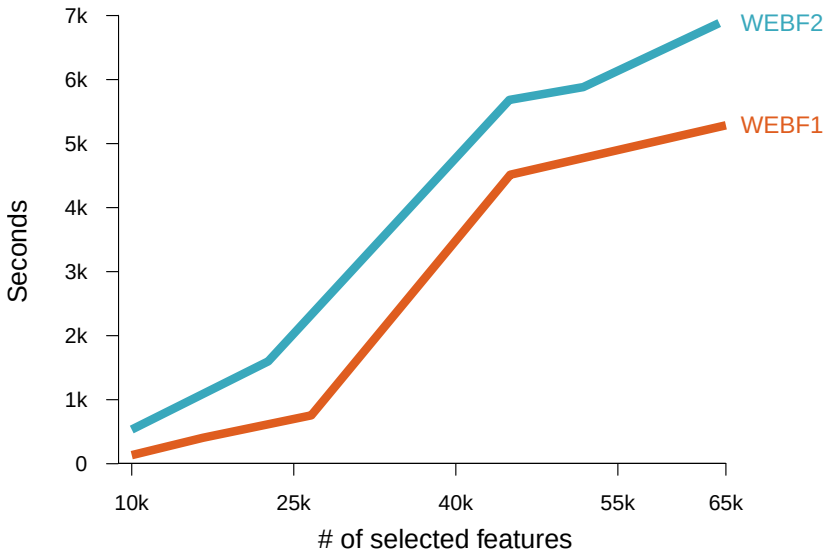


Figure 77: CPU runtime for algorithms with same number of features.

Table 39 and Figure 77, references Subsection 5.4.4.4. *Introduces*

The difference in the CPU runtimes of the feature selection algorithms is important when deciding which one to chose. As promised in Subsection 5.4.4.4, this section delivers a comparison between WEBF1 and WEBF2. *Motivation*

Table: shows the difference between WEBF1 and WEBF2 in seconds for the best performing classification experiment for both. *Legend*

Figure: the x-axis shows the number of selected features, the y-axis shows the number of seconds the algorithms needed. Five different numbers of selected features are shown.

Both: 10 cores on the 100k corpus with LIBLINEAR is used.

The analysis should be looked at with caution because the runtime depends on the implementation of the algorithms. Both algorithms are similar and share a large portion of code. Nonetheless a faster implementation for both might be possible. *Disclaimer*

At all times WEBF2 is slower than WEBF1. The best performing experiment of WEBF2 is 25% slower than WEBF1, in the other five experiments WEBF2 is 10% to 28% slower. This sounds reasonable due to the additional work WEBF2 has to go through. *Technical Analysis*

5.6 WORD-ENTROPY BASED STOP WORD LIST

This section shows the experiments performed for WEBSOL. Furthermore it will be compared to MI, IG and TF. WEBSOL has two independent parameters, s and β. Both will be analyzed in Subsection 5.6.3.

The best performing experiment of WEBSOL does not remove the most features. There are experiments removing more features but still beating the baseline. Therefore a second experiment, named WEBSOL-L, is introduced and presented on various occasion. It increases the performance of the baseline by a very small margin but removes around 40% more features than WEBSOL. It will only be shown for LIBLINEAR .

5.6.1 Comparison to WEBF1

In this section WEBSOL and WEBSOL-L are presented. While the focus is on those two, a comparison to WEBF1 is drawn whenever possible and useful. Table 40 shows all the subsections sporting a comparison between WEBSOL and WEBF1.

	Section Name
5.6.2.1	Best Experiment Selection
5.6.2.3	Micro-average Results
5.6.2.4	Features Removed
5.6.2.5	Method Comparison
5.6.2.8	Precision and Recall on Subclass
5.6.4.3	Sparseness
5.6.5.1	Classifier CPU Runtime
5.6.5.2	Algorithm CPU Runtime

Table 40: List of comparisons between WEBSOL and WEBF1.

5.6.2 Classification Analysis

5.6.2.1 *Best Experiment Selection*

		WEBSOL	WEBF1	Diff	Base	Diff
		%	%	PP	%	PP
		A	B	C = A - B	D	E = A - D
LIB	100k	84.1	84.3	-0.2	82.5	+1.6
	200k	85.6	85.7	-0.1	84.0	+1.6
	b100k	76.6	76.3	+0.3	76.2	+0.4
SVM	100k	67.0	67.8	-0.8	66.8	+0.2
	200k	79.6	79.9	-0.3	79.1	+0.5
	b100k	66.6	66.7	-0.1	66.2	+0.4

Table 41: F1.

		WEBSOL	WEBF1	Diff	Base	Diff
		%	%	PP	%	PP
		A	B	C = A - B	D	E = A - D
LIB	100k	76.6	77.2	−0.6	73.3	+3.3
	200k	80.7	81.0	−0.3	76.3	+4.4
	b100k	71.5	71.8	−0.3	69.1	+2.4
SVM	100k	60.0	61.2	−1.2	59.7	+0.3
	200k	71.6	71.9	−0.3	70.7	+0.9
	b100k	58.4	58.4	0.0	57.3	+1.1

Table 42: Recall.

		WEBSOL	WEBF1	Diff	Base	Diff
		%	%	PP	%	PP
		A	B	C = A - B	D	E = A - D
LIB	100k	93.1	92.9	+0.2	94.2	-1.1
	200k	91.1	90.9	+0.2	92.7	-1.6
	b100k	82.4	81.4	+1.0	84.8	-2.4
SVM	100k	75.8	76.1	−0.3	75.9	-0.1
	200k	89.6	89.9	−0.3	89.8	-0.2
	b100k	77.4	77.8	−0.4	78.3	-0.9

Table 43: Precision.

100k			200k			b100k		
F1	R	P	F1	R	P	F1	R	P
%	%	%	%	%	%	%	%	%
82.6	74.8	92.1	83.8	77.1	91.8	76.4	69.8	83.6

Table 44: Statistics for WEBSOL-L with LIBLINEAR.

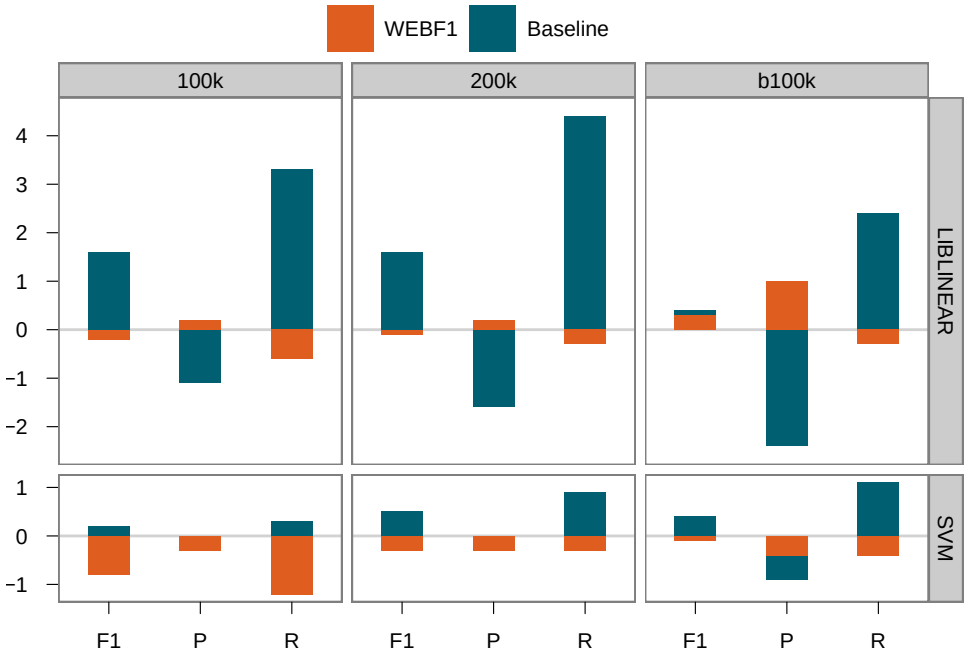


Figure 78: Comparison WEBSOL to WEBF1.

Introduces Tables 41, 42, 43 and 44, Figure 78.

Motivation The best classification performances for WEBSOL as well as a comparison to WEBF1 and the baseline is shown.

Legend **Table 41** shows F1 for WEBSOL and compares it to WEBF1 and the baseline in the columns B and D. Rows show both classifiers on all three corpora. Columns C and E show the difference between WEBSOL and WEBF1 or the baseline respectively.

Table 42: proceeds accordingly for recall.

Table 43: proceeds accordingly for precision.

Table 44 shows results for LIBLINEAR with WEBSOL-L.

Figure 78 illustrates the data from the previous tables. The y-axis shows the difference between WEBSOL and WEBF1/baseline. A positive value means WEBSOL performs better than WEBF1/baseline. The x-axis shows F1, precision (P) and recall (R), grouped by corpora.

Technical Analysis WEBSOL improves the baseline but only on recall. Precision drops, same pattern as we have seen for WEBF1, and F1 increases by a maximum of 1.6%. WEBSOL is worse then WEBF1 in everything but precision although the improvement is only between 0.2% and 1%.

The difference between LIBLINEAR and SVM^{light} on the classification performance can be attributed to different, imperfect parameter settings, since it is only around 1%. It is very much possible to improve both by doing a more rigorous grid search on the parameters.

WEBSOL-L performs better than the baseline by a small margin.

5.6.2.2 *Best Experiment Settings*

			Kept Features		Params	
			#	%	s	β
			k		k	k
WEBSOL	**LIB**	100k	630	60.8	15	450
		b100k	490	53.7	1	450
		200k	540	54.6	10	450
	SVM	100k	580	53.7	15	500
		b100k	390	41.7	10	550
		200k	390	39.5	10	600
WEBSOL-L	**LIB**	100k	430	39.8	20	650
		b100k	340	36.3	10	600
		200k	340	33.4	20	650

Table 45: Best settings for WEBSOL.

Table 45. *Introduces*

The table shows the best parameter combination for WEBSOL for LIBLINEAR and SVM^{light}. LIBLINEAR for WEBSOL-L is shown as well. *Motivation*

The first and second columns show the number of kept features and the percentage. Columns three and four show the settings for the two parameters s and β. *Legend*

The reduction of the number of features is significantly smaller than for WEBF1. Between 40% and 45% can be removed for the best classification performing experiment. However, without sacrificing the classification performance of the baseline, WEBSOL-L is capable of removing approximately 63% of the features. *Technical Analysis*

WEBSOL with SVM^{light} removes more features. On the 100k and b100k corpora, the reduction ratio is approximately 10% higher, whereas on the 200k corpus, it is about 15% higher. Classification performance drops by only half a percent.

The setting for s seems arbitrary. For example, $s = 15000$ splits on the concatenation of three patent documents. $s = 100$ splits on less than a single abstract.

WEBSOL-L consistently selects less than 60% of the features and always has a different setting for s than WEBSOL.

5.6.2.3 *Micro-average Results*

		WEBSOL	Base	Diff	WEBF1
		%	%	PP	PP
		A	B	A - B	A - WF1
F1	100k	94.5	92.0	+2.5	+4.7
	200k	90.7	90.0	+0.7	+0.8
Precision	100k	95.9	96.3	−0.4	+0.7
	200k	92.2	93.8	−1.6	0.0
Recall	100k	93.1	89.1	+4.0	+8.2
	200k	89.2	86.5	+2.7	+1.4

Table 46: Micro-averages comparison between WEBSOL and baseline.

Introduces Table 46.

Motivation While the concentration of this thesis lies on macro-averages measures, this table shows micro-averaged classification performances to give the reader a rudimentary understanding of how WEBSOL performs on micro-averaged measures. This is also the only mention of micro-averaged results in the entire WEBSOL section.

Legend The difference shows PP between WEBSOL and the baseline. Column WEBF1 shows the difference between WEBSOL and WEBF1 in PP. A positive value indicates that WEBSOL is better than WEBF1/the baseline. LIBLINEAR is used but SVM^{light} behaves similarly. Only 100k and 200k are shown. b100k is neglected since for this corpus micro- and macro-averages are the same. Why? b100k has exactly 2000 documents assigned to each of the 50 subclasses. Hence each subclass weighs in with 2%. This holds for both micro- and macro-averages.

Technical Analysis Contrary to micro-averages for WEBF1, WEBSOL shows improvements in both corpora. The gain for all three measures is less than for macro-averages, both in absolute and relative terms. Precision decreases on both corpora, recall increases. WEBSOL significantly outperforms WEBF1 on all three measures, on recall the difference is 8.2 percentage points.

As it was with WEBF1, the absolute value of the classification performance is higher than macro-averages. On recall, the difference is up to 15 percentage points. This leads to an excellent classification result.

5.6.2.4 *Features Removed*

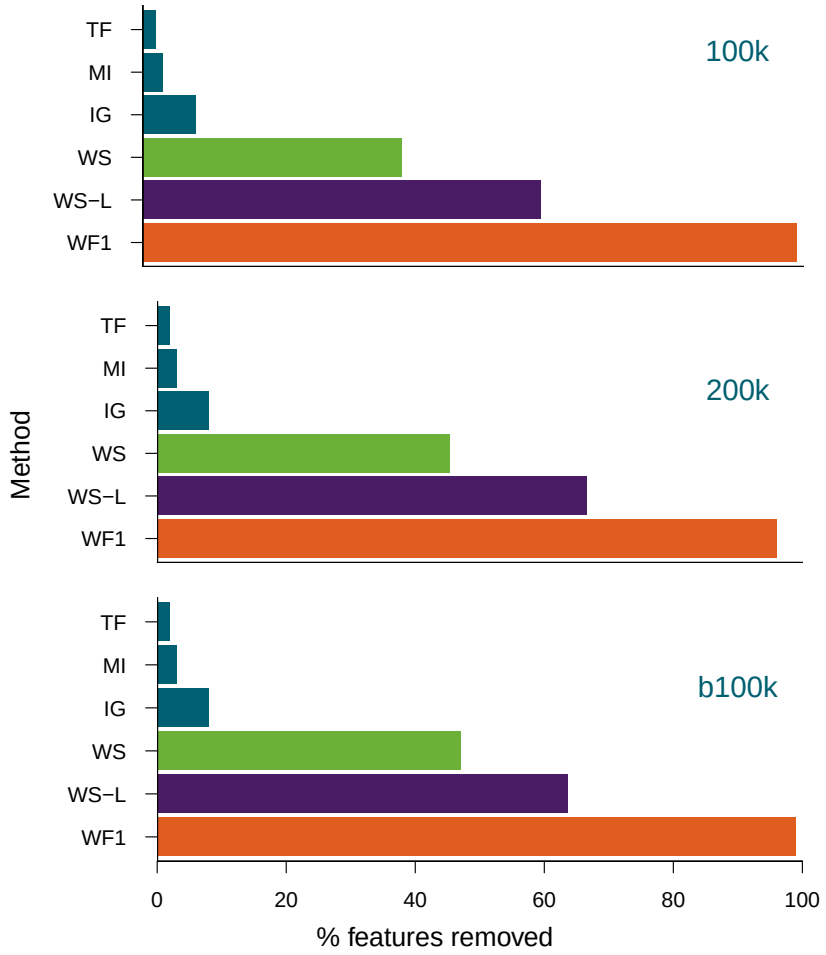


Figure 79: Comparison of the number of removed features.

Figure 79. *Introduces*

Compares the number of features removed by WEBSOL to other feature selection algorithms and to WEBF1. *Motivation*

The y-axis shows the various feature selection methods, TF-IDF (TF), MI, IG, WEBF1 (WF1), WEBSOL (WS) and WEBSOL-L (WS-L), grouped by the three corpora. The x-axis shows the percentage of features removed by the method compared to the baseline. *Legend*

WEBSOL removes more features than the other three feature selection methods by almost 35 percentage points. It removes approximately 21 percentage points less than WEBSOL-L. It is also clearly outperformed by WEBF1, which removes 99%, an increase of 130% compared to WEBSOL. *Technical Analysis*

Not surprisingly WEBSOL-L outperforms WEBSOL on all three corpora. Yet, it is not possible to reduce the features any further without sacrificing classification performance. There is no way to reduce the features as much as WEBF1.

