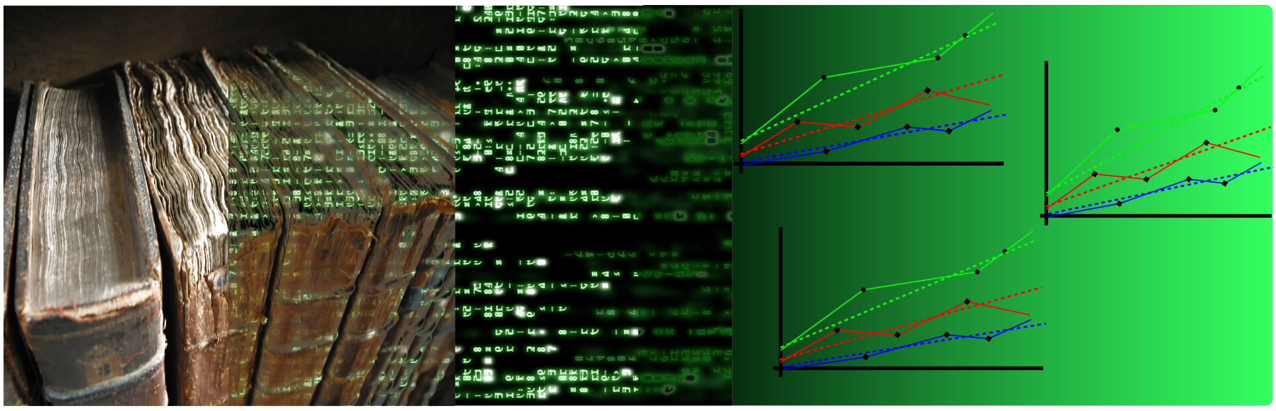
CHALMERS



A Methodology and Algorithm for Automatically Classifying Text Documents to Strategic Intents

Master's thesis in Engineering Mathematics

ROBIN KARMAKAR

Department of Mathematics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2011

Master's thesis 2011:01

MASTER'S THESIS IN ENGINEERING MATHEMATICS

A Methodology and Algorithm for Automatically Classifying Text Documents to Strategic Intents

ROBIN KARMAKAR

Department of Mathematics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2011

A Methodology and Algorithm for Automatically
Classifying Text Documents to Strategic Intents
ROBIN KARMAKAR

Master's thesis 2011:01
ISSN 1652-8557
Department of Mathematics
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone: +46 (0)31-772 1000

Cover:
Transformation of documents into graphical trend analyses.

Chalmers Reproservice
Gothenburg, Sweden 2011

A Methodology and Algorithm for Automatically
Classifying Text Documents to Strategic Intents
Master's thesis in Engineering Mathematics
ROBIN KARMAKAR
Department of Mathematics
Chalmers University of Technology

Abstract

The thesis develops and presents a strategy, methodology and tool for automatically classifying documents according to strategic intents and describes how to monitor changes in this distribution over time. The methodology uses artificial neural networks and latent semantic indexing to evaluate the similarity of documents to generate the proper allocation to a set of predefined clusters. The documents to be classified are patents and publications. The document classes are based on a company internal classification scheme, that is strategically aligned with the company's business objectives. The methodology is structured and constructed as a process considering the steps from raw data acquisition to the final output: a classification of documents and their temporal distributions. The results show that using the internally defined strategic intents supports a high-level analysis of technology areas and competitor activity within the same. The classification algorithm performed well to classify different sources of information, with best performance F-measure values in the range $0.73 - 0.90$ depending on the dataset used. For publications the algorithm performed well given that the number of training documents in the prelabeled set was sufficient. For patents the algorithm performs well, even if the classifier is created and trained using publications. Performance also increases with the ratio of prelabelled to unlabelled documents. The number of neurons in the hidden layer does not significantly affect classifier performance, but the number of correcting iterations does. In choosing between a high number of neurons or a high number of iterations, for a given computational effort the focus should be on increasing the number of iterations. Finally the thesis shows that company specific publishing trends can successfully be analyzed and evaluated over time using the suggested framework.

Keywords: automatic text categorization, document classification, artificial neural networks, latent semantic analysis, trend analysis, strategic analysis

Sammanfattning

Uppsatsen utvecklar samt beskriver en strategi, metod och verktyg för att automatisk klassificera dokument enligt strategic intents och analysera dessas utveckling över tid. Metodologin använder sig av artificiella neurala nätverk och latent semantisk indexering för att utvärdera dokuments likhet samt för att allokera dessa enligt ett antal fördefinierade grupper. Dokumenten som klassificerades var patent och vetenskapliga artiklar. Dokumentklasserna är baserade på ett företagsinternt klassifikationsschema, som tagits fram för att vara i linje med företagets affärsmål. Metodologin är strukturerad som en process som beskriver samtliga steg, från datahämtning till den slutgiltiga analysen av datan - färdigklassificerade dokument och dessas distribution över tid. Resultaten visar att den företagsinterna klassifikationen strategic intents bidrar till analyser av teknologiområden vid en högre abstraktionsnivå samt konkurrenters aktivitet inom dessa. Klassificeringsalgoritmen presterar väl för att klassificera olika typer av data, med F-measure-värden inom intervallet $0.73 - 0.90$ beroende på datatypen. För vetenskapliga artiklar presterar algoritmen väl, givet att en tillräcklig mängd förklassificerade dokument tillhandahålls. För patent presterar algoritmen väl, även om klassificeringsalgoritmen tränas med vetenskapliga artiklar. Prestandan ökar med förhållandet mellan hur många förklassificerade dokument och oklassificerade dokument som används. Valet av antalet neuroner i det gömda lagret av det artificiella neurala nätverket påverkar inte prestandan signifikant, däremot gör antalet korrigeringsiterationer det. När valet måste göras mellan ett högt antal neuroner och ett högt antal iterationer visar resultaten att ett högt antal iterationer är att föredra givet en viss mängd beräkningsarbete. Slutgiltligen så visar uppsatsen och resultaten att metoden kan användas för att analysera företags trender i publiceringsbeteende. Förändringar i dessa över tid kan också utvärderas med hjälp av den föreslagna metoden.

Acknowledgements

The author would first like to thank SKF and Alejandro Sanz for the support and sponsoring in creating this piece of work. Furthermore this thesis would never have been possible without the advice and support of Ingo Schulz. The author would also like to thank his friends and girlfriend for loving support throughout this work, without whom this thesis never would have achieved its full potential. Finally I would like to thank my mother for her support during the period of my studies, for her advice and of course for her wonderful cooking.