5.6.2.5 *Method Comparison*

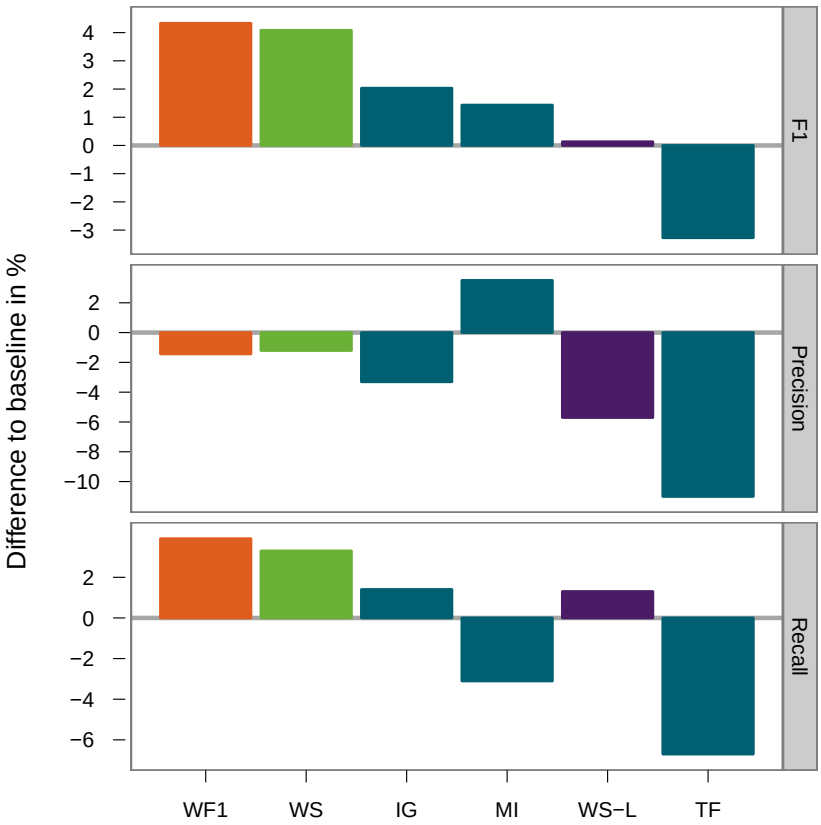


Figure 80: Comparison of WEBSOL to other methods.

Introduces Figure 80, references Table 41.

Motivation Compares WEBSOL to three other feature selection algorithms, the baseline, WEBF1 and WEBSOL-L.

Legend The y-axis shows the difference to the baseline in percent grouped by the three measures. If the value on the y-axis is positive, the method performs better than the baseline. The 100k corpus is used. Five methods are compared, WEBF1 (WF1), MI, IG, TF-IDF (TF), WEBSOL (WS) and WEBSOL-L (WS-L). The best performing experiment was chosen for each method.

Technical Analysis WEBSOL performs better on F1 than the baseline. It also performs better than the other three feature selection algorithms. As previously seen in Table 41, it is 0.2% worse than WEBF1. Since it mirrors the behavior of WEBF1 in both recall and precision, the analysis from WEBF1, found in Subsection 5.4.1.2, is also valid for WEBSOL. It outperforms every method on recall, except for WEBF1. This is the main advantage of WEBSOL and therefore should be used when an increase in recall is needed.

WEBSOL-L is close to WEBSOL on recall but drops by approximately 6% on precision.

5.6.2.6 *Performance with same Number of Features*

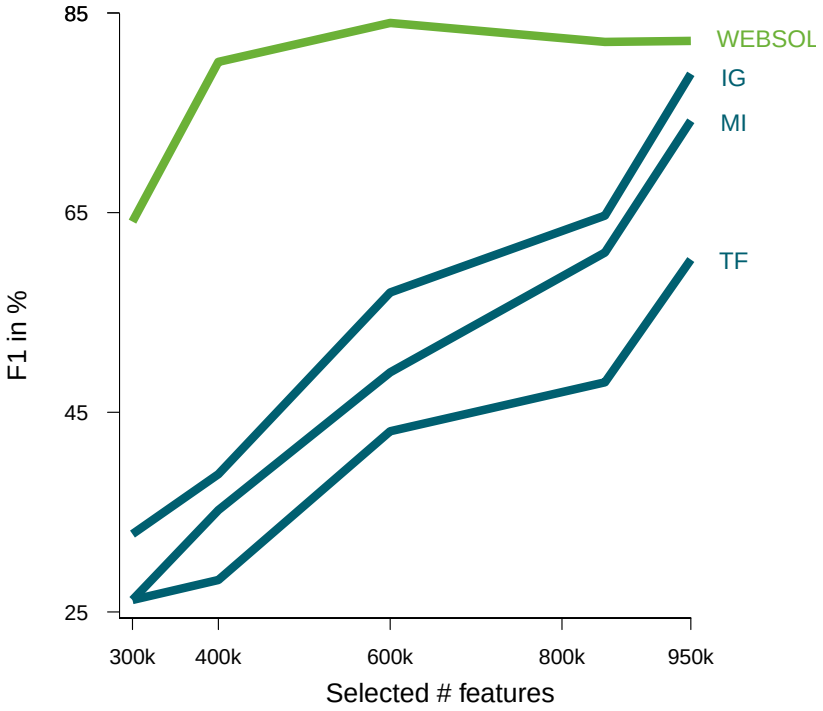


Figure 81: F1 comparison with the same number of features.

Figure 81, references Table 45. *Introduces*

The figure shows the performance of WEBSOL compared to three other feature selection methods when all four of them use the exact same number of features. *Motivation*

The 100k corpus is used. Since WEBSOL can be set to use an exact number of features, it does not have the problem WEBF1 has. On the x-axis the number of selected features is shown. Five different numbers of selected features are explored, hence the different gaps between the axis ticks. The y-axis shows the macro-averaged F1 in percent. *Legend*

WEBSOL performs at approximately 63% with only 300 000 features but increases to near-optimum with 400 000 features, which is already better than the baseline. WEBSOL peaks at around 600 000 features, correlating with the overall best performing experiment for WEBSOL seen in Table 45. With 950 000 features (or approximately 90% of the features), MI and IG are getting closer to WEBSOL but are unable to outperform it. *Technical Analysis*

The figure does not show WEBF1 because WEBF1 selects fewer than 100 000 features. Going lower than 300 000 features with WEBSOL is possible but the classification performance will converge to zero.

5.6.2.7 *Classification Performance on Subclasses*

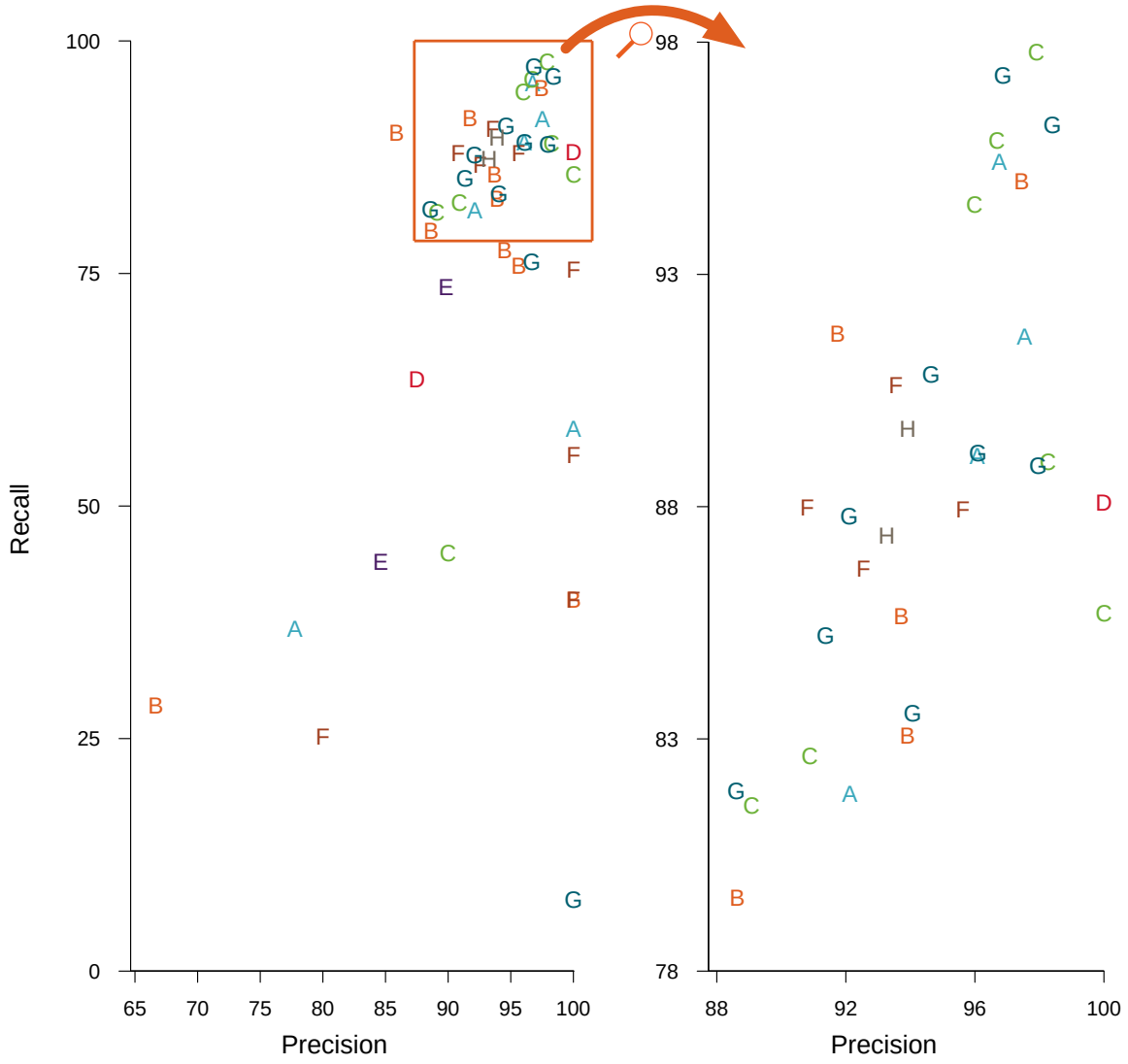


Figure 82: Precision and recall for all subclasses.

Introduces Figure 82.

Motivation The figure shows the outliers and clusters on recall and precision for every subclass.

Legend The y-axis shows recall in percentage, the x-axis shows precision in percentage. The right figure shows the top right corner of the left figure.

Technical Analysis Seen for WEBF1, there is a cluster of subclasses for WEBSOL as well. 39 out of 49 subclasses are in the top right corner, performing at 88% or more on precision and 70% or more on recall.

What has to be underlined is the fact that every section except for D has one outlier and recall can get as bad as 8% for a single subclass. The worst performing subclass on precision still achieves 64%.

5.6.2.8 *Precision and Recall on Subclass*

Introduces

Figures 83 and 84, references Tables 42 and 43.

Motivation

First, the difference on precision and recall between WEBSOL and the baseline is analyzed. Then the difference between WEBF1 and WEBSOL is investigated.

Legend

Figure 83: shows the difference between WEBSOL and the baseline. The x-axis shows the subclasses, each dot being one subclass. The y-axis shows the difference to the baseline in percent. It is grouped by the three corpora. The size of the dots has no meaning other than increased visibility for lonely dots at the top.

Figure 84: follows the legend of Figure 83, except it shows the comparison between WEBSOL and WEBF1.

Both: LIBLINEAR is used. If the difference between the two methods is zero, no dot is shown. Colors and shapes show which of the two methods is better.

Technical Analysis

Unsurprisingly WEBSOL performs worse on precision than the baseline. Similar to WEBF1 it performs better than the baseline for recall, as all but three subclasses on the 100k corpus outperform the baseline. Compared to WEBF1, there are fewer spikes with the exception of one subclass. Said subclass increases by approximately 50%.

Looking at WEBF1, WEBSOL confirms the previously mentioned tables. On recall on the 100k corpus the difference between the two methods is relatively small, except for three subclasses where WEBF1 outperforms WEBSOL by approximately 10%. This also explains the difference in recall between the two, as shown in Table 42. In the other two corpora the difference is smaller and the peaks are less pronounced. Both methods perform better in about the same number of subclasses.

As far as precision is concerned, WEBSOL performs better on all three corpora, especially on b100k.

F1 follows accordingly.

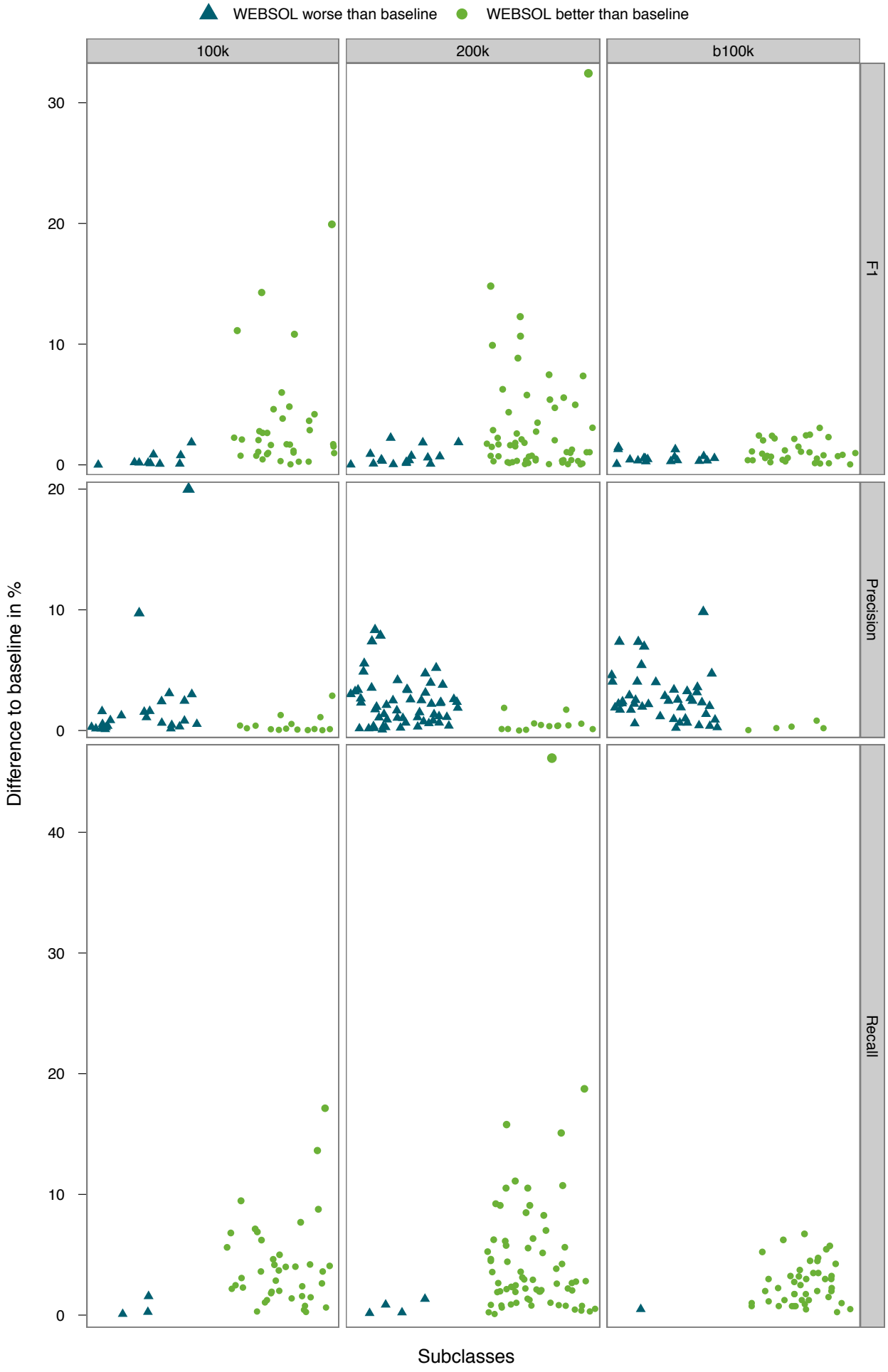


Figure 83: Classification comparison between WEBSOL to the baseline.

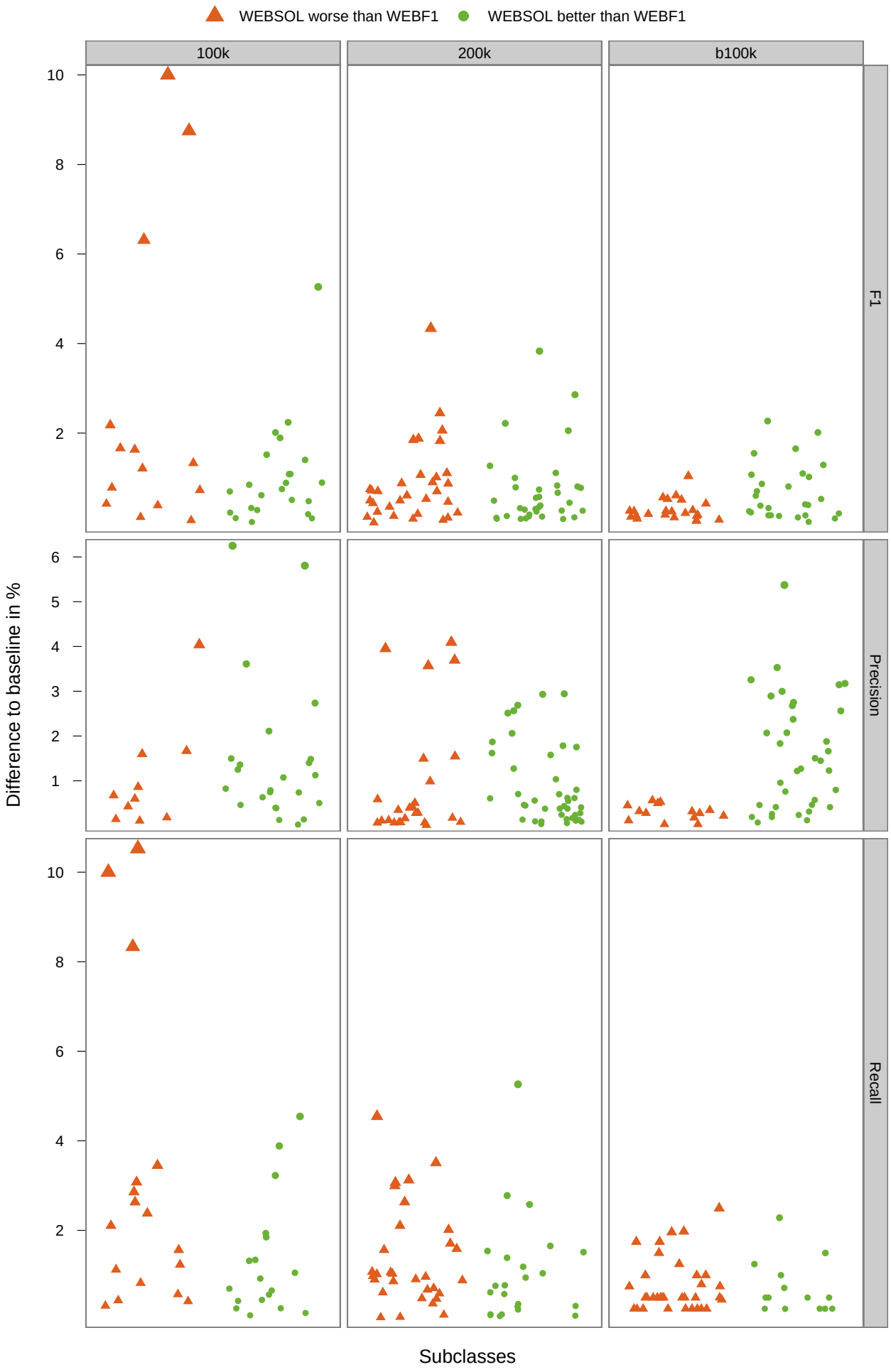


Figure 84: Classification comparison between WEBSOL to the WEBF1.

5.6.2.9 *Precision and Recall grouped by Section*

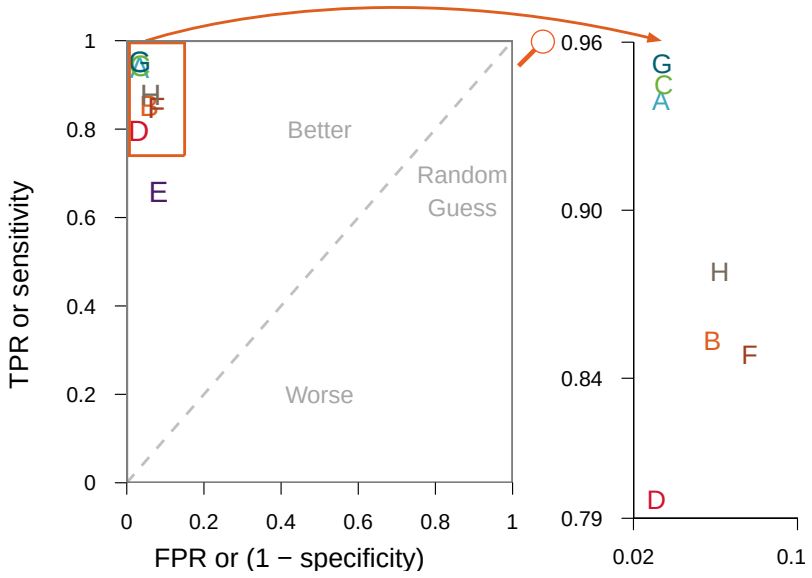


Figure 85: Sections in ROC space.

Introduces Figures 85, 86 and 87.

Motivation These figures show the classification performance of all the subclasses but grouped by their section. This shows whether or not a section is responsible for a large change of a specific measure.

Legend **Figure 85**: shows the ROC space on section level. ROC spaces are further explained in Fawcett [13]. The experiment with the highest F1 value on the 100k corpus was used. The right chart zooms into the upper left corner of the left chart.

Figure 86: the y-axis shows recall in percent, the x-axis displays precision in percent. Each item represents the value of a section by macro-averaging all the subclasses of that section.

Figure 87: Its y-axis shows the F1 values for a subclass and its x-axis the sections.

Technical Analysis The ROC space shows that the classification is better than a random guess and that sections A, C and G achieve a close to perfect classification. When looking at precision and recall, section E is the only negative outlier. Sections C and H are positive outliers. Because section H does not have many subclasses assigned to it, its impact on F1 is low. Looking at F1, the median in the sections are distributed over approximately 25%.

Th sections responsible for most of the macro-averaged classification performance are C and G. Since they are at approximately 95% on precision and are two of the three best performing sections on recall, it seems infeasible to push the classification performance further by improving those two. Sections B, E and F have to be increased in order of achieving an overall improvement.

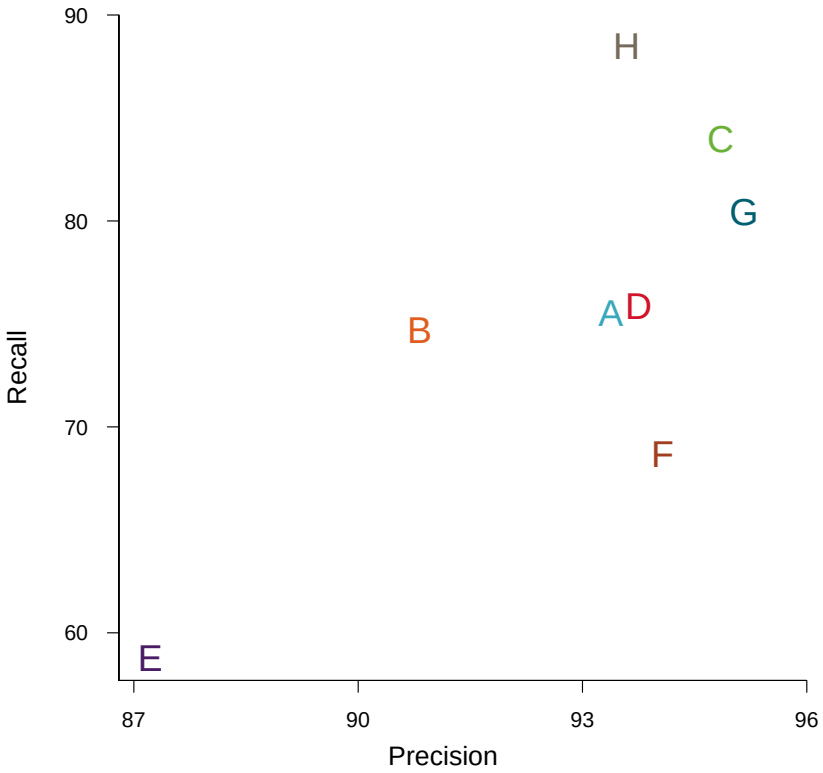


Figure 86: Precision and recall for macro-averaged sections.

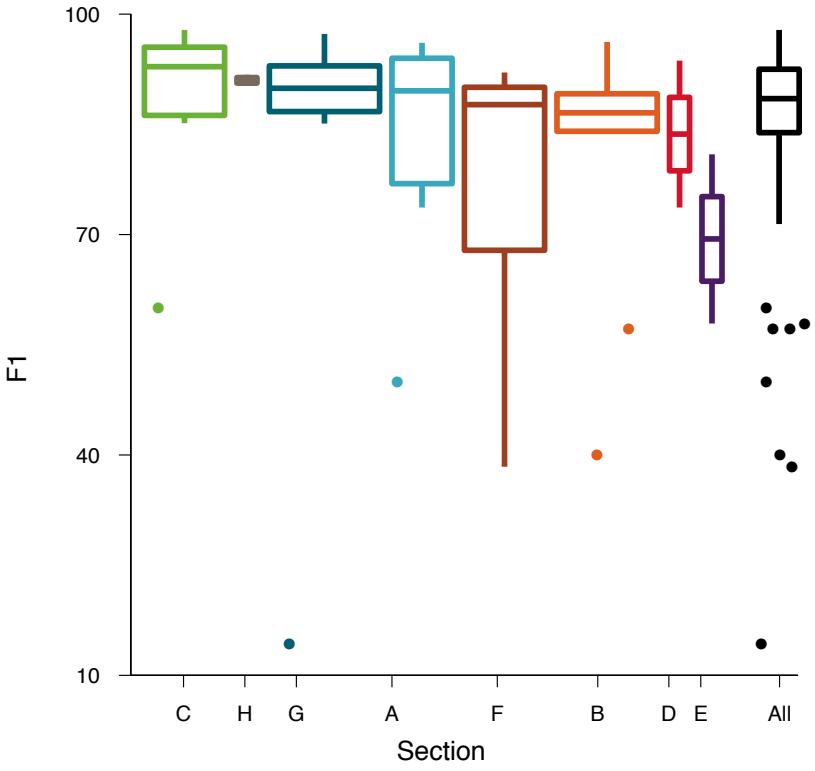


Figure 87: F1 for subclasses grouped by sections.

5.6.2.10 *Performance Analysis by Size*

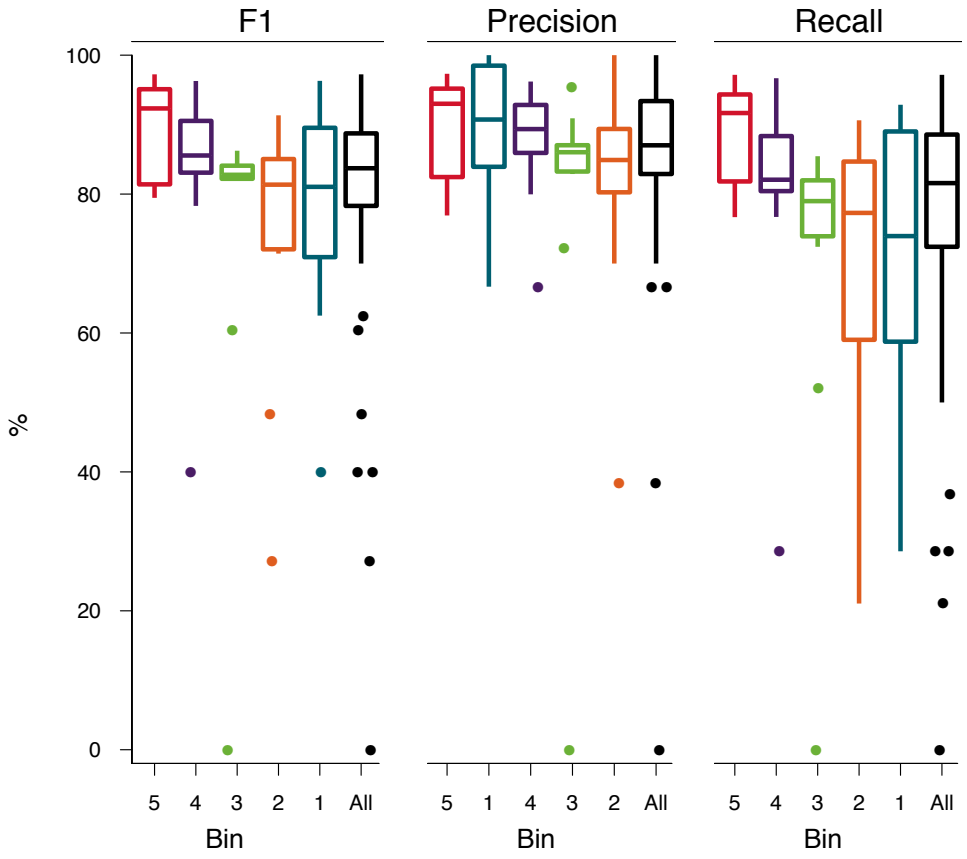


Figure 88: WEBSOL-L on 100k inn descending order of median.

Introduces

Figures 88, 89 and 90, references Subsection 5.4.1.10.

Motivation

As in Subsection 5.4.1.10, it is important to know what impact the size of a subclass, measured by the number of documents assigned to it, has on the various performance measures. This is analyzed in three figures.

Legend

The legend is the same as for WEBF1 in Subsection 5.4.1.10. Figure 88 shows WEBSOL-L on the 100k corpus, the other two show WEBSOL.

Technical Analysis

On the 100k corpus, the rule seems to be the more documents are assigned to a subclass, the higher the chance it performs better. This is especially true for recall, where the difference between the medians of bin 1 and 5 is approximately 25%.

On the 200k corpus, the same characteristic is valid. However, the bins have fewer outliers than the 100k corpus (8% vs. 12.5%), making a prediction based on the number of documents of a subclass slightly easier.

WEBSOL-L introduces roughly 2% more outliers but keeps the characteristics intact. It also brings the medians of the bins closer together compared to WEBSOL.

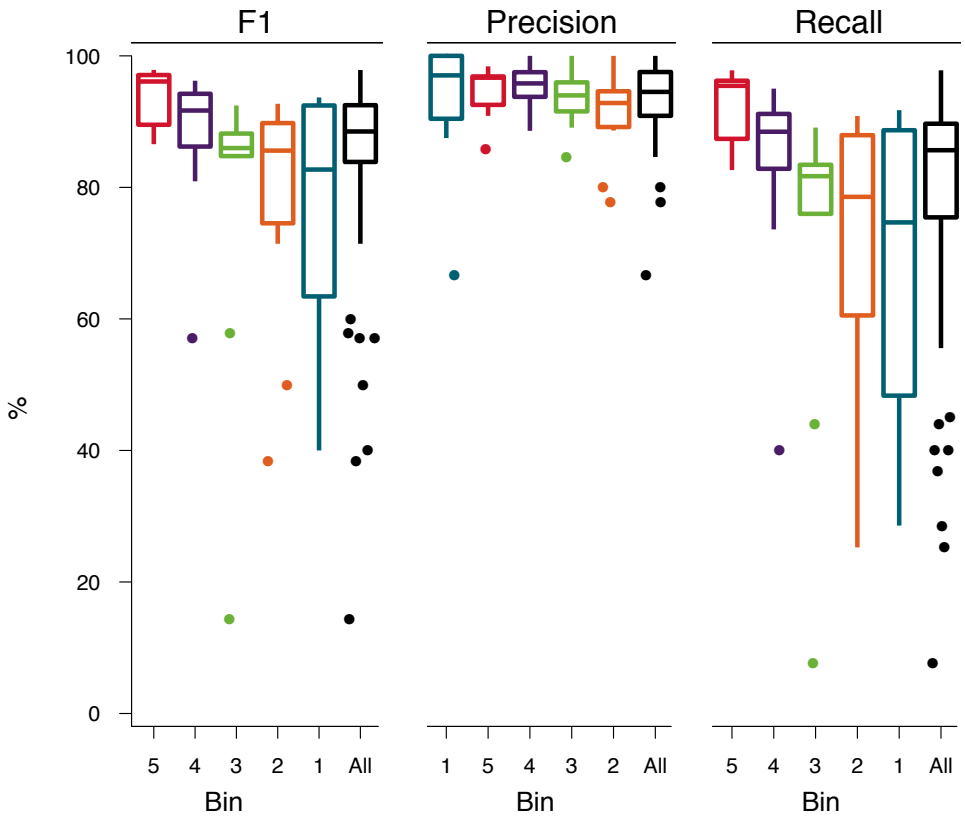


Figure 89: WEBSOL on 100k in descending order of median.