List of abbreviations

ANN	Artificial Neural Network
B.V.	Besloten Vennootschap (Limited Liability)
BI	Business Intelligence
BOW	Bag-Of-Words
BPNN	Back-propagation Neural Network
ECLA	European Classification
EPO	European Patent Office
INPADOC	INternational PAtent DOcumentation Center
IP	Intellectual Property
IPC	International Patent Classification
IT	Information Technology
k-NN	k-Nearest Neighbours
LDA	Latent Dirichlet Allocation
LSA	Latent Semantic Analysis
LSI	Latent Semantic Indexing
ML	Machine Learning
NN	Neural Network
R& D	Research & Development
SI	Strategic Intents
SKF	AB SKF (Svenska Kullagerfabriken)
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TD	Term-Document
TDA	Thomson Data Analyzer
TDM	Term-Document Matrix
TDV	Term-Document Vector
tf-idf	term-frequency inverse-document-frequency
TI	Technology Intelligence
TIT	Technology Intelligence Team
TT	Technology Tree
USPTO	United States Patent Office
VSM	Vector Space Modelling
WIPO	World Intellectual Property Organization

Contents

List of Figures

List of Tables

1 Introduction

In the society of today information is becoming available at an ever increasing rate. This immense sea of information can be a hard for organizations to navigate. Technological information is also being produced at an increasing rate, leaving many organizations behind, baffled by the management of this vast amount of data. At the same time studies show that *research and development* (R&D) managers in many companies recognize the importance to be working with the state-of-the art in their respective technologies [Por05]. In this section and in the subsequent paragraphs we will describe the background of the research and what was to be done in the context of the project at *Svenska Kullagerfabriken AB* (SKF). Finally we will present the hypotheses and the scope of the thesis.
As companies are adapting to this "new era" of information overload, the importance of keeping a clear focus and leveraging this information is becoming increasingly important. One aspect of this management of information is keeping information stored, and accessible, based on keywords or tags which is already used extensively in online information [Mic11]. Other areas where these types of data classification schemes are important is in database management and web-search engines. This type of indexing purposefully enables the users needing the information to search very large amounts of data in a faster way. Thus indexing serves two purposes: Primarily to define the structures of information, i.e. how to organize information, and secondly, it makes information more accessible.

The area of automated document classification has been a growing area of research over the last decades [Seb02] mainly due to the increasing amount of data available in digital form. This development coupled with the possibility of analyzing vast amounts of data using personal computers has enabled a new way of looking at data and managing it. Since the 1980's the processing power in a standard PC has multiplied following Moore's law, which describes how the computing power (number of transistors on a single chip) has doubled every second year during the last 40 years [Kan05]. This strong increase in computing power is also now available on a laptop and can be carried with you, anywhere you go.
The need for an automated approach to categorizing data is an important topic for companies that are involved in many different technological areas. The fact is that the number of sources of information on almost every topic is increasing this is no exception for SKF. SKF has 5 main platforms, 3 divisions and approximately 45000 employees globally and the management of information is already a true challenge within the company. More interestingly the importance of categorizing and filtering data becomes essential to support the organization in dealing with the pressure of making strategic decisions.
This is a great opportunity and an immense challenge for SKF and thus this area has become a point of focus, more specifically within the area of business and technology intelligence. *Business intelligence* (BI) is defined as actionable knowledge regarding markets, competitors, customers, and the company itself. In contrast *technology intelligence* (TI) could be seen as the technological complement of BI in the domain of technological knowledge. This knowledge usually presents itself in knowledge made available internally at a company or publicly by means of freely available sources.
The thesis thus relates to the *technology intelligence teams* (TIT) approach to data management. Specifically the thesis will focus on an automatic algorithm for allocating different sources of information according to a predefined set of classes known as *strategic intents* (SI). These SI are intimately connected to the main strategic areas of focus for SKF. An important topic for the TI function are trend analyses of the different SI. A central aspect in this analysis is the temporal development. Which areas have been expanding or contracting for SKF or a specific competitor over time?

Earlier research within this topic has been done with regards to allocations according to specific predefined ontologies. Studies exploring different clustering algorithms have been done, see for instance Kumar [PNT05], Raghavan et al. [CDM08] and Sebastiani [Seb02]. Furthermore the mathematics behind the clustering algorithms has developed and is available in many computational software packages. In the case they are not available directly, they are readily downloadable from sources such as Netlib [Net]. Literature is widely available within the different topics. The focus of this thesis work has, however, not been done before since the use of the SI is a new concept at SKF.

The project is not only an academic endeavor, but also has a strong interface towards the strategic management of technologies at SKF. If the mapping of the SI is successful this can be used to support the monitoring, classification and evaluation of competitors, potential acquisition targets or possible licensing partners for SKF. Furthermore the strategic aspects are essentially the main objective of producing TI. The goal of which is to create actionable knowledge, regarding the technologies that are important for SKF.

1.1 Hypotheses

The hypotheses will be the topics to be validated by the thesis work. The hypotheses in this thesis are of a mixed nature. This has two reasons, the first is the fact that the research is cross-functional and thus the results are subject to some discrepancies due to the non-analytical answers given. The second is that the research is focused on providing qualitative answers on the feasibility of the approach.

Hypo. 1) It is possible to *automatically* create a document classifier for new documents based on previously (manually) classified documents

Hypo. 2) An automatic allocation algorithm can perform at 95% accuracy in a binary setting for classifying documents according to the strategic intents defined by SKF

Hypo. 3) A one-against-all approach with binary classification performs better than a multiclass classification

Hypo. 4) Using an insufficient amount of documents decreases the performance of the allocation algorithm

Hypo. 5) Using a sufficient amount of documents saturates major improvements in the performance of the allocation algorithm

Hypo. 6) Performance is strongly influenced by the number of iterations used in the learning of the allocation algorithm

Hypo. 7) Using automated allocation, inferences can be made on the direction of competitor research and patenting activity

1.2 Limitations & scope of the thesis

The thesis will focus on the allocation mechanisms for documents. It will not explicitly deal with *information technology* (IT) solutions or the aspect of implementing a finished information management system in the company. This is already in place and any modifications should be left to more suitable studies. This thesis will focus on answering the fundamental questions of if data can be organized automatically according to the SI and if it can, how well? Furthermore different models for classifying data into clusters will be studied. These algorithms will not be evaluated using all possible feature allocation concepts but will only evaluate the semantic relatedness of documents.

The study will only focus on allocating publications and patents. The study will not give advice for in what computational system these allocations should be done, but recommendations for the algorithm based on the results will be made. The study will also not address computational aspects such as scaling of the algorithms. This is because the scale of the implementation will be for a fairly limited number of documents in the foreseeable future and in the case this becomes an issue it may be evaluated at that time. In the case that a larger scale system is to be implemented, this study shows what underlying framework could be used and from this basic framework different parallelizable aspects could and should be addressed if necessary.

The study will only discuss the strategic aspects of the management of technology. This is a natural consequence since this thesis is focused on the automated allocation algorithm and not the broader TI framework within which it will be implemented.
Thus the three main topics of the thesis is:

- What sources to use, and how to and which raw information to extract from them
- When we have this data, how do we analyze it and what methods can be used to evaluate this data
- When this data has been evaluated we have classified the documents according to the given SI framework, how can we evaluate the performance of the algorithm

These three main points are the scope of this thesis work. In the next section we will address the planning and execution of the project, specifically what areas were covered and how.

2 Methodology

In this section we will give a brief description of how the project was structured and what aspects were considered during the research. Here the focus is on the process behind the results and we will describe each in turn. Starting from the background research, then discussing what data was to be analyzed and finally how this data should be processed. Furthermore we also motivate the choice of analysis software and the chosen sources of information. Initially we present the project plan and the structure for the different parts of the project.

2.1 Project outline

The project took place during the spring of 2011 at SKF B.V. in the Netherlands. The purpose was to develop an algorithm for automatically classifying documents into a taxonomy based on a predefined set of classes. The project thus naturally started with an evaluation of general frameworks for TI. What frameworks exist and which are used at SKF. The purpose was also to define where in the TI process this project fitted in. After this the evaluation of which sources of information to use was done. This concerned which sources that were appropriate and which should be used when gathering information on technologies. The next step was to evaluate which methods that could be used for classifying this data, based on the source they came from and what types of data were available. A smaller pilot case was done to evaluate the suggested framework. After this the evaluation of the pilot results was done to support the approach. Finally the approach was validated against a proper business case at the company and applied to evaluate the performance of the suggested algorithm.