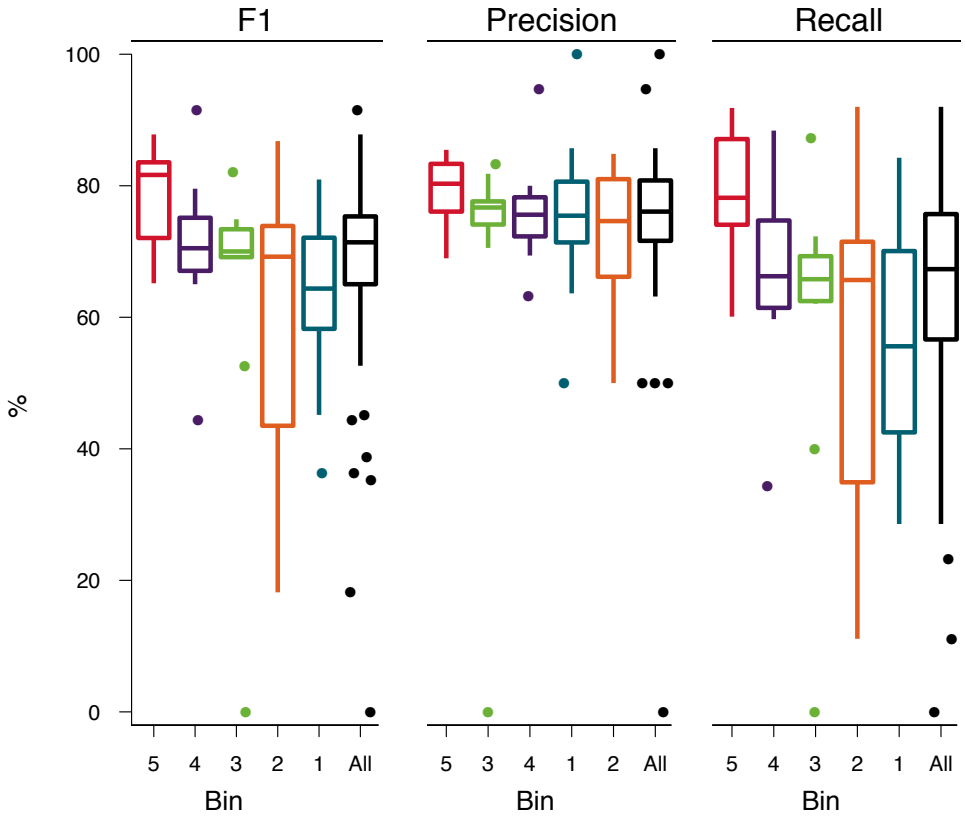


Figure 90: WEBSOL on 200k in descending order of median.

5.6.3 Parameter Analysis

5.6.3.1 *Parameter s*

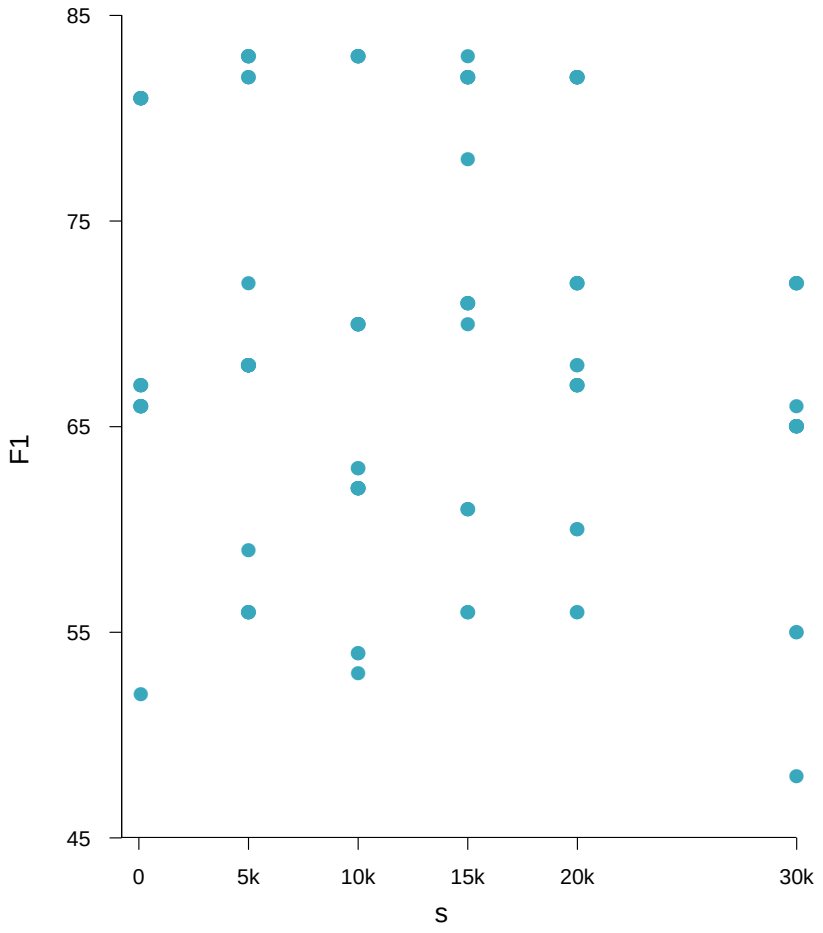


Figure 91: Parameter s.

Introduces Figure 91.

Motivation Shows the impact of parameter s on F1.

Legend The y-axis shows macro-averaged F1, the x-axis parameter s. No experiments were run for $20000 < s < 30000$. The 100k corpus and LIBLINEAR are used. All the experiments shown in the figure have the same parameters for the classifier and only change the settings of the two parameters of the algorithm, s and β.

Technical Analysis Every parameter setting for parameter s has at least one experiment that performs below average and most have at least one that performs in the top 5. This shows that s is not the sole responsible setting for the F1 output.

5.6.3.2 *Parameter* β

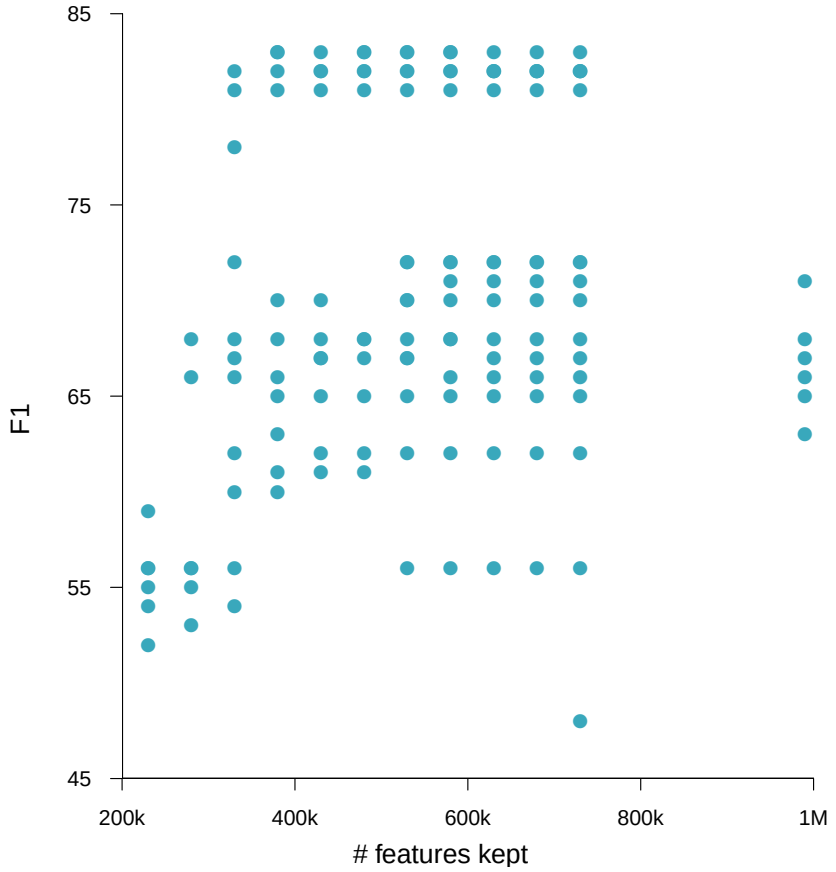


Figure 92: Parameter β.

Figure 92. *Introduces*

Shows the impact of parameter β on the macro-averaged F1. *Motivation*

The y-axis shows macro-averaged F1, the x-axis the number of features kept. This is calculated by subtracting parameter β from the total number of features of the corpus. The 100k corpus and LIBLINEAR are used. All the experiments shown in the figure have the same parameters for the classifier and only change the settings of the two parameters of the algorithm, s and β. *Legend*

Same as with parameter s, this parameter is not solely responsible for the outcome of F1. Each number of selected features has bad and well performing experiments. The only exception are experiments using less than 300 000 features. Below that, almost all of the experiments are bad. *Technical Analysis*

5.6.3.3 *Parameter Relation*

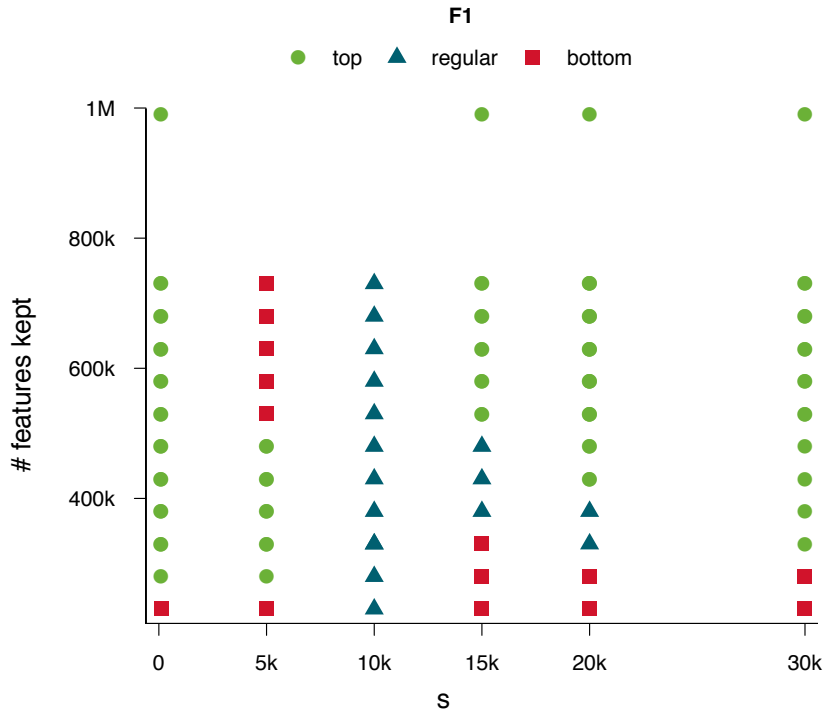


Figure 93: Comparison of β and s.

Introduces

Figure 93.

Motivation

Since the previous two subsections were unable to identify the reason for changes in F1, this section will analyze the impact of the combination of the two parameters.

Legend

The y-axis shows the number of selected features, the x-axis shows parameter s. As before, a dot represents a single experiment and its shape and color indicate how well it performs regarding F1. The 100k corpus and SVM^{light} are used. All the experiments shown in the figure have the same parameters for the classifier and only change the settings of the two parameters of the algorithm, s and β.

Technical Analysis

As indicated in the previous two sections, each parameter setting has negative and positive experiments. In the figure the positive ones are distributed over the entire area, while the bad ones are associated with a lower number of selected features. This means a high number of features can automatically avoid the worst performing experiments but for finding the best parameter setting, one has to try all the different parameter combinations. This is different to WEBF1 and WEBF2, where parameter α was solely responsible for the outcome of F1. This means more classifiers have to be trained and therefore WEBSOL requires more resources.

5.6.3.4 *Impact of s on Subclass-Level*

Figure 94. *Introduces*

As we have seen in the previous three subsections, the significance of parameter s on macro-averaged F1 is not big because it also depends on the setting of parameter β. While this is true on macro-averaged level, this figure analyzes the impact of various settings for s when β is fixed to a specific value on the three classification measures. *Motivation*

The x-axis shows the subclasses. The y-axis shows the difference between the result of a subclass with $s = 15000$ and $s = 5000$ as well as $s = 30000$ in percent. Each dot represents a single subclass. Colors and shapes show which of the s setting performs better. Size has no meaning other than increased visibility for lonely dots at the top. The differences are shown in PP. A subclass without any difference between the two settings is not shown. The y-axis is grouped by three classification measures and has the same value range for all three. Parameter β was set to 630 000, which resulted in the best classification performance with $s = 15000$ for WEBSOL. The 100k corpus and LIBLINEAR are used. *Legend*

The difference between the three experiments is significant on recall, where $s = 15000$ outperforms the other two experiments in 80% to 95% of the subclasses. Surprisingly though $s = 15000$ loses in almost 75% of the subclasses on precision compared to $s = 5000$. The reason why $s = 15000$ is still the best performing experiment is because the improvement on recall is higher and happening in more subclasses. *Technical Analysis*

F1 follows accordingly.

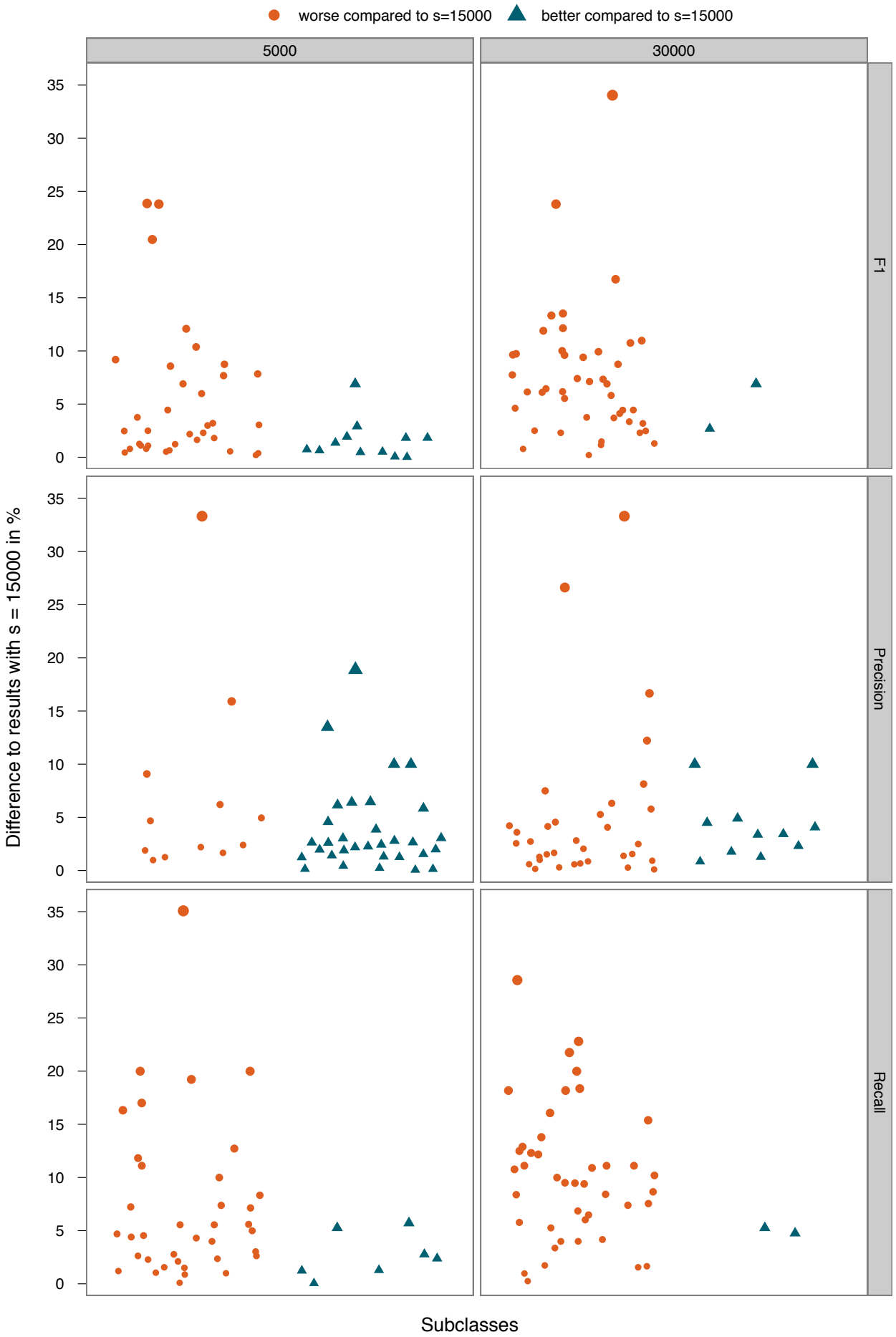


Figure 94: Comparison of the classification for different s settings.

5.6.4 Feature Analysis

5.6.4.1 *Unique Features per Subclass*

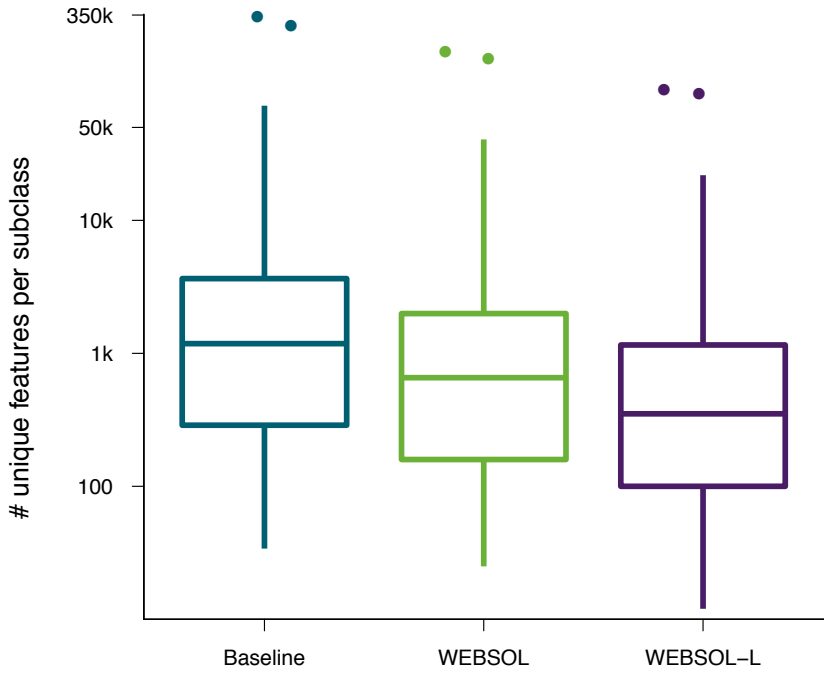


Figure 95: Number of unique features in a subclass for the 100k corpus.

Figure 95. *Introduces*

Corpus-wide unique features should in theory be important to distinguish subclasses from each other. The figure shows the change in the number of unique features between the baseline, WEBSOL and WEBSOL-L. *Motivation*

The y-axis shows the number of corpus-wide unique features for a single subclass on a $\log_{10}$ scale. The x-axis shows the three methods. The 100k corpus is used. *Legend*

While WEBSOL already reduces the number of unique features per subclass compared to the baseline, WEBSOL-L decreases the number even more. *Technical Analysis*

All three also share the same characteristics of not having outliers on the bottom but having exactly two at the top. The outliers in WEBSOL and WEBSOL-L stem from the outlier subclasses of the baseline.

5.6.4.2 *Selected Features per Subclass*

Introduces Figures 96 and 97.

Motivation The figures show the impact of both WEBSOL and WEBSOL-L on the number of features on subclass level.

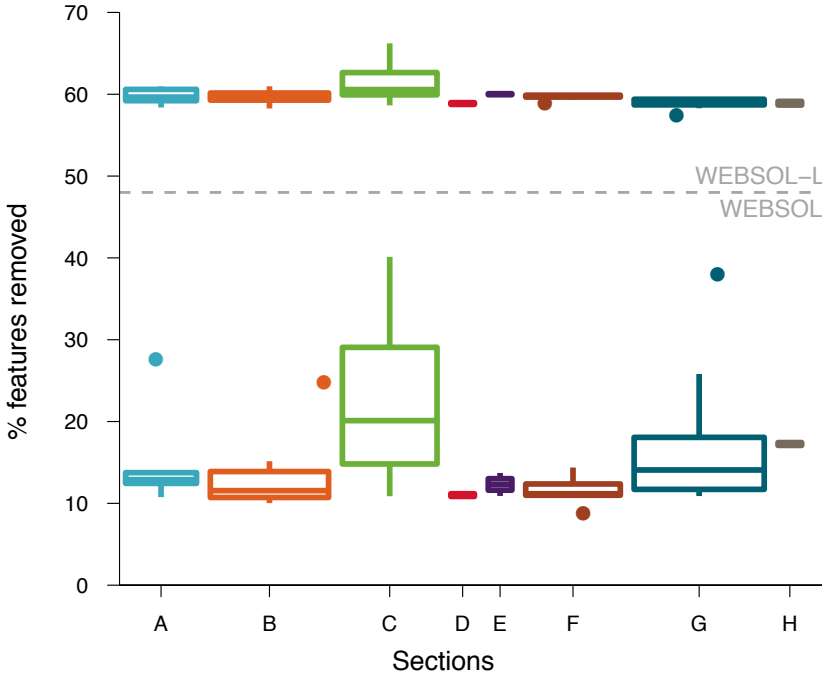
Legend **Figure 96**: **Top**: The y-axis shows the number of removed features in percent to the baseline on the 100k corpus. Both WEBSOL and WEBSOL-L are shown. The x-axis shows the subclasses grouped by section. **Bottom**: Same as the top figure but without WEBSOL-L and for the 200k corpus.

Figure 97: Shows the same data as Figure 96 but with absolute numbers instead of percentages. **Both**: the y-axis shows the number of features per subclass. The axis is in $\log_{10}$ scale. The x-axis shows all the subclasses in descending order by relative reduction of WEBSOL. The best performing classification experiment for the corpus is shown. The color of the line corresponds to the section the subclass belongs to. **Top**: each subclass is shown with a filled rectangle and a line at the bottom of the rectangle. The top of the rectangle shows the number of features in the baseline for the subclass. The bottom of the rectangle shows the number of features for WEBSOL. The bottom of the line shows the number of features for WEBSOL-L. **Bottom**: each subclass of the 200k corpus is represented by a single vertical line. WEBSOL-L is neglected. The top of the line shows the number of features for the baseline. The bottom of the line shows the number of features for WEBSOL.

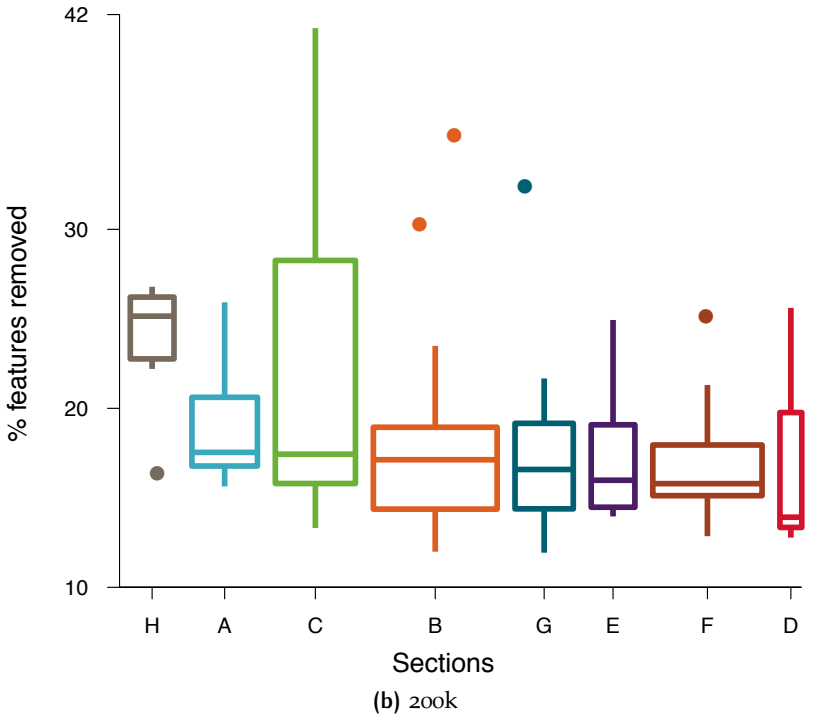
Technical Analysis On all subclasses of the 100k corpus WEBSOL-L removes more features than WEBSOL. Furthermore WEBSOL-L consistently removes about 60% of the features whereas WEBSOL is not as consistent, leaving the smaller subclasses almost untouched with 90% of their original features.

On the 200k corpus, WEBSOL is again inconsistent and prefers subclasses containing more features in the baseline. Therefore most of the change happens in sections B, C and G, sections that have the subclasses with the most documents.

Except for a few outliers the rule is: the more features a subclass has, the more features will be removed in absolute and relative terms. Interestingly, section C, responsible for most features and their reduction, is also the one section providing most of the outliers with the smaller subclasses.



(a) 100k



(b) 200k

Figure 96: Percentage of selected features compared to baseline.

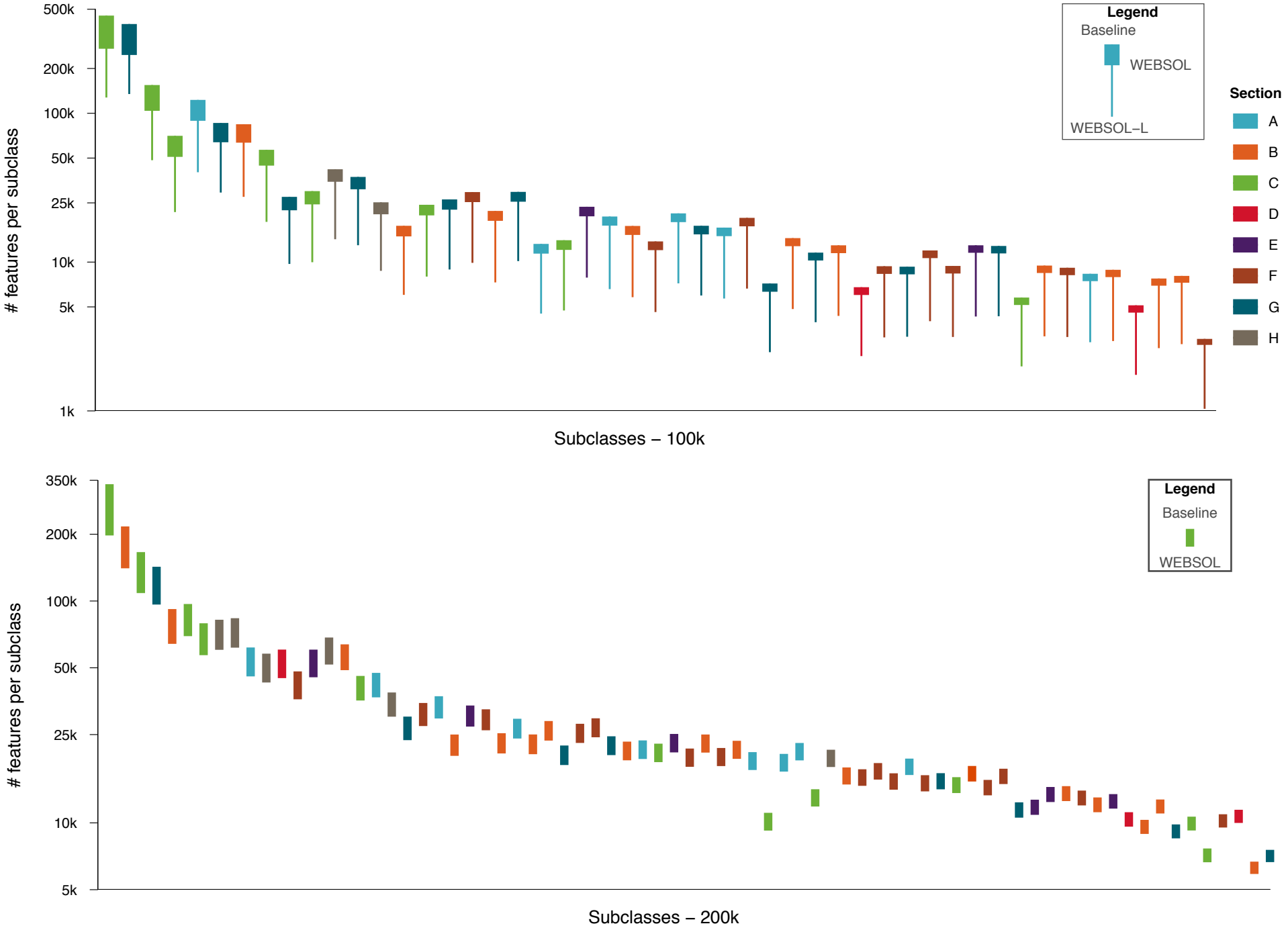


Figure 97: Number of selected features. In descending order by relative reduction by WEBSOL.

5.6.4.3 *Sparseness*

Figure 98a and 98b. *Introduces*

The sparseness of the corpus has a direct impact on the classification. At least fewer resources to read and write files are required if sparseness is low. At best one can use a highly optimized algorithm for dense data structures. The figures are analyzing the impact of WEBSOL and WEBSOL-L on the sparseness and how different both are compared to WEBF1. *Motivation*

The y-axis shows the percentage of features used in a document using $\log_{10}$ scale. The x-axis shows the three different methods plus the baseline for the 100k corpus and two for the 200k corpus. *Legend*

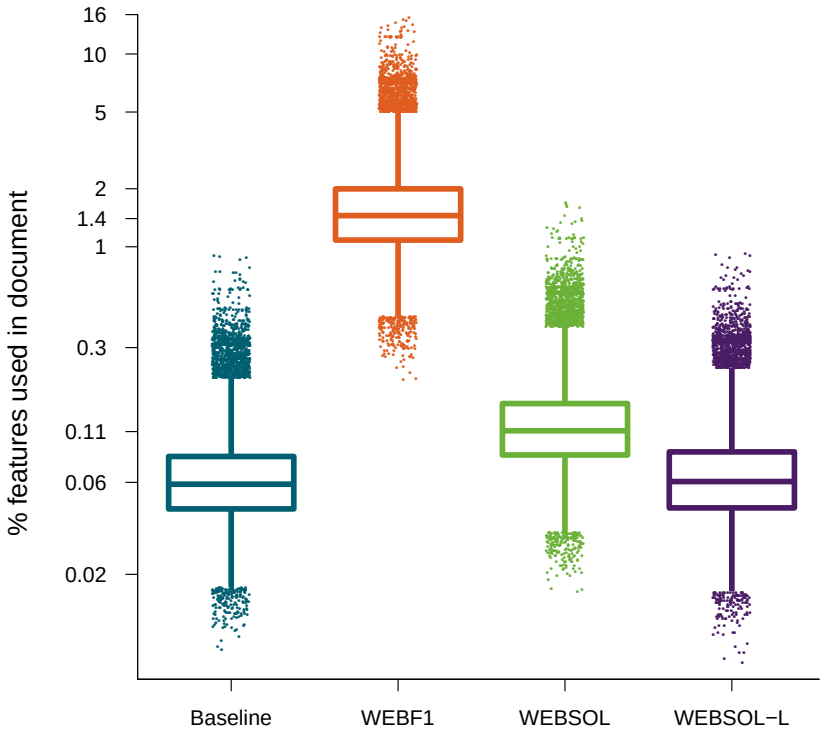
Top: shows the 100k corpus.

Bottom: shows the 200k corpus.

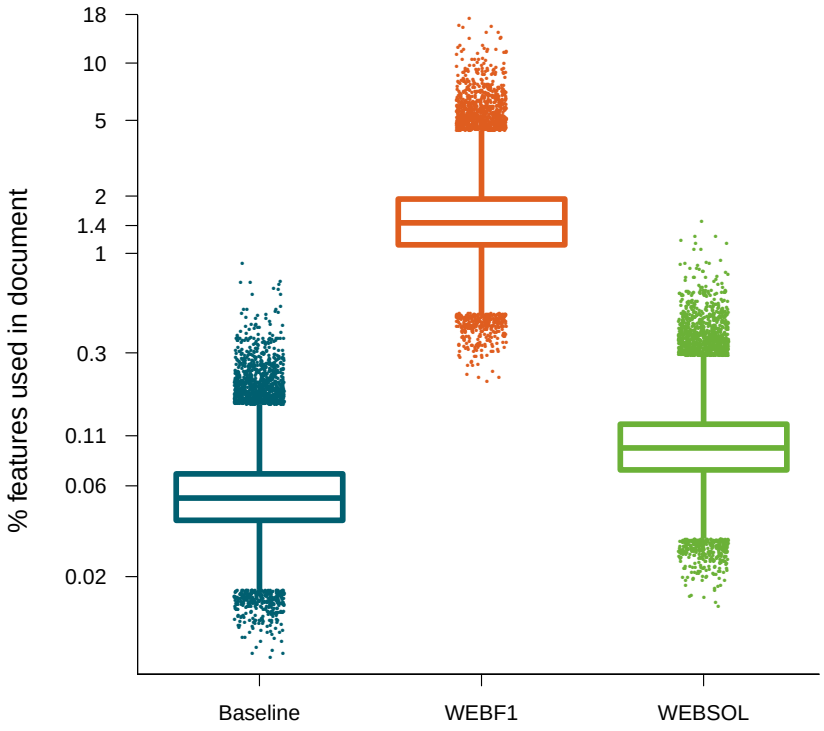
On the 100k corpus, WEBSOL doubles the percentage of unique features in a document compared to the baseline. There are many outliers at both ends of the spectrum for all three methods. While they have almost the same number of documents that are outliers at the top end, WEBSOL-L is able to half the number of outliers at the bottom end compared to the baseline. Both methods are about 1/10th as sparse as WEBF1. *Technical Analysis*

A possible explanation why WEBSOL-L is less sparse is because it removes more bad features, which occur more frequently. The number of features in a document does not change with the same ratio as the number of total features in the corpus.