2.2 Literature research

When studying a topic from many different perspectives, the need for a proper literature research in each respective domain is needed. The scope of this study as outlined in the section 1.2 on limitations and scope, give the focus area to be analyzed. The study focuses mainly on what sources of information are to be analyzed, and how this information can be analyzed.

There are in essence five areas that need to be addressed by the literature study. Firstly the focus is on the conceptual frameworks that exist for TI and how this can be done, to give the context of the thesis work. Secondly the area of automatic document management, what types of text can be analyzed, what can be deduced from such categorizations. Given the sources of data available to us, which part of the data is most important to analyze? Thirdly given this data and the sources we have, what algorithms and which measures of similarity between documents and information can be used, how can different metrics be calculated to ascertain how close a document is to a predefined group? Fourthly what results will we get from this, what types of measures are relevant to consider and how do we interpret the results? The fifth and last area of focus is the management and the strategic aspect of the project, what context the project is used within and how it adds to a larger framework for managing strategic information.

With regards to the handling of vast amounts of data and the scalability of the approach we will not mainly focus on these areas. The topic of scalability does not only encompass the aspect of how to save and store data, but also considers the usage of multi-node computer networks. These topics were left for possible follow-up projects, should this pilot prove successful. This and further research into the topic are to be considered in later iterations of developing this framework at SKF.
Another aspect that does not need to be covered is the classification scheme, since this information is available within the company. Also the strategic recommendations are not of primary concern within the scope of the thesis, due to the fact that each requesting party will differ in how this intelligence and the results will be used. This is a part of the TI function and will be discussed together with the team, but research into this area is not a focus for this thesis.

An interesting aspect for this thesis are the patent and publication code systems provided by different information suppliers. Some of these are INSPEC, Compendex, IPC and other classification codes provided by publication databases and patent authorities, for the interested reader they are presented in appendix D on classification schemes. These are different paradigms of arranging information to a predefined set of groups usually, but not always, built on a hierarchical framework. This framework is normally organized as a class$\rightarrow$sub-class$\rightarrow$group$\rightarrow$sub-group paradigm. A review of the underlying concepts of these measures is not only important for use as a source of information to be evaluated in the thesis, but will also serve as an inspiration to how these paradigms deal with the classification of data. These classification systems are well thought out and similar approaches are being pursued within the company, which may be leveraged in the thesis work. This is a science in its own right and thus it is not an area of focus for this thesis.

For each of the aforementioned areas of interest a literature research has been done, as well as searches concerning surrounding themes. Therefore the literature research runs in parallel to the full project, whereas each researched area will be investigated more deeply during the corresponding stage of the project.

2.3 Developing the theory

For developing the methodology many different concepts were investigated and reviewed. In principle the purpose was to automatically assign documents to predefined groups based on their contents. Therefore the most important aspects are what parts of the data that will be analyzed, and how this data can be processed. After preprocessing this data we need an algorithm to categorize the information sources based on the results. We also need to consider both the computational and linguistic problems that may occur when using this form of data, see for instance Raghavan [CDM08] for a discussion on this topic.

An aspect that needs to be investigated is how classes are defined and how documents can be allocated. We need to evaluate the different approaches for describing documents and what features of the documents to analyze. This can then be used to measure the similarity between words and other elements of the documents. Another measure of similarity that may be interesting to analyze is the different classification codes for patents and publications. There are no obvious translations between different classification systems for patent or publication documents. The approach for classifying the documents and the framework for doing this is a non-trivial part of the project. In this part of the project several resources will be leveraged within the organization, especially with regards to a structured approach of organizing knowledge and information systematically.

The interpretation of the findings also needs strong critical evaluation. This will be based on different subjective and quantitative assessments, partly by the researcher but also by domain experts within the organization. This is usually referred to in the literature as an ”expert validation” of the approach. The approach will need revision so primarily a first test round and interpretation of the results will be done to check the performance of the algorithm. Subsequent iterations will then improve upon the approach, before the final framework is handed over.

2.3.1 Using a cross-functional approach

Using an approach that ranges across many areas of expertise will, of course, not be exact in every domain of knowledge. The author has limited knowledge of each of the areas, therefore there may occur discrepansies between the terms used in this thesis and those that are norm in each specific field. The need for mixing sources of information and gathering the information based on SI is powerful since it enables an ascended perspective and a common mode of comparison across companies. This also presents a challenge to the research since it combines many different areas of focus.

2.3.2 Modular approach

One of the more important aspects of an automated process in general and this case in particular is a modular approach. The suggested algorithm should not only be used for automatically classifying documents for this specific business case and request. The algorithm should be able to analyze any textual document source for any given case. This entails modularizing the framework, clearly defining what areas that are addressed and in turn how each part supports the subsequent analysis. This is crucial in creating a flexible approach that will be useful in the future. Of course some parts will have to be case specific, but the general algorithm and parts of the process should be as interchangeable as possible to support re-usability in future implementations.
In principle this comes down to allowing different sources of input data, choosing a flexible algorithm for analyzing any type of data and finally generating similar outputs independent of the types of inputs.
This also is a basic challenge with a project for an organization of this size. Thus the focus will primarily be on the needs for the specific project within the TIT. This can then be extended and augmented as often as needed to allow for other implementations to leverage this work in future applications of the suggested algorithm. This will be addressed further in the theoretical section 4 when describing the algorithm.

2.4 Data

The data that will be considered according to the scope described in the introduction section are publications and patents. Data will be gathered from Thomson Innovation and other company internally available databases of information. Furthermore company internal data on classifications and paradigms for technology taxonomies will be used. Data will be researched based on a primary test case, and for the final business case it will be provided based on relevant research done by the TIT.

2.5 Choice of computing packages & analysis software

The choice of different analysis tools is highly dependent on the needs of the application. For this application a high accuracy computational program was needed. This since the grouping of documents and similarity measures involved accurate and computationally heavy calculations. Furthermore the program should be easy to use and have a simple way of both importing and exporting data to and from the software.

With regards to the general structuring of data, for instance data cleaning or ordering, the need is for a software that to a large extent cleans and organizes data automatically, with minimal input needed from the user. In this case it is also important with the interfacing, i.e. how easy it is to import and export files to and from the software.
Furthermore an aspect that is important to consider is the visualization of the results. For instance a typical categorization problem is easy to visualize. This means that the results not only should be grouped to the different classifications but for comprehension they should also be visualized in some way, to convey the results of the classifications. Here the inspiration may come from progressive pursuits such as ThemeScape from Thomson Innovation [Tho11a] or GapMinder [Gap].

2.5.1 Matlab

In the case of this thesis the mathematical software used is Matlab [The11b]. This had a number of reasons, but it was mainly due to the familiarity of the researcher with this software. The software is also easy to import data into, well structured, stable and can manage very large amounts of data. It is also fairly efficient built on a C++ core that is fast and efficient on many systems.
Matlab also has many algorithms efficiently implemented in the software. This is possible due to the built in "toolboxes" that supply preprogrammed functions within a wide area of applications. By using already implemented solutions we assure the efficiency of the algorithms considered as well as the efficiency of the computations.