On the 200k corpus the picture does not change. WEBSOL more than doubles the percentage of unique features in a document compared to the baseline. It has approximately 20% fewer outliers at the bottom end.



(a) 100k



(b) 200k

Figure 98: Sparseness of WEBSOL compared to the baseline, WEBF1 and WEBSOL-L.

5.6.5 Runtime Analysis

5.6.5.1 *Classifier CPU Runtime*

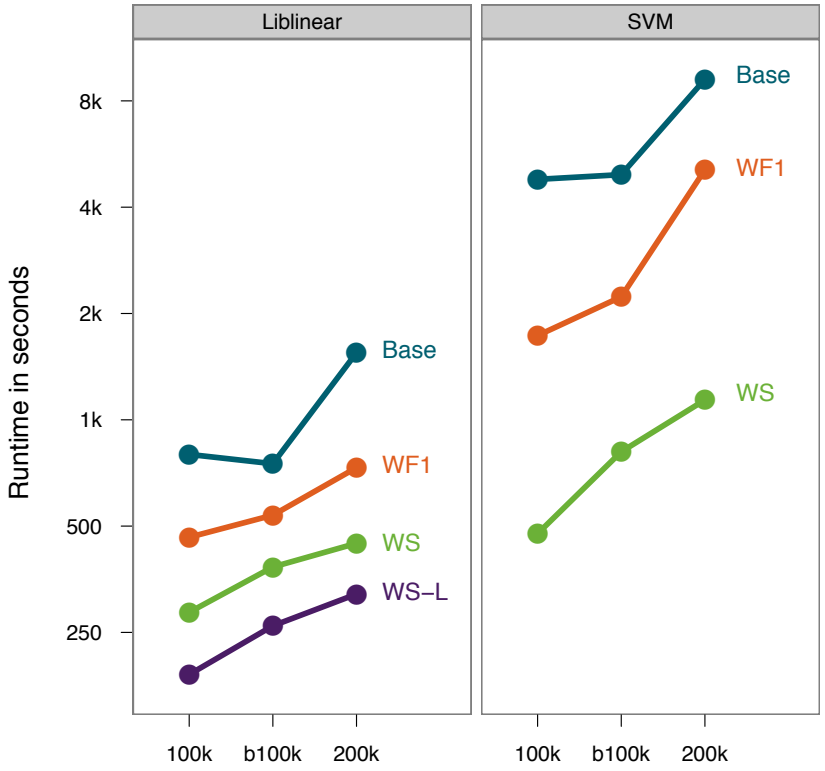


Figure 99: CPU runtime.

Figure 99, references Subsection 5.4.4.1. *Introduces*

The figure analyzes the impact of WEBSOL on the CPU runtime of the training phase of the two classifiers and how much it changes compared to WEBF1. *Motivation*

Legend according to Subsection 5.4.4.1. In addition to the baseline and WEBF1 (WF1), both WEBSOL (WS) and WEBSOL-L (WS-L) are shown for LIBLINEAR and WEBSOL for SVM^{light}. *Legend*

The classifier with WEBSOL is faster than both the baseline and WEBF1 in all six scenarios, despite a significantly higher reduction of features by WEBF1, using only 2% of the features compared to approximately 60% for WEBSOL. Furthermore, WEBSOL-L performs three times faster than WEBF1 and twice as fast as WEBSOL, on all three corpora. *Technical Analysis*

The reason for this is unknown at this point.

For LIBLINEAR there is also a steep incline between the 100k and b100k corpora visible for WEBSOL and WEBSOL-L. The baseline and WEBF1 have no such spike. With such a spike b100k gets close to 200k although it has half the documents and should be finished closer to 100k than 200k. The reason for this is also unknown at this point and should be investigated in future works.

5.6.5.2 *Algorithm CPU Runtime*

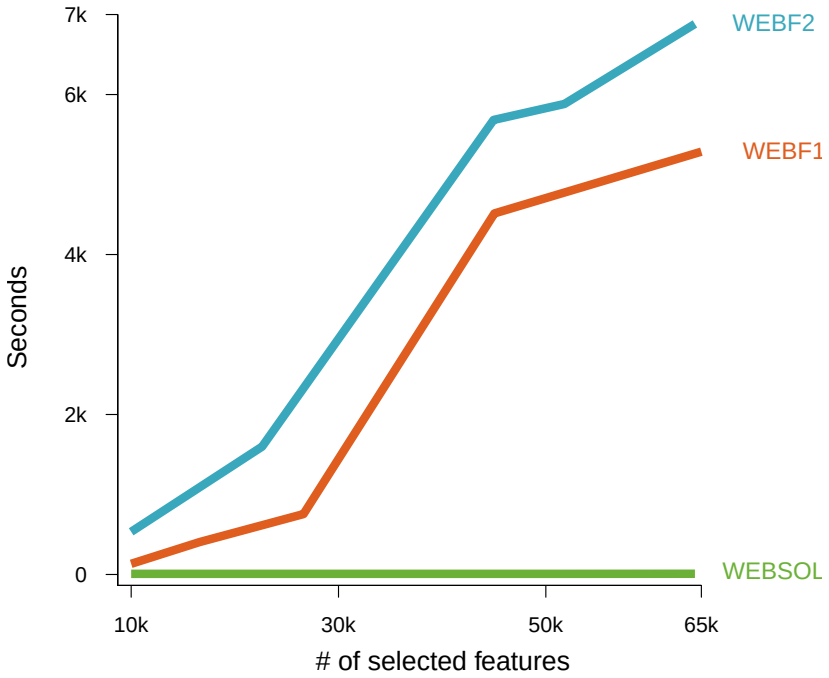


Figure 100: Algorithm CPU runtime with same number of features.

Introduces Figure 100, references Subsection 5.5.4.2.

Motivation This subsection analyzes the CPU runtime of WEBSOL and compares it to that of WEBF1 and WEBF2.

Legend The y-axis shows the seconds each algorithm takes to execute. The x-axis shows the number of selected features, five different data points are shown.

Disclaimer The same disclaimer as in Subsection 5.5.4.2 applies to this subsection too.

Technical Analysis Not surprisingly WEBSOL is faster than both WEBF1 and WEBF2. It is basically constant and finishes within one second, even at 60 000 selected features. It stays under one second even with more than 99% of the features selected.

5.7 CONCLUSION

5.7.1 Overview

	WEBF1	WEBF2	WEBSOL
Easy grid search?	✓✓	✓	✗✗
Subclass independent?	✓✓	✓✓	✓✓
Classifier independent?	✓✓	✓✓	✓✓
Huge feature reduction?	✓✓	✓✓	✗
Decrease feature sparseness?	✓	✓	✗
Decrease unique features?	✓✓	✓✓	✓
Increase F1?	✓	✓	-
Increase precision?	✗	✗	✗
Increase recall?	✓	✓	✓
Low CPU runtime?	-	✗(✗)	✓✓
Low classifier runtime?	-	-	✓
Low on resources?	✓	✓	✓

WEBF1 and WEBF2 are perfect for low resource situations. Both can reduce a dataset to 1% or 2% of the features without sacrificing classification performance, saving many resources and creating a dense dataset. Furthermore they are fast, taking up a small percentage of the classification task that follows. Unless more than 10% of the features have to be kept, which slows both down substantially. They also speed up the classification task that follows. Both require little resources and can be made disk- or memory based without changing the core of the algorithm. Even when only 2% of the features are kept, both WEBF1 and WEBF2 do not have a single document that is trimmed to a blank document. There are always a few features remaining in every document.

WEBSOL is extremely fast, identifies important words accurately and filters out the least important ones, hence only taking a few microseconds after the initial metric is applied onto the corpus. It also leads to the fastest classification task, albeit reducing the number of features by only 40% to 60% — significantly less than WEBF1 or WEBF2 remove.

All three methods have no limit in terms of the number of features or documents, since they concatenate the documents to a single artificial document, making them a perfect method to identify important words in a stream of websites or other areas with a high document creation rate.

5.7.2 Classification Performance

Tables 47, 48 and 49 show the classification performances of WEBF1, WEBF2 and WEBSOL as well as the baseline. Columns show F1, precision (P) and recall (R), all of which are macro-averaged.

	F1	P	R
	%	%	%
Base	82.5	94.3	73.3
WEBF1	84.3	92.9	77.2
WEBF2	84.3	92.9	77.2
WEBSOL	84.1	93.1	76.6

Table 47: 100k.

	F1	P	R
	%	%	%
Base	82.5	92.7	76.3
WEBF1	85.7	90.9	81.0
WEBF2	85.7	90.9	81.0
WEBSOL	85.6	91.1	80.7

Table 48: 200k.

	F1	P	R
	%	%	%
Base	76.2	84.8	70.7
WEBF1	76.3	81.4	71.9
WEBF2	76.4	81.8	71.7
WEBSOL	76.6	82.4	71.6

Table 49: b100k.

6 CONCLUSIONS

As the USPTO gets ever more patent applications, the number of granted patent applications as well as the number of unique words in the patent corpus are constantly increasing. This puts an enormous strain on the patent examiners of the USPTO and decreases the quality of their work and as a result, diminishes the reputation of the patent office. As of today there is a backlog of more than 720 000 patent applications[1], not surprising, considering the USPTO takes seven years for the first action taken on a continuous patent application. However, with a decline in the quality of its work and its reputation, the decisions of the patent office on patent applications are already successfully attacked in court and will get attacked even more in the future (e.g. software patents).

This thesis offers a new approach for minimizing the number of unique words used during the auto-classification of the patent documents. Improving the quality of the auto-classification process and decreasing the resources it needs will inadvertently speed up the process of patent applications and underline the significance and importance of patent offices. Both suffered under severe criticism in the last few years.

Three algorithms, WEBF1, WEBF2 and WEBSOL, have been developed and their impact on the number of unique words is shown in Chapter 5. While all three of the algorithms are filter-based, meaning the actual classifier is not used during the reduction of the words, it was only tested with two linear classifiers, namely SVM^{light} and LIBLINEAR.

The underlying metric for all three algorithms, based on Montemurro and Zanette [33], can easily identify the least and most important words in the documents, at least when the classification is done on subclass level. With such an identification, it is possible to remove up to 99% of the words, while still outperforming the baseline by 4% on macro-averaged F1. Not only does it outperform the baseline on F1, it also outperforms all the other methods on recall. The baseline is even 5% worse on recall than the proposed algorithms. Such a reduction frees up resources that can be used for other tasks and might speed up the time-consuming process of patent applications altogether.

[1] http://www.uspto.gov/dashboards/patents/main.dashxml – visited on 22 Aug, 2011

When compared to other feature selection algorithms, it becomes obvious that WEBF1 and WEBF2 are indeed doing a good job. The other methods reduce the number of features at most by 15%. Even WEBSOL decreases the number by up to 60% without sacrificing the classification performance. To prevent arguments about any kind of corpus bias the three algorithms might have, I tested them on three different corpora, sub-sampled from the USPTO corpus provided by the IRF. On all three corpora, the results are very similar and positive.

This thesis can serve as a basis for further investigations of the Montemurro metric and its potential for feature selection. The method does not only work on patent documents but can be used for any text document. Consequently future work should explore the efficiency of the three proposed algorithms on text corpora with different characteristics than the USPTO corpus used in this thesis (e.g. Reuters Corpora[2], CLEF-IP[3]). Applying it to a larger corpus with millions of documents and thousands of categories should be tried as well. With the ever increasing importance of the World Wide Web, batch algorithms are becoming less useful and being replaced by online algorithms. This is why an online version of this algorithm should be created, opening up completely new fields of usage, making it even more accessible.

[2] http://trec.nist.gov/data/reuters/reuters.html – visited on 22 Aug, 2011

[3] http://www.ir-facility.org/clef-ip – visited on 22 Aug, 2011

A APPENDIX

A.1 BASELINE RESULTS FOR ALL SUBCLASSES

A.1.1 Results alphabetically ordered

Symbol		F1	P	R	Pos
		%	%	%	
A	22B	63.2	85.7	50.0	38
	22C	81.0	77.1	85.5	**7**
	23F	38.7	50.0	31.6	**45**
	45D	75.4	74.9	75.9	13
	61B	87.9	84.5	91.6	**2**
	62B	67.4	74.6	61.5	31
B	01D	72.6	70.1	75.3	19
	22D	74.8	80.8	69.7	16
	23C	71.1	85.7	60.7	25
	25D	57.9	81.5	44.9	39
	27L	73.8	73.2	74.4	17
	44B	36.3	50.0	28.6	**46**
	60B	75.2	78.3	72.3	14
	60G	42.3	64.7	31.4	**44**
	60P	80.1	76.8	83.7	**10**
	62K	68.0	81.0	58.6	28
C	01G	69.3	79.6	61.4	27
	02F	64.2	72.2	57.8	37
	07C	82.3	80.2	84.6	**6**
	07K	83.9	80.8	87.1	**4**
	08H	65.6	76.9	57.1	35
	11D	71.3	76.1	67.1	21
	13F	45.2	63.6	35.0	**43**
	25B	70.2	74.6	66.3	26

Symbol		F1	P	R	Pos
		%	%	%	
D	04D	79.5	80.5	78.6	11
	06Q	50.0	64.3	40.9	42
E	03D	52.6	76.9	40.0	41
	04H	65.6	75.0	58.3	34
F	01B	66.2	63.8	68.8	32
	21V	83.0	80.1	86.1	5
	22G	57.1	100	40.0	40
	23C	74.5	74.1	75.0	15
	23R	71.1	78.7	64.9	23
	24B	35.3	87.5	22.1	47
	25D	90.2	94.6	86.2	1
	41C	18.2	50.0	11.1	48
G	01C	65.4	69.8	61.6	33
	01G	0.0	100.0	0.0	49
	01T	73.2	77.0	69.8	18
	04B	72.7	76.1	69.5	20
	04F	86.5	82.1	91.4	3
	06F	81.9	85.7	78.4	8
	06G	71.6	82.8	63.2	24
	06J	80.5	74.2	88.0	9
	10G	65.3	79.3	55.4	36
	10H	67.9	70.2	65.8	29
	11B	79.0	83.8	74.7	12
H	03H	67.1	70.7	63.9	30
	05B	71.5	76.0	67.4	22

Table 50: Performance of the baseline on subclass level on the 100k corpus with LIBLINEAR in alphabetical order. Position in descending order by F1 performance. F1, precision (P) and recall (R) are shown. Blue position numbers are subclasses in the top 10, red numbers are in the bottom 10.

A.1.2 Results ordered by Rank

	F1	P	R	Pos	S
	%	%	%		
F25D	90.2	94.6	86.2	1	F
A61B	87.9	84.5	91.6	2	A
G04F	86.5	82.1	91.4	3	G
C07K	83.9	80.8	87.1	4	C
F21V	83.0	80.1	86.1	5	F
C07C	82.3	80.2	84.6	6	C
G06F	81.9	85.7	78.4	8	G
A22C	81.0	77.1	85.5	7	A
G06J	80.5	74.2	88.0	9	G
B60P	80.1	76.8	83.7	10	B
D04D	79.5	80.5	78.6	11	D
G11B	79	83.8	74.7	12	G
A45D	75.4	74.9	75.9	13	A
B60B	75.2	78.3	72.3	14	B
F23C	74.5	74.1	75	15	F
B22D	74.8	80.8	69.7	16	B
B27L	73.8	73.2	74.4	17	B
G01T	73.2	77.0	69.8	18	G
B01D	72.6	70.1	75.3	19	B
G04B	72.7	76.1	69.5	20	G
G06G	71.6	82.8	63.2	21	G
H05B	71.5	76.0	67.4	22	H
C11D	71.3	76.1	67.1	23	C
F23R	71.1	78.7	64.9	24	F
B23C	71.1	85.7	60.7	25	B
C25B	70.2	74.6	66.3	26	C
C01G	69.3	79.6	61.4	27	C
B62K	68.0	81.0	58.6	28	B
G10H	67.9	70.2	65.8	29	G
H03H	67.1	70.7	63.9	30	H
A62B	67.4	74.6	61.5	31	A
F01B	66.2	63.8	68.8	32	F

	F1	P	R	Pos	S
	%	%	%		
E04H	65.6	75.0	58.3	33	E
C08H	65.6	76.9	57.1	34	C
G01C	65.4	69.8	61.6	35	G
G10G	65.3	79.3	55.4	36	G
C02F	64.2	72.2	57.8	37	C
A22B	63.2	85.7	50.0	38	A
B25D	57.9	81.5	44.9	39	B
F22G	57.1	100	40.0	40	F
E03D	52.6	76.9	40.0	41	E
D06Q	50.0	64.3	40.9	42	D
C13F	45.2	63.6	35.0	43	C
B60G	42.3	64.7	31.4	44	B
A23F	38.7	50.0	31.6	45	A
B44B	36.3	50.0	28.6	46	B
F24B	35.3	87.5	22.1	47	F
F41C	18.2	50.0	11.1	48	F
G01G	0.0	100.0	0.0	49	G

Table 51: Performance of the baseline on subclass level on the 100k corpus with LIBLINEAR ordered by rank. Last column shows the section of the subclass. F1, precision (P) and recall (R) are shown.