2.5.2 Thomson Data Analyzer & ThemeScape

In the case of text analysis software when using Thomson's databases as a source the natural choice is *Thomson Data Analyzer* (TDA), their own proprietary analysis software. This software gives the user the possibility to automatically import Thomson Innovation records and provides powerful statistical analysis tools. Specifically for this thesis it has the possibilities of performing natural language processing on the data upon importing.
It also provides the possibility to merge data sources or different data files for more comprehensive analysis. This gives some aspects for pre-managing the data before the analysis starts. There are built-in functions for data cleaning, and for using thesauri to clean up company names, inventors lists and other elements of the raw data.

For visualizing the documents found we should strive to give a comprehensive yet in some sense complete picture of what focus areas the documents have. Thomson has a software for supporting thematic mapping of documents based on their thematic contents. Once documents are grouped into SI these can be appended as complementary information to the documents in Thomson Innovation (the document source). Thus when documents have been classified this information can be used for analyzing documents in Thomson Innovation more elaborately using the technology landscaping software ThemeScape. For more information on these solutions we refer the reader to the sources available from Thomson Innovation [Tho11a].

2.6 Company specific solution

At SKF the TIT already has processes for information flow, request management and assignment prioritization. They also use a systematic way of dealing with new document sources and structuring the information connected to one request, thus the framework for how to deal with storing and managing of data was already in place at the company. It was thus important that the suggested framework was fitted into already existing processes used by the team. To enable this the suggested process was developed in collaboration with the TIT at SKF.

With regards to the software used as an intermediary and a data gathering and formatting tool Excel is very valuable. The solution should strive to be implementable in Excel, since this software is widely available within the company. Another software that is used within the company is Minitab, a statistical software package. This software should be used to implement the statistically heavy calculations once the algorithm is finished and handed over to SKF. This to make the final solution as accessible as possible for the end users, which is why this will be attempted once the first pilot has been completed. This will be a priority during the end of the project at SKF, after an evaluation of the requirements of the software implementation. The thesis work is more suitably seen as a proof-of-concept project to validate the approach. The work is done to establish if the methodology may be useful in the future for the TIT at SKF.

We now move on to describing the underlying theory for the thesis and describing how the proposed framework will function. The request from SKF was that the developed solution for the company was to be structured as a process. This was needed so that the framework could accept different sources of information and based on the knowledge of the area the system should categorize the documents as well as possible. Thus the output of the framework was to assign each document the most appropriate label based on the underlying predefined classes that was made available from the company, and to map the development of these over time for each to generate basis for an analysis company by company.

3 Background Theory

In this section we will give an introduction to the various theoretical aspects that the reader should be acquainted with. To comprehend the contents of chapter 5 where the framework is described the reader is encouraged to focus on the topics where they have the least knowledge so far and focus less on familiar topics. The subsections are arranged to first explain the background to TI, and specifically TI at SKF. The following section deals with the data and the information sources that will be analyzed. After this we will turn to what algorithms and methods may be used for analyzing this data. Following this we will turn to the results and what performance measures that may be used to evaluate the algorithm. Finally we propose the developed framework and method which was the aim of the thesis.

3.1 Technology intelligence

The area of TI has been developing steadily during the last decades, and is increasingly becoming a very important topic for any organization dealing with technology [Seb02]. This can be compared to "keeping your ears open" and being observant of major changes in the industry where the firm is operating. Earlier many of these possible sources of information were analyzed by sales people or similar functions that observed changes in the market environment. These functions then disseminated this information formally or informally within the organization [Por05].

In the current business climate with globalization providing companies new markets but also new competition there is a need to gather data and information in a more systematic way. This is one important aspect of TI. To generate and provide decision makers within companies with the necessary information for making timely decisions based on relevant and specific data [SKF]. Therefore one of the main needs in TI is to provide information quickly and accurately. From this perspective an algorithm that can extract the topical nature of a large amount of documents is a valuable tool. This becomes important partly because the amount of information generated is too large to handle efficiently. It also is important because the amount of information that is generated needs to be allocated into meaningful repositories where the documents are indexed efficiently. Such structuring of information facilitates the further analysis using these sources of information. It also enables specialized engineers and analysts to focus on their core area, instead of having a full team dealing with all different types of information in all different topics at the same time. This is especially important when the company is of a considerable size and functions vary across technological fields. For SKF and the TI function in the company this is a true challenge, but also a major opportunity due to the vast amount of information available and the number of competitors that SKF has across its many platforms [SKF].

Another aspect of TI is inherent in the phrasing of the term, intelligence. Intelligence is not simply words or statistics produced automatically by a computer, rather we define intelligence as actionable knowledge. This is usually described as non-obvious and non-trivial connections of more than one element of data [Por05]. This also connects the raw data with a level of refinement needed to make decisions based on that information. The current number of patent applications by a specific competitor in one of SKF's busniess areas may or may not be relevant. If instead one can describe the trend in the data, for instance the change of focus for the competitor from another business area to the area that is important for SKF this information supports in the strategic management of the current situation. Thus the interconnection between TI, BI and strategic management is an important aspect of TI. The fact that the knowledge should be actionable and well formulated creates the need for evaluations of what the requesting party needs. To ensure that the resulting intelligence products are up to date it is necessary to have a clear view of what information is needed and what information can be left out.

What is important for SKF is that the right people get the right information, at the right time. This is, and should be, the goal of all TI activities. From the perspective of SKF this entails providing the requesting party with the information that they are in need of in a timely fashion. Besides this the company internal TIT focuses on analyzing the core changes in the business climate of SKF. This is mainly done to assess changes in the market and the positioning of competitors in terms of product offerings and intellectual property (mainly patent) portfolios. Therefore the focus of the TI activities are around publications, patents and similar sources of public information that competitors and their affiliations produce.

3.2 Theory of technology intelligence

In this section we will investigate the different theories for TI. There are many different paradigms and methodologies for doing TI. In this section the focus will be on what frameworks are available for doing TI. Specifically the topic of focus will be on algorithmic approaches in TI, with a bias towards mathematical methods for classifying data.
When studying mathematical approaches to automatically classify information, machine learning is the topic of interest. These topics include the usage of supervised or unsupervised learners that are trained to classify data. The classifiers can be trained using prelabeled training sets, which is called supervised learning. Classifiers can also use induced learning, a rule based learning where the clustering of data decides on how to classify data based on the experience of the algorithm, this is known as unsupervised learning. These will be described below in chapter 4 as well as other approaches for automatically dealing with TI.

There are a multitude of frameworks that can be used for TI and this has been thoroughly studied in the literature, see for instance Arman [AF10] Diaz-Prado [JADP10] and Porter et al. [Por05]. The following section, gives a brief outline of the main points that are relevant for the thesis work.

3.2.1 A framework for technology intelligence

A general framework for dealing with TI can be found in the book by Porter and Cunningham [Por05]. In their book "Tech Mining" they describe the technology mining process. In the context of this thesis our definition of TI, and what Porter and Cunningham call tech mining is the same. Tech mining is therefore concerned with providing the right people with actionable knowledge at the right time. The framework that Porter and Cunningham suggests is based on a nine-step process, which is outlined in table 3.2.1.

The nine steps in the tech mining process
1. Issue identification
2. Selection of information sources
3. Search refinement and data retrieval
4. Data cleaning
5. Basic analyses
6. Advanced analyses
7. Representation
8. Interpretation
9. Utilization

Table 3.2.1: The nine-step tech mining process as outlined in the book by Porter and Cunningham. Source: Porter and Cunningham [Por05].

The general steps describe the standard flow for any TI activity. The first step is to identify an issue, or a topic of analysis, to be able to limit the scope to an appropriate amount of data. To further limit this the second step entails the choice of which databases and other sources of information to source for data. The third step is usually conducted in cooperation with the requester when for instance a search query is to be created. Once the query has been created and the search has been narrowed down the information is downloaded or gathered in some manner.
The follow up and fourth step is the first cleaning of the raw data. In textual data this may concern outliers such as misspelled words, possible errors when parsing the data, and a general structuring of information. This indexing serves the basic purpose of organizing the information and the next step provides some basic statistics based on the collected data. This may for instance entail how many words per document there is, or how many publications and patents a certain company has produced. All the raw measures are then refined in step six, where a more advanced analysis of the data is done. This can entail looking at researcher networks and affiliation analysis, or could for instance cover the citation network of patents. In these more advanced analyses the temporal aspect also becomes apparent, looking into trends over time in the data.
The seventh step concerns the representation of the results, or simply presenting the findings in a meaningful manner. Usually this concerns itself with the topic of presenting key mesurables consistently and having a common format for generating the TI reports. A typical example of this is also given in the form of a "one-page" report as presented in the book by Porter and Cunningham [Por05], see figure 3.2.1.

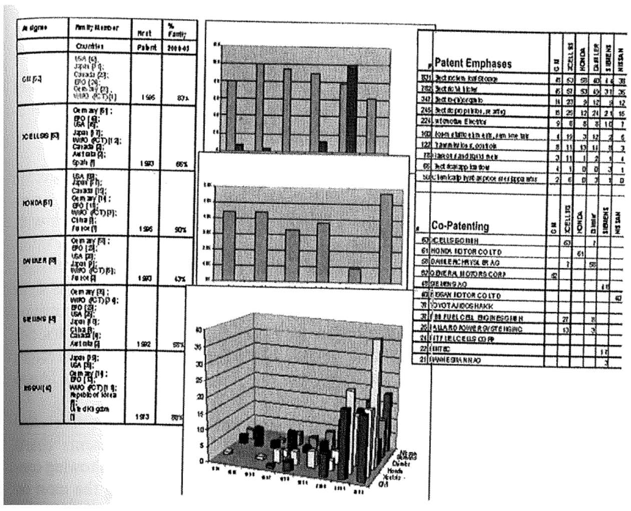


Figure 3.2.1: *A shorter example of a one-pager, from the book by Porter and Cunningham. As can be seen patent emphases and co-patenting activity are some topics that may be addressed. Source: Porter and Cunningham [Por05]*

The eight step concerns the interpretation of the findings and this is when the requesting party, together with the TI provider gather the information and try to make sense of the findings. This step usually is the important, but also final step for the team, whereas the ninth step is the implementation of the findings as basis for decision-making on the part of the requesting party. We will now turn to describing the TI process at SKF.

3.2.2 Frameworks for technology intelligence in SKF

The TI department at SKF primarily deals with specialized in-house requests from upper management. These requests are prioritized based on different allocation rules. Among these allocation priorities are the level in the organization from which the request is coming from, what type of request it is and also the urgency of the request. A specific approach that is being investigated is to restructure the team as an internal consulting department dealing with the semi-joint areas of IP, TI and competitor analysis [SKF]. The TI department functions as a separate department that is disjoint from the division based structure within the company. This facilitates that the team can work on different topics, but also limits the level of technological details that the team can know within any specific area.
The TIT is set up with team members speaking many different languages covering all main regions of competition for SKF. Currently the TIT working with this are based in different regions globally and work with different areas of the framework. Within the TI function the following structural framework exists for dealing with TI, see figure 3.2.2.

Different topics that are analyzed are, among others, business news, scientific publications, conference papers, proceedings and patent data. Different sources come into the framework at different stages of the analysis. All sources are not used for all evaluations. For instance a freedom-to-operate search is not improved specifically when business news is included and vice versa publications do not give any financial business specific information.
Some sub-areas of this framework are the focus of this thesis. Another important topic is the temporal descriptions of the evolution in the different areas for instance how patenting/publication activity increases decreases over time. These different sub-topics focus on smaller segments of the different sources of information or knowledge that are being used within the company. The schematic in figure 3.2.2 provides the general framework. The specific focus areas of this thesis are on the topics of SI and the interface between patents, publications and other sources of information and their temporal aspect, at the right end of the schematic. This is then used to create indicators that will give basis for different trend analyses within the TIT [SKF].

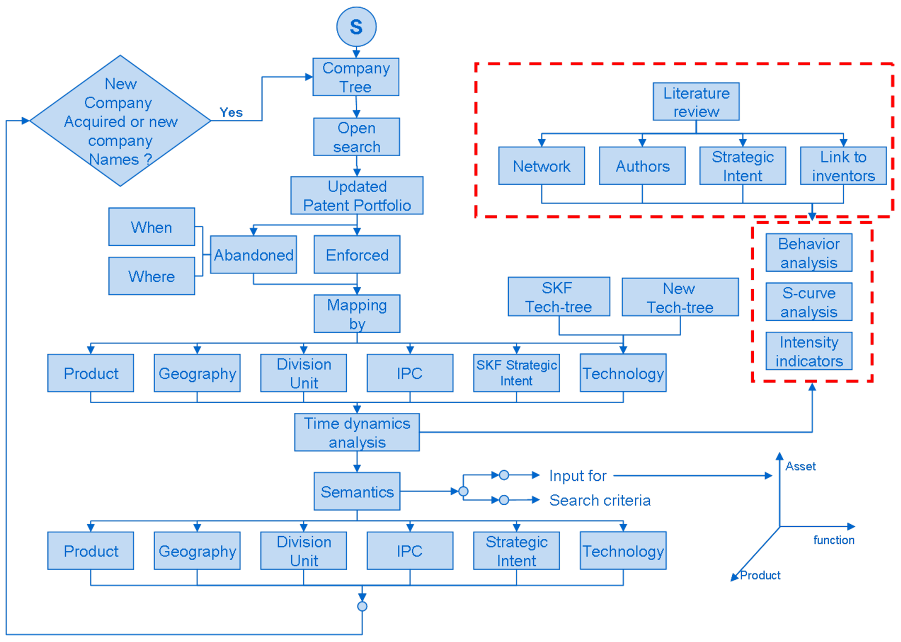


Figure 3.2.2: *SKF's internal structure for dealing with TI for a specific competitor [SKF]. The focus of the thesis work is on the right-most dash-marked areas of the scheme. Source: Sanz [SKF]*

For analyzing the TI process at SKF we can use Porters nine-step process, outlined in table 3.2.1. When studying the nine-step process we see that it denotes a very general framework. To narrow this down to what is relevant in our case we give examples of how this process works. The TI process at SKF can be described using these nine steps. Usually a request is placed for a TI briefing on a specific or competitor (step 1). Then the TIT uses different sources of information for sourcing data (step 2). The team then uses a feedback system with the requesting party, to refine the search query and search area (step 3). After the data has been collected the TIT cleans and structures the data (step 4). Standard analyses are then completed based on what companies are the source of the information and data, who is publishing and where (step 5). Then a more comprehensive analysis is done to generate measurables and company comparisons (step 6). After the advanced analysis, the representation of results become an important part of the analysis. This data is then used to present the results in a systematic manner when all results have been made available (step 7). The interpretation of results is then partially based on the observations of the technology team, and further analyzed together with domain experts to assess the relevance of the findings (step 8). In the final step, the requesting party makes use of this information as one part of supporting info in the decision making process (step 9).