A.2 MENTIONED SUBCLASSES AND THEIR SYMBOLS

Symbol		Name
A	22B	Slaughtering
	01P	Biocidal, pest repellant, pest attractant or plant growth regulatory activity of chemical compounds or preparations
	22C	Processing meat, poultry, or fish
	23F	Coffee; tea; their substitutes; manufacture, preparation, or infusion thereof
	45D	Hairdressing or shaving equipment; manicuring or other cosmetic treatment
	61B	Diagnosis; surgery; identification
	62B	Devices, apparatus, or methods for life-saving
B	01D	Separation
	22D	Casting of metals; casting of other substances by the same processes or devices
	23C	Milling
	25D	Percussive tools
	27L	Removing bark or vestiges of branches
	44B	Machines, apparatus, or tools for artistic work, e.g. for sculpturing, guilloching, carving, branding, inlaying
	60B	Vehicle wheels; castors; axles; increasing wheel adhesion
	60G	Vehicle suspension arrangements
	60P	Vehicles adapted for load transportation or to transport, to carry, or to comprise special loads or objects
	62K	Cycles; cycle frames; cycle steering devices; rider-operated terminal controls specially adapted for cycles; cycle axle suspensions; cycle sidecars, forecars, or the like
	81C	Processes or apparatus specially adapted for the manufacture or treatment of micro-structural devices or systems
	82B	Nano-structures; manufacture or treatment thereof

Symbol		Name
C	01G	Compounds containing metals not covered by subclasses
	02F	Treatment of water, waste water, sewage, or sludge
	07C	Acyclic or carbocyclic compounds
	07K	Peptides
	08H	Derivatives of natural macromolecular compounds
	11D	Detergent compositions
	13F	Preparation or processing of raw sugar, sugar, or syrup
	25B	Electrolytic or electrophoretic processes for the production of compounds or non- metals; apparatus therefor
D	04D	Trimmings; ribbons, tapes, or bands, not otherwise provided for
	06Q	Decorating textiles
E	03D	Water-closets or urinals with flushing devices; flushing valves therefor
	04H	Buildings or like structures for particular purposes; swimming or splash baths or pools; masts; fencing; tents or canopies, in general
F	01B	Machines or engines, in general or of positive-displacement type, e.g. steam engines
	21V	Functional features or details of lighting devices or systems thereof; structural combinations of lighting devices with other articles, not otherwise provided for
	22G	Superheating of steam
	23C	Combustion apparatus using fluent fuel
	23R	Generating combustion products of high pressure or high velocity, e.g. gas-turbine combustion chambers
	24B	Domestic stoves or ranges for solid fuels; implements for use in connection with stoves or ranges
	25D	Refrigerators; cold rooms; ice-boxes; cooling or freezing apparatus not covered by any other subclass
	41C	Smallarms, e.g. pistols, rifles

Symbol		Name
G	01C	Measuring distances, levels, or bearings; surveying; navigation; gyroscopic instruments; photogrammetry
	01G	Weighing
	01N	Investigating or analysing materials by determining their chemical or physical properties
	01T	Measurement of nuclear or x-radiation
	04B	Mechanically-driven clocks or watches; mechanical parts of clocks or watches in general; timepieces using the position of the sun, moon, or stars
	04F	Time-interval measuring
	06F	Electric digital data processing
G	06G	Analogue computers
	06J	Hybrid computing arrangements
	10G	Aids for music
	10H	Electrophonic musical instruments
	11B	Information storage based on relative movement between record carrier and transducer
	99Z	Subject matter not otherwise provided for in this section
H	03H	Impedance networks, e.g. resonant circuits; resonators
	05B	Electric heating; electric lighting not otherwise provided for

Table 52: The names for all mentioned subclass symbols.

A.3 PARTICIPANTS OF SURVEY

Name	Affiliation
Mark Ashworth	Intellectual Property Exchange
Hynek Bakstein	EPO
Simon Carr	Kraft Foods
Franz Eder	Vienna UT
Allan Hanbury	IRF
G. Krawczyk	
Parvaz Mahdabi	Universita della Svizzera italiana
Yousaf Malik	
Evert Nijhof	ASML
Stefan Sieberer	TU Wien

Table 53: All non-anonymous participants of the survey conducted to check the results of the Montemurro metric.

A.4 FULL KIND CODES OF PATENTS

	Kind Code	Kind of document
A	A	Utility Patent Grant issued prior to January 2, 2001.
	A1	Utility Patent Application published on or after January 2, 2001
	A2	Second or subsequent publication of a Utility Patent Application
	A9	Correction published Utility Patent Application
B	Bn	Reexamination Certificate issued prior to January 2, 2001. NOTE: "n" represents a value 1 through 9.
	B1	Utility Patent Grant (no pre-grant publication) issued on or after January 2, 2001.
	B2	Utility Patent Grant (with pre-grant publication) issued on or after January 2, 2001.
C	Cn	Reexamination Certificate issued on or after January 2, 2001. NOTE: "n" represents a value 1 through 9 denoting the publication level.

	Kind Code	Kind of document
E	E	Reissue Patent
H	H	Statutory Invention Registration (SIR) Patent Documents. SIR documents began with the December 3, 1985 issue.
I	I1	"X" Patents issued from July 31, 1790 to July 13, 1836.
	I2	"X" Reissue Patents issued from July 31, 1790 to July 13, 1836.
	I3	Additional Improvements – Patents issued between 1838 and 1861.
	I4	Defensive Publication – Documents issued from November 5, 1968 through May 5, 1987.
	I5	Trial Voluntary Protest Program (TVPP) Patent Documents
P	P	Plant Patent Grant issued prior to January 2, 2001
	P1	Plant Patent Application published on or after January 2, 2001
	P2	Plant Patent Grant (no pre-grant publication) issued on or after January 2, 2001.
	P3	Plant Patent Grant (with pre-grant publication) issued on or after January 2, 2001.
	P4	Second or subsequent publication of a Plant Patent Application
S	S	Design

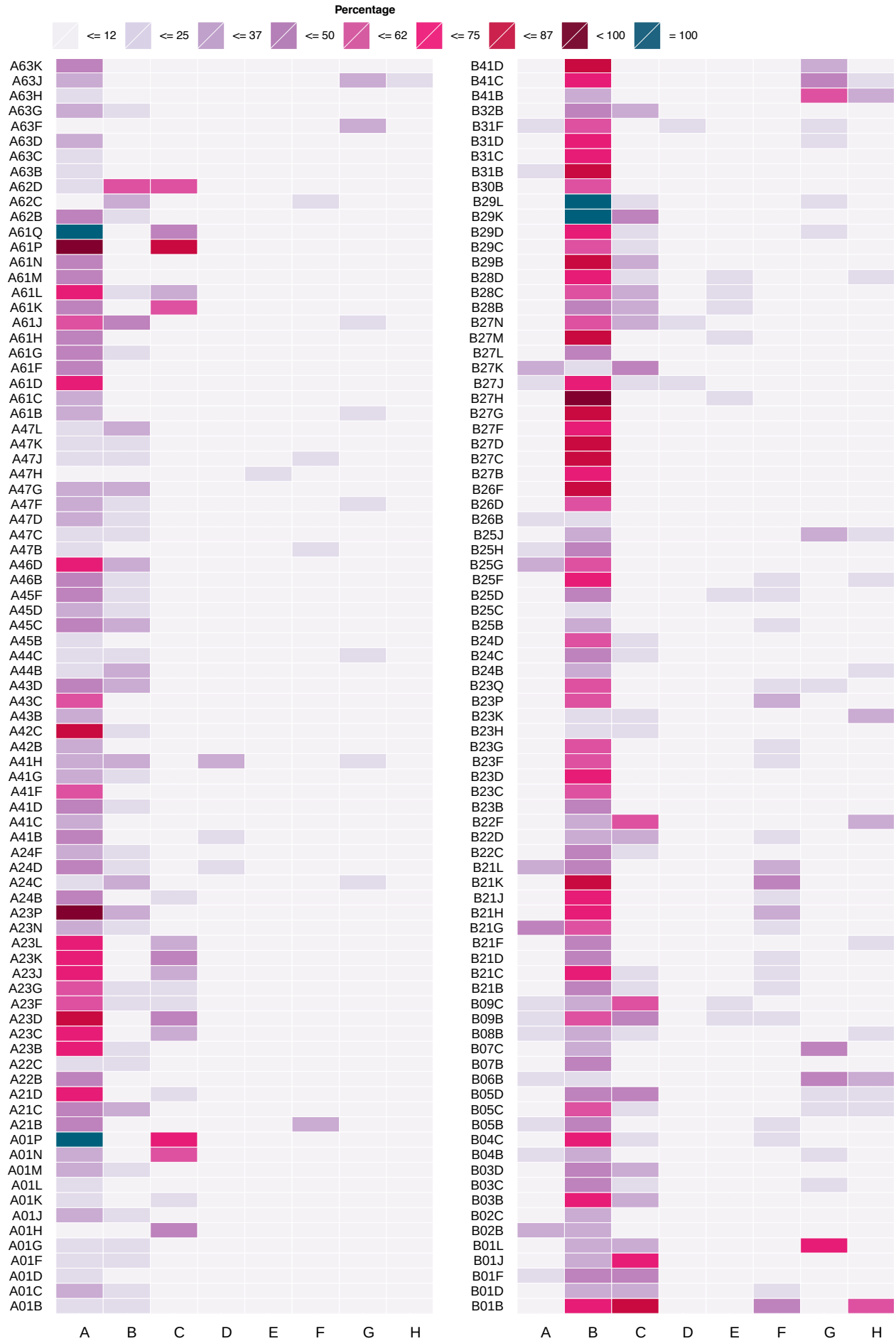
Table 54: All kind codes/stages of a patent in alphabetical order. Kind codes are taken from the USPTO website[1].

A.5 CONNECTIVITY OF SUBCLASSES

The following nine graphs show the connectivity of every subclass to every section. The connectivity from one subclass to another is presented in Section 3.5. As can be seen in the thesis, on average a subclass is connected to approximately 250 other subclasses, resulting in a network with only a few nodes (635) but hundreds of thousands of edges.

Since section B contains 170 subclasses, it is spread over two columns.

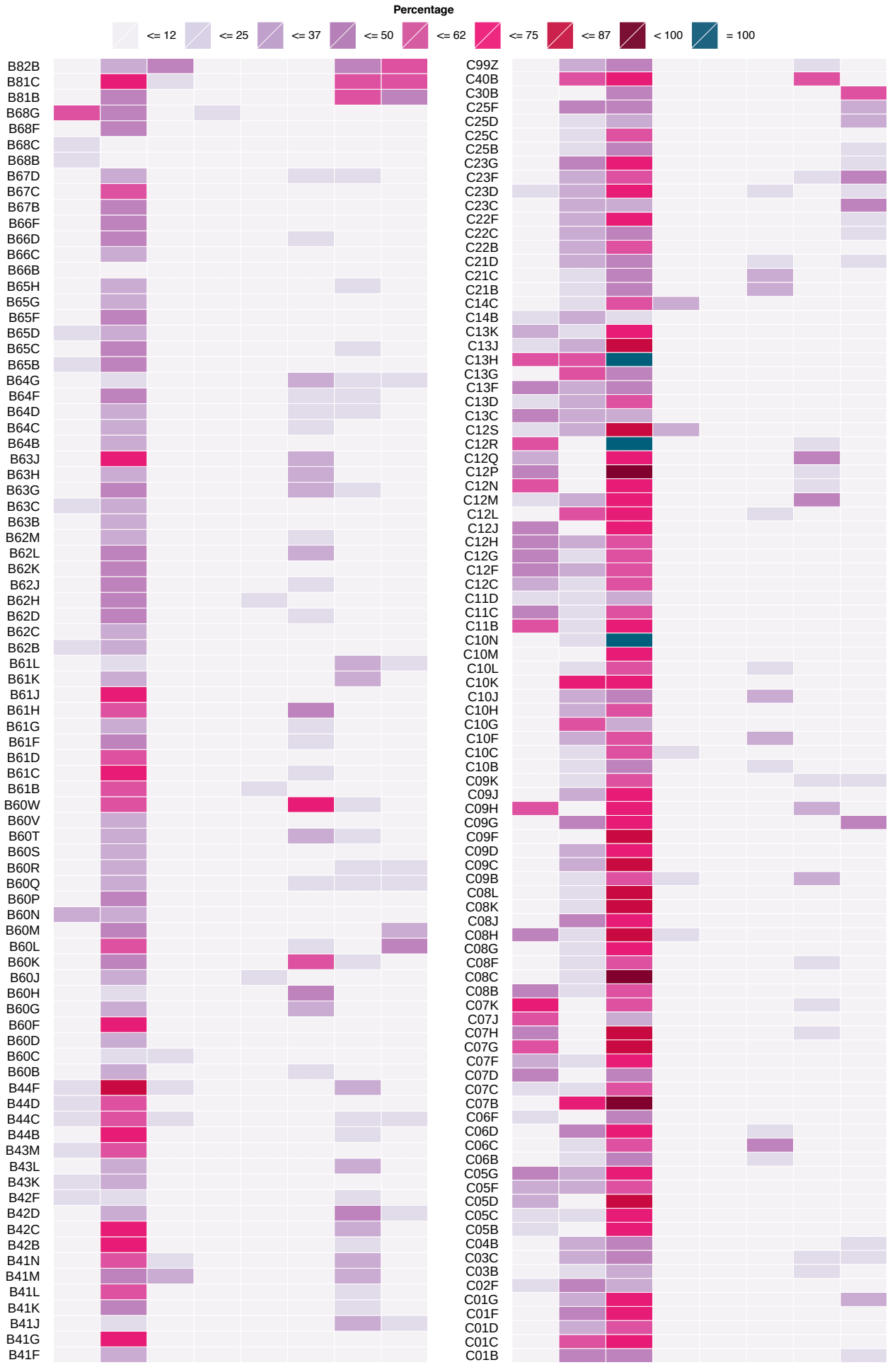
The darker a cell, the higher the connectivity between the subclass on the y-axis and subclasses from the section on the x-axis. A white space visualizes a lack of connections between the subclass and the section. A characteristic of the connectivity is the fact that a subclass from one section is extremely likely to be connected to subclasses from its own section. There are exceptions to this rule, for example B82B and B81C, which both are also highly connected to subclasses from sections G and H. For subclass B28B there is a higher connection to subclasses from sections C, G and H than there is to subclasses originating from section B. Why a subclass with the title “Shaping Clay or Other Ceramic Compositions, Slag or Mixtures Containing Cementitious Material, E.G. Plaster" is so highly connected to subclasses from the sections “Physics" (G) and “Electricity" (H) is beyond the scope of this thesis.