3.2.3 Knowledge allocation paradigms at SKF

When organizing knowledge there are many possible approaches that may be used. Within SKF and specifically within the TIT there are two principle allocation paradigms that are used. One of these are SI, which can be thought of as a form of higher level ”topics” that are interesting for SKF. The other is the allocation according to a *technology tree* (TT), which goes into detail and tries to map out any given area of analysis (for instance for one technology) into a tree based structure. The purpose of the tree together with the branches is to go as deep as needed to have topics that are perfectly disjoint when allocating each piece of information. We here strive to create a mutually exclusive and collectively exhaustive set of branches and sub-branches. Thus these two different methods serve two different purposes. The allocation to SI gives a non-disjoint set of classes according to which the information can be organized. The allocation according to a TT is appropriate when one wishes to create a perfectly disjoint set of classes of data. One can also draw an analogy with the situation when one wishes to have an overview (SI) or when one wishes to allocate specific knowledge regarding a given technology (TT).

Strategic intents at SKF

To structure the different strategically important areas for the firm, SKF has initiated the development of a number of Strategic Intents. These cover core topics of technology that are focus areas for SKF's business as well as long term areas of interest for the firm. The SI are essential and describe the perspective used when generating the list. These serve as a basis for discussing decisions within strategically important areas for SKF and the SI are also to be used for classifying documents and information internally.

There are 18 different SI, 10 for products and 8 for processes. These together define the 18 strategic areas that are the most important to SKF. The 18 SI are as follows, A)-J) are product intents and K)-R) are process intents (censored for the purposes of this thesis, definitions available at SKF [SKF]):

A) -

B) -

C) -

D) -

E) -

F) -

G) -

H) -

I) -

J) -

K) -

L) -

M) -

N) -

O) -

P) -

Q) -

R) -

An important observation is that these constitute a very wide area of knowledge and are not necessarily disjoint classes of knowledge. These topics are not perfectly separable, and this will make the analysis of the topic more complex. This could be compared with the case of separating two things. In some cases this distinction is very simple, but how do you distinguish a plastic apple from a real apple, using only what can be observed externally. This is one of the main problems of the thesis: the knowledge of how to separate documents that each belong to one of these SI. For the purposes of the thesis we will only use the first 10 intents A) through J). These intents cover the product SI, and are the only ones that will be used in the context of this thesis [SKF].

Technology trees at SKF

One method used within both the intellectual asset management and the TIT at SKF is the so called Technology Tree. A TT is a conceptual arrangement in the form of a tree of information/assets/knowledge/data so that each main branch of the tree covers a certain topic exhaustively. The tree is usually first derived and structured using knowledgeable experts in the area, it is then reviewed until these reach a consensus on the structure. When this has been done the tree is thus a structure for arranging information, or knowledge regarding the specific area.

In the case of intellectual asset management this comprises mapping intellectual assets to a given tree structure, using a hierarchical arrangement. This means that each branch is comprised of sub-branches that all are sub-sets of the super-branch and also that the super-branch is comprised of the sum of all its sub-branches. Therefore the tree structure gives a disjoint description of the whole set of topics that may be included within that technology area [SKF].

When categorizing a document within the tree the priority is to first find the main (highest) branch appropriate to file the document/asset/ knowledge/data under. After this one should try to map it as deep within the tree structure as possible, adding more sub-branches if/when it is necessary. Thus the TT or intellectual asset tree becomes an ideal structure for dealing with information and arranging it. This is also a powerful construct when we wish to assign documents to the different nodes, since the classification structure strongly resembles a computational allocation of the information.

The TT is a different paradigm than the SI for arranging information or similar knowledge. The SI provide a thematic level view of a topic within SKF's areas of interest. In comparison the TT are used to specifically define knowledge concerning one technology area, which may or may not be, part of a SI [SKF].
The TT fills the function of giving a more hands on, low-level structure for arranging information. It is therefore used when technology specific cases are initalized in the TIT. The main distinction between the TT and the SI is that the tree structure creates a disjoint space which can then be populated with data. Now that we have discussed the different frameworks that exist within the TIT, we move on to what we define as knowledge in the context of this thesis.

3.2.4 What is knowledge?

In this thesis we wish to describe the developments within certain technology areas and strategic areas that are important for SKFs business. To analyze this we already limit ourselves from all possible sources of information, only focusing on publicly available knowledge that has scientific or business value. Thus we only take into consideration a limited set of all available data. We cannot for instance gain access to company internal data from the competitors, or assess the distribution of that content. The focus of this thesis is thus on *publicly available* sources of information such as patents and publications. This is the main way of generating information in the TIT at SKF as described in the section above and the main database for gathering information is Thomson Innovation and the related solutions provided by Thomson Reuters [Tho11a] [SKF].

It is also important to consider the relational aspect of data and information. For instance when studying knowledge we will never have perfectly disjoint areas of knowledge when considering any specific technological domain. Usually there is a part of the knowledge that is common, and then specific parts are intertwined in complex structures. Since the Strategic Intents described above do not represent perfectly disjoint groups of knowledge, for instance a full product that measures and predicts the life-time of a ball bearing could belong to intent A), B) and G) simultaneously we need to structure this information. Thus the following framework for separating knowledge was adopted from Porter and Cunningham [Por05]:

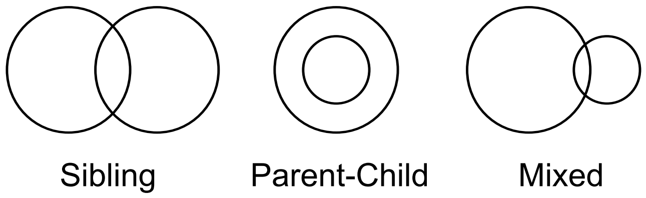


Figure 3.2.3: *Knowledge relationships between different sources. Source: Porter and Cunningham [Por05]*

This can also be described using set theory, and will be used in the analysis and classification of documents. This framework gives a clear hierarchy of where information may be placed in relation to each other and will be adopted for describing the knowledge and the sets that we wish to classify the documents into. In the analysis we will also consider how to assign documents to classes, which may be done in many ways. Some examples are: as distributions over the SI, as a binary allocation to one of them in turn or as a multinomial allocation with multiple allocations for each document.

Tacit vs. explicit knowledge

There is a need to distinguish between the two basic types knowledge, tacit and explicit. These two areas of knowledge are characterized by different characteristics, as described by Nonaka [Non91], [Non94]:

Explicit knowledge is

- formalized and systematic
- easy to understand
- simple to communicate

Tacit knowledge is

- highly individual
- difficult to structure and formalize
- hard to communicate

Thus the topics that are covered by the documented information in publications, business news and patents, are only explicit knowledge. We do not cover the thoughts behind the publications, and we do not cover the knowledge generated through the experiments or experiences in creating this knowledge. This is a problem when we wish to model the interest in, or knowledge of, a specific area [Lic03].