(a) Section A

(b) First part of section B

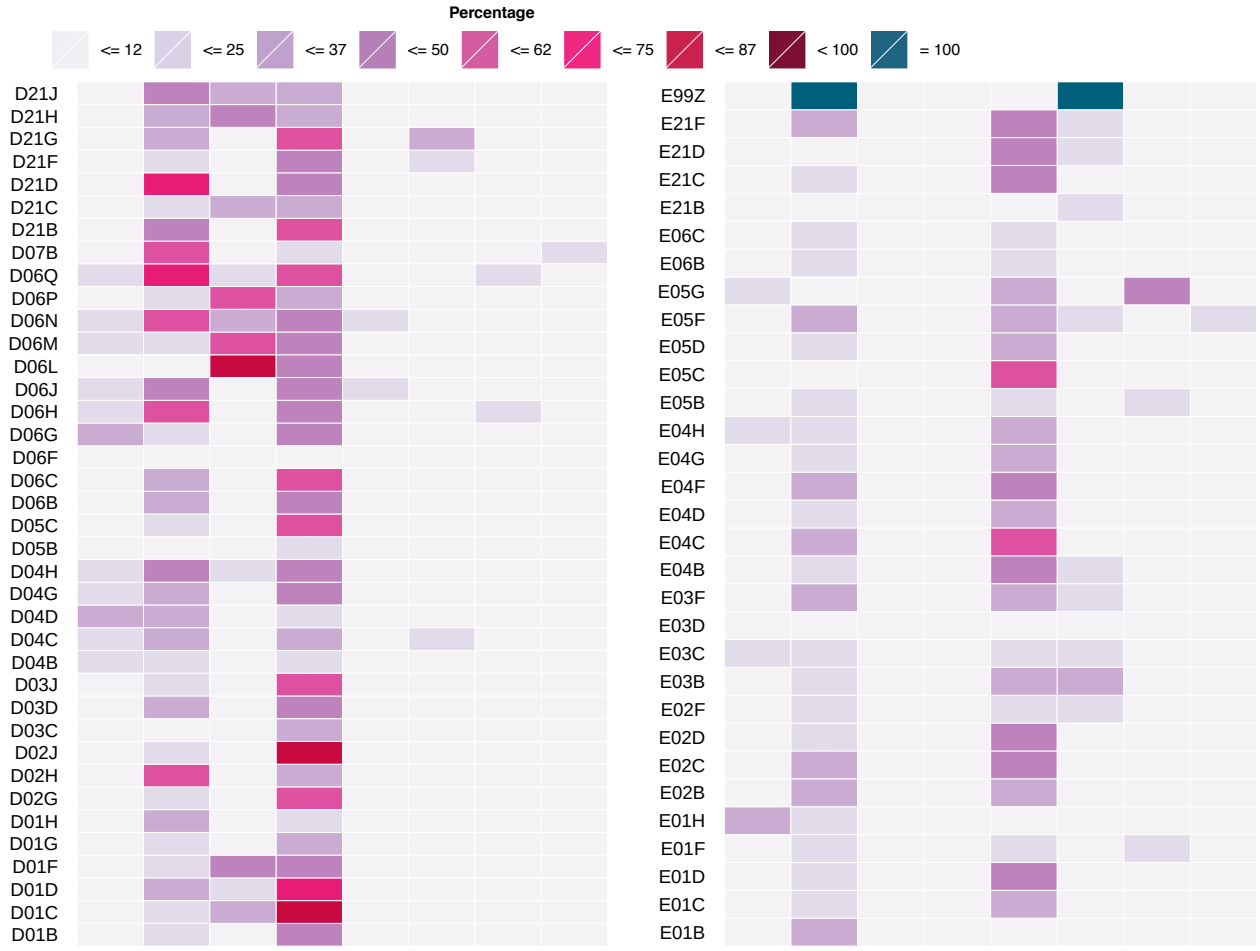
Figure 101: All subclasses from section A and first half of section B.



(a) Second part of section B

(b) Section C

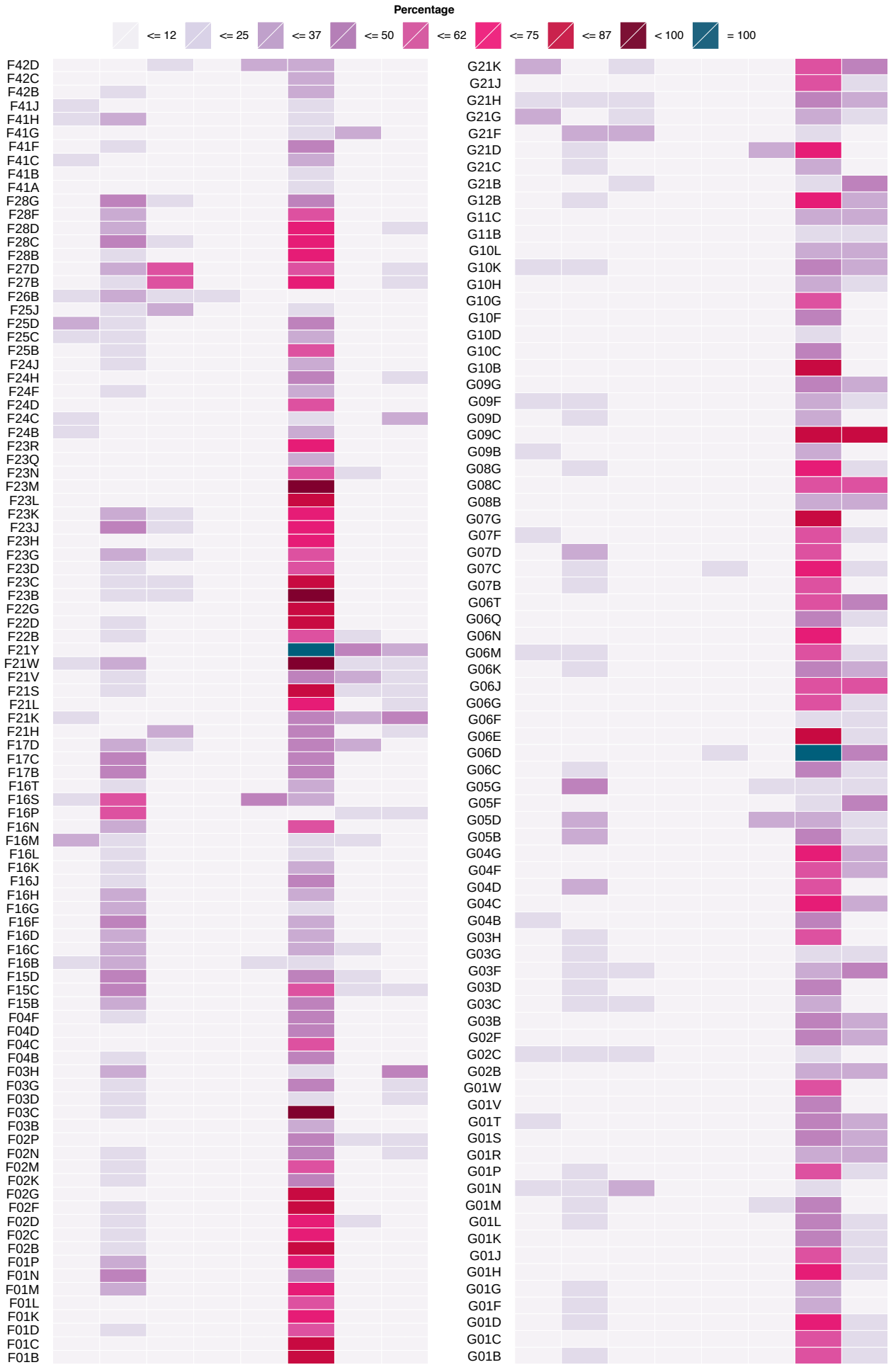
Figure 102: Second part from section B and all subclasses from section C.



(a) Section D

(b) Section E

Figure 103: All subclasses from section D and E.



(a) Section F

(b) Section G

Figure 104: All subclasses from section F and G.

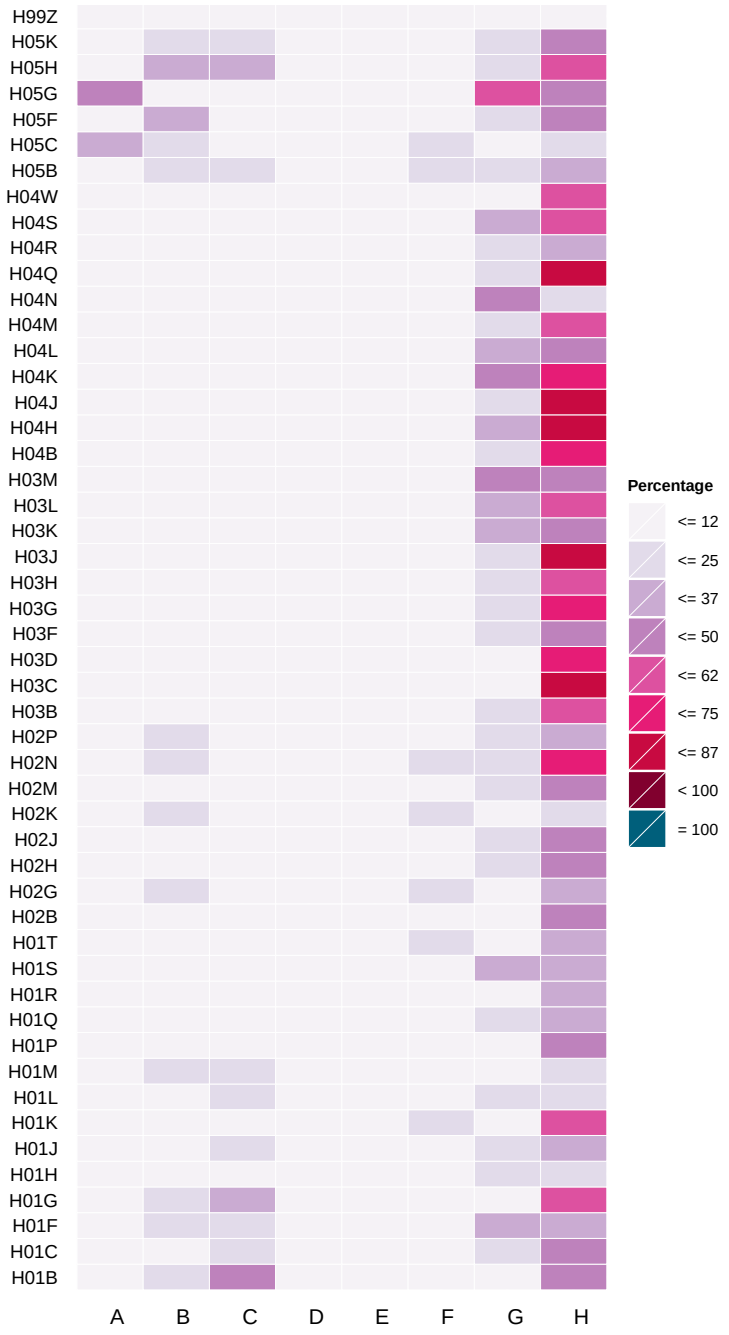


Figure 105: All subclasses from H.

WORKS CITED

[1] S. Amari and S. Wu. "Improving support vector machine classifiers by modifying kernal functions." In: *Neural Netw.* 12.6 (1999), pages 783–789. (Cited on page 15).

[2] K. H. Atkinson. "Toward a More Rational Patent Search Paradigm." In: PaIR 08, 2008. (Cited on page 51).

[3] D. Becks et al. "Patent Retrieval Experiments in the Context of the CLEF IP Track 2009." In: *Multilingual Information Access Evaluation I. Text Retrieval Experiments*. Volume 6241. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2010, pages 491–496. (Cited on page 18).

[4] K. Beuls, A. Hanbury, and B. Pflugfelder. "Comparative Analysis of Balanced Winnow and SVM in Large Scale Patent Categorization." In: *Proceedings of the 10th Dutch-Belgian Information Retrieval Workshop, 2010*. 2010, pages 8–15. (Cited on page 10).

[5] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007. (Cited on page 14).

[6] A. Bordes et al. "Fast Kernel Classifiers with Online and Active Learning." In: *Journal of Machine Learning Research* 6 (2005), pages 1579–1619. URL: http://leon.bottou.org/papers/bordes-ertekin-weston-bottou-2005. (Cited on pages 15, 16).

[7] L. Bottou. URL: http://leon.bottou.org/projects/lasvm (visited on 01/31/2009). (Cited on pages 15, 16).

[8] L. Breinman et al. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984. (Cited on page 23).

[9] EPO. URL: http://www.epo.org/patents/Grant-procedure/About-patents.html (visited on 03/01/2011). (Cited on page 1).

[10] S. Ertekin et al. "Learning on the Border: Active Learning in Imbalanced Data Classification." In: *Proceedings of the 16th Conference on Information and KnowledgeManagement, CIKM2007*. Lisboa: ACM Press, 2007. (Cited on page 16).

[11] C. J. Fall et al. "Automated categorization in the international patent classification." In: *SIGIR Forum* 37.1 (2003), pages 10–25. (Cited on pages 9, 10, 18).

[12] R.-E. Fan et al. "LIBLINEAR: A Library for Large Linear Classification." In: *Journal of Machine Learning Research* 9 (2008). (Cited on page 16).

[13] T. Fawcett. "An introduction to ROC analysis." In: *Pattern Recognition Letters* 27.8 (2006), pages 861 –874. (Cited on page 154).

[14] R. Feldman and J. Sanger. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, 2006. (Cited on page 11).

[15] W. Feller. *An Introduction to Probability Theory and Its Applications*. Wiley, 1968, pages 50–53. (Cited on page 77).

[16] S. Few. *Time on the Horizon*. Technical report. 2008. (Cited on page 42).

[17] T. M. J. Fruchterman and E. M. Reingold. "Graph drawing by force-directed placement." In: *Softw. Pract. Exper.* 21 (11 1991), pages 1129–1164. (Cited on page 62).

[18] B. Goetz. *Java Concurrency in Practice*. Addison-Wesley Longman, 2006. (Cited on page 75).

[19] I. Guyon and A. Elisseeff. "An Introduction to Variable and Feature Selection." In: *Journal of Machine Learning Research* (2003). (Cited on page 21).

[20] J. Heer, N. Kong, and M. Agrawala. "Sizing the horizon: the effects of chart size and layering on the graphical perception of time series visualizations." In: *Proceedings of the 27th international conference on Human factors in computing systems*. CHI 2009. ACM, 2009. (Cited on page 42).

[21] D. Hunt, L. Nguyen, and M. Rodgers. *Patent Searching: Tools and Techniques*. John Wiley & Sons, Inc., 2007. (Cited on page 67).

[22] T. Joachims. *Learning to Classify Text using Support Vector Machines*. Springer, 2002. (Cited on pages 12, 16).

[23] T. Joachims. "Transductive Inference for Text Classification using Support Vector Machines." In: *International Conference on Machine Learning (ICML)* (1999). (Cited on pages 15, 16).

[24] V. Kecman. *Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models*. MIT Press, 2001. (Cited on page 13).

[25] J. Kim et al. "Patent Document Retrieval and Classification at KAIST." In: NTCIR-5 Workshop Meeting, 2005., (cited on page. 11).

[26] J.-H. Kim and K.-S. Choi. "Patent document categorization based on semantic structural information." In: *Inf. Process. Manage.* 43 (5 2007), pages 1200–1215. (Cited on page 10).

[27] S. Kullback. *Information theory and statistics*. John Wiley and Sons, 1959. (Cited on page 72).

[28] X. Li et al. "Automatic patent classification using citation network information: an experimental study in nanotechnology." In: *JCDL '07: Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*. Vancouver, BC, Canada: ACM, 2007, pages 419–427. (Cited on page 10).

[29] N. Littlestone. "Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm." In: *Machine Learning*. 1988, pages 285–318. (Cited on page 10).

[30] H. Liu. "Evolving feature selection." In: *IEEE Intell. Syst.* 20.6 (2005), pages 64–76. (Cited on page 72).

[31] Y.-H. Lu and Y. Huang. "Document Categorization with Entropy Based TF/IDF Classifier." In: *GCIS '09: Proceedings of the 2009 WRI Global Congress on Intelligent Systems*. Washington, DC, USA: IEEE Computer Society, 2009, pages 269–273. (Cited on page 19).

[32] D. Manning. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999. (Cited on page 70).

[33] M. Montemurro and D. Zanette. "Towards the quantification of the semantic information encoded in written language." In: (2009). (Cited on pages 3, 5, 19, 69, 72, 74, 76, 173).

[34] European Patent Office. *12th EPO Patent Information Beginners Seminar*. 2008. (Cited on page 52).

[35] United States Government Accountability Office. *Hiring Efforts Are Not Sufficient to Reduce the Patent Application Backlog*. Technical report. 2007. URL: `http://www.gao.gov/new.items/d071102.pdf`. (Cited on page 32).

[36] H. Peng, F. Long, and C. Ding. "Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 27.8 (2005), pages 1226–1238. (Cited on page 72).

[37] M.F. Porter. "Readings in information retrieval." In: Morgan Kuafmann, 1997, pages 313 –316. (Cited on page 17).

[38] G. Salton and C. Buckley. *Term Weighting Approaches in Automatic Text Retrieval*. Technical report. Ithaca, NY, USA, 1987. (Cited on page 18).

[39] F. Sebastiani. "Machine Learning in Automated Text Categorization." In: *ACM Computing Surveys, Vol. 34, No. 1, pp. 1-47* (2002). (Cited on page 19).

[40] J. Sengupta. "Econometrics of Information and Efficiency." In: Springer, 2009, pages 13–18. (Cited on page 71).

[41] M. Sewell. *Structural Risk Minimization*. Technical report. University College London, 2008. (Cited on page 13).

[42] C. E. Shannon. "A Mathematical Theory of Communication." In: *Bell System Technical Journal* (1948). (Cited on page 70).

[43] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1992. (Cited on page 24).

[44] J. W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977. (Cited on page 24).

[45] S. Verberne et al. "Quantifying the Challenges in Parsing Patent Claims." In: *Proceedings of the 1st International Workshop on Advances in Patent Information Retrieval (AsPIRe 2010)*. 2010, pages 14–21. (Cited on page 52).

[46] L. Yu and H. Liu. "Efficient Feature Selection via Analysis of Relevance and Redundancy." In: *J. Mach. Learn. Res.* 5 (2004). (Cited on page 11).

[47] G. K. Zipf. *Human behavior and the principle of least effort*. Addison-Wesley, 1950. (Cited on page 79).