The situation becomes even harder when we consider that not all information is public. Estimates indicate that the "deep web" contains approximately five times more information than the public Internet. This "deep web" consists of information that is accessible only through some proprietary sources or databases. Another issue with accessibility of data is that much of the knowledge generated in a company is never disseminated outside of the company, or institution that generated this knowledge. Thus accessibility of knowledge and the fact that all knowledge cannot be explicitly stated are the main limiting factors when trying to model knowledge [Lic07]. These are the main limitations with the data we are using for this analysis.

Bearing this in mind, the other perspective is that whatever is not accessible to one organization is usually not available to most other organizations. This lessens the importance of this factor in the competitiveness of SKFs TI function. For instance a competitor may have knowledge in some areas, which are secret to SKF, and on the contrary SKF may have knowledge in areas that are secret to the competition. Therefore there exists a competitive balance between the two, and the main focus of analysis should be on data that is possible to evaluate, i.e. publicly available, explicit or semi-explicit knowledge. To exemplify this one can use the iceberg concept, see figure 3.3.1, replacing the *external/internal* with *available/not available* information emphasizing that only a subset of all possible information is available for analysis purposes at any given time.

3.3 Data

The data that was to be analyzed in the automatic classification were patents and publications. These two have different characteristics but simplifying we may consider table 3.3.1 as a description of these.

Type of Source	Publications	Patents
Full-text	✓	✓
Structured fields	(✓)	✓
Classification codes	(✓)	✓

Table 3.3.1: What fields are available in the automatically downloaded data. () indicate that some data may contain this information or contains it only partially.

For the analysis of these sources it is important to consider some points. Firstly what can be analyzed, and secondly if adding this information adds to the quality of the results. This was one of the aspects of the study that needed to be considered. Also the degree of success in categorizing data from the different sources is an important aspect to evaluate the performance given the data type.
Yet another aspect that is important to take into consideration is that different sources of information have different features that may be analyzed. Patents, as mentioned above, have many classification systems, some of which are the International Patent Classification (IPC), the F-code system in Japan or the US-classification. The interested reader is referred to appendix D for a description of these classification schemes. Besides this some firms supply additional information to patents by reading through these documents and classifying them according to their own paradigm, for instance Thomson Reuters provide what is called the Derwent system with supplemental information for patent documents.
Thomson Reuters provides this through their service Thomson Innovation. Publications on the other hand are not classified in the same way, and generally not by the same people as the patents. This discrepancy, and the limited specificity of the different indexing schemes such as Inspec and Compendex codes present a limitation to how well-categorized publications are. For a description of the Inspec and Compendex classifications, see appendix D.2 In turn we can consider patents as the most well-classified, publications generally somewhere in between and finally business news as the least specific source of information. This is of course a generalization, but given the information sources we have considered in this thesis, this has also been the case in the analyzed data.

For SKF this translation into the company internal framework of SI is crucial. This helps in communicating across different functions in the company and also helps the TIT members to each deal with individual areas. For instance some team members in the TIT may have a background in electronics and are thus more suited for analyzing electronics papers and patents. These analysts usually have an in-depth knowledge of specific areas and understanding of the terminology associated with those areas, facilitating future analysis of documents within those areas for SKF once the documents have been classified.

3.3.1 Choice of information sources

The reason for using the specified sources of data was a weighted decision. Firstly the aim of the current approach of the TIT is an approach best described by the ”ice-berg” concept meaning that we will only consider the ”visible” part for analysis. Secondly all the information that is physically available anywhere would be preferable, but given this situation we wish to measure as much information as possible. There is an important distinction here between what could be known if there are no limitations to accessing all data available anywhere, and what can be known given the openly available sources for data. The purpose of this study is to deal with the information that is publicly available and can be found by searching of open databases or general public information that is neither confidential nor limited in any way. The data that we are looking at is thus best exemplified by the ”tip of the ice berg” in figure 3.3.1, i.e. the part which is above the water level and thus visible to the public.

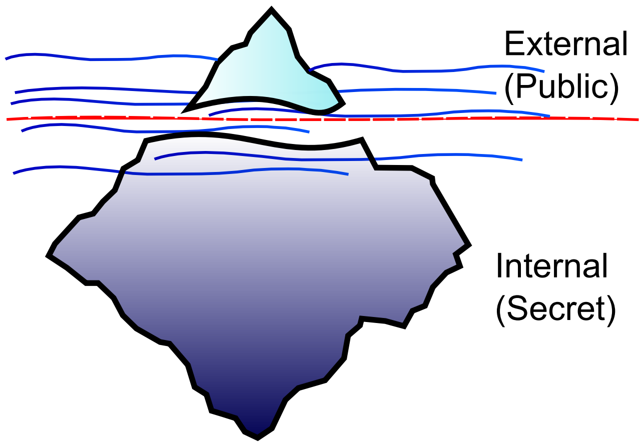


Figure 3.3.1: *The ice berg concept of external and internal knowledge. Source: SKF Internal [SKF]*

This results in that what can be known about a company is two-fold. There is the *company internal* aspect of information and there is *external information* which the company publishes. One of the primary approaches of the current work of the TIT is to describe what can be known about SKF from an external perspective. By also knowing the internal knowledge one can then make inferences on the correlation between the publicly available information and the company internal perspective, giving indications of the company's strategic profile, and the correlation it has with the publicly available information [SKF].

The primary sources of publicly available information from a company come from the company's own material such as product catalogs, business news and news related to that company. It also consists of publications and patents published by the company or by people or academic institutions affiliated with the company. This type of public information may be analyzed in a multitude of ways. One typical analysis that is usually done is citation analysis (between patents or within research domains) and network analysis (of scientists and affiliations of collaborators) [Por05]. These concern the graph structure of documents and how they relate to each other. For instance one could view the forward-, backward- and multi-citation networks as graphs describing the neighborhood of a single document and also its most related documents and topics. In the area of network analysis one would connect one scientist in one organization with one academic institution and thus build networks from the data, based on automatically generated correlation matrices [Tho11b], for an example see figure 3.2.1.

3.3.2 Limitations with this type of data

The data analyzed only has one common type of information that may be the basis of comparing and associating the documents, based on one single framework. This common data is the raw text in the document or the abstract. This text is thus the only part of the data that we can truly analyze across all the three different types of the data. In the case of publications we will usually have structured text, separated into abstracts and full-text. Publications also contain information in the form of classification codes for a general classification of the document into one or more superseding knowledge groups. For patent documents we in turn have the classifications according to different patent classification schemes and also structured text fields. Besides this patents also have the rewritten Derwent abstracts and titles, which give these documents a higher value from a data analysis perspective [Seb02].

The limitations depending on what type of document being analyzed will be due to the information available in that document. Also since algorithms dealing with automated text analysis in principle are statistics based, no "intelligent" analysis of the documents is done. A human might recognize a document as a chemical patent, whilst an automated algorithm cannot discern this automatically. Thus the automated approach is highly compatible with the raw text documents, but lacking at the same time, since the computer cannot analyze that which is not explicitly stated or can be extracted using the suggested algorithms. The implicit associations regarding for instance the topic of the data are not discerned through this analysis.

Furthermore the different sources of documents require different methods of analysis. For the patents we can discern more information than from publications. This is best exemplified by the very structured format in patents. In essence there are some specific limitations when one wishes to model all knowledge produced within a specific area. Human beings can cognitively assess more complex relationships between different sources and gain a experience of the material. This experience cannot be formulated into text, and is the main difference between having a human analyst analyzing documents, or using a machine built algorithmic approach. Another difference is that while a human can discern more complex relationships in the data, a computer has the advantage that it can analyze a much larger set of data, in much less time [Por05].

3.3.3 Sources used, cleaning of and access to data

The primary sources of publicly available objective information regarding companies stem from the two types of documents: patents and publications. Therefore these two were also the primary sources to be used for the thesis. The data is publicly available, but the task of comprehensively mining all possible sources for this data is very cumbersome [CDM08].

The TIT in SKF uses Thomson Innovation for sourcing patents and publications. For patents Thomson Innovation provides a large coverage over many regions in the world, together with translated versions of Chinese and Japanese patents, and patent applications [Tho11a]. For publications the TIT uses Thomson Innovation which make these accessible through online databases via a paid subscription and the resources can be customized to provide different types of data, and different levels of analysis. This source is supplemented by the company internal business intelligence department. The BI department provides business news regarding the areas of interest for the company and is the primary source of BI within SKF.

The data was downloaded in plain text format for further analysis. The data was imported and structured in Excel, together with the corresponding identification number of the record and other information available in the structured fields within the exported data from Thomson Innovation.

Other information sources considered were online repositories, using information provided by Google Scholar or other means of acquiring data, e.g. the websites of different patent offices. However the effort in accessing larger amounts of information would severely limit the amount of data that could be collected and analyzed. Thus to simplify and leverage the available resources provided by SKF the choice was made to make use of the Thomson Innovation database. This was made available by the company for the researcher as a subscription to the online service.

Thomson Innovation

Thomson Innovation is the main source of data for this project. This database will be used to generate the results from search queries regarding the topic of focus. The Thomson Innovation database is searchable in nearly all indexed fields. In the case of patents this provides the possibility to search abstract, full-text, title, Derwent data generated by Thomson's own analysts, patent assignor, patent assignee, and many other fields.

Thomson Innovation is not a complete database, and it is not perfectly up to date due to delays in updating new data from the different authorities supplying the information. However as a source for historical data for analysis it is more than sufficient. For this study we are not tending to the front-line of technology research, but rather focusing on what has already been published and is available digitally through Thomson Innovation. The experiments performed in the context of this thesis do not require perfectly up to date data. However this consideration is an important topic when using the suggested framework in the future, where consideration of which databases are most up to date should be evaluated.

A parallel can be drawn with the financial markets, where the suppliers of information that are paid on a subscription basis have a requirement to deliver data as fast as is possible for financial analysts to act on. In this extremely competitive area even the physical location of the computer receiving the information is important for descreasing the time-to-act for the automated algorithms that are used.
Fortunately the technology monitoring in the case of industrial companies is not quite as fast, so the efficacy of the data provider is not the main concern. However in particular cases direct changes in the competitive environment, i.e. in mergers or acquisitions, follow up reviews of analyses should be performed to reflect the new intellectual property of the company that made the acquisition or was acquired [Por05].

Cleaning of data

The purpose of data cleaning is to reduce noise in the generated data before analysis. This is particularly important to consider when working with text based information. There are many caveats that can be avoided by preprocessing data for the needed analyses. In this sense we focus on what features to consider and analyze before cleaning the data. This also applies to what data fields are useful.
In our case we will not consider assignees or organization names. In the case of the Thomson Innovation data the same company could file patents under different organization names, and applications also may have spelling errors. This type of noise can be eliminated using automatic data cleaning and customized thesauri in Thomson Data Analyzer [Tho11b].
For this assignment in both the test phase and the full scale case we use data that is provided by the SKF internal TIT. This preprocessing of data based on the principles mentioned above has been performed and therefore these considerations are not a major problem in our analyses.

The cleaning usually consists of matching names of applicants, names of organizations, augmenting fields with text (for instance adding descriptions to the field codes for patent classification codes). Depending on the source this spell checking may already have been done for some fields. For instance news agencies generally have comparatively many spelling errors when compared with dictionaries. This means that depending on the source of the data different levels of cleaning are needed [Por05], [Tho11b]. For this project the same general approach was taken for each source: the full text and abstracts were cleaned. This was the case since these are the main fields that were analyzed and these fields are common to all documents.

Ordering of data

An algorithm leveraging many different sources of information is always dependent on the structure of these information sources. Since the common fields between the sources are the abstracts and full texts, these can be analyzed using the same part of the algorithm for all the documents, independent of type. Consideration of these aspects is important when choosing the allocation algorithm for the suggested framework. This will be further addressed in chapter 5.

3.3.4 Patents

Patents can be thought of as a three-party contract that provides the holder of the contract a *negative right.* Without going too far into details we describe a patent as a contract that grants the holder the *right to deny someone else* to use or produce the invention that has been patented. Thus it is not a right to produce it yourself, but rather it gives you the right, for a limited amount of time, to prevent someone else from doing so.

Another aspect that is important is that it is a three-party contract between the holder, the state and the public of the country where it has been issued. Thus patents are only applicable or valid in the country where they have been filed and are maintained, by paying the annuities associated to holding the patent in that region. Therefore for instance a *European patent office* (EPO) application for instance needs to be followed up with payments in each of the countries where one wishes to be able to enforce the patent. In the same way a granted patent by the patent office in India is only valid within their given jurisdiction, i.e. India [Ern03].

There are some important properties of patents and patent applications that need to be considered. Patents are granted if and only if they fulfill the basic pre-requisites:

- Novelty (that there is a novel aspect to the application)
- Inventive step (that there is a clear inventive step, and that this is not a natural development by using what was previously known)
- Industrial applicability (it has some practical use)

This can essentially be interpreted using the "hand metaphor". You (most probably) have a right hand and a left hand. Imagine now that you had no left hand, and thus only a right one. This is usable in many ways to you and it is known to you, thus you can use it and it has much applicability. For instance you can shake hands, you can hold things and you can thrust it. Imagine now that you were to "invent" the left hand. This would imply only that you take the mirror image of the right, however the left hand is not similar, it is not the same. Thus the novelty aspect is fulfilled, especially there is an inventive step and also, it has applicability. It enables you to now do things with the two hands together. This would fall under a patentable invention, given that the left hand was not known previously.

To give a further introduction to patent information and the information made available through the Thomson Innovation database we present the main classification schemes available world-wide, IPC, ECLA, US, F-terms and the Derwent system in appendix D.1 on patent classification schemes.

3.3.5 Publications

Scientific publications are the main source for disseminating new research results or new knowledge in the scientific domain. Usually the process is initiated by a research team that after thorough analysis decides on a topic of research. This research is then conducted and if successful the results are then submitted in the form of a scientific paper to a journal for evaluation. If the topic is found to be appropriate and the peer reviewing procedure grants the article this results in the paper being accepted for publication. After this it is published and the information becomes available through the journal [Por05].

Publications are some of the most important sources of scientific knowledge. Many types of research rely heavily upon publications as the natural arena to promote ones results and to disseminate this knowledge [TLG04]. Therefore publications are a natural choice when one wishes to analyze documents for examining the spreading of knowledge and interest in a specific scientific area. As mentioned above this is not a perfect source, and with regards to information communicated it is only explicit knowledge that may be captured using automated algorithms, but nonetheless publications are still respected as one of the top sources of information, especially in TI being ranked as the number one source in research done among firms by Lange [Lan94]. For a description of different publication classifications we refer the reader to the references and appendix D.2 on publication classifications.

4 Text classification theory

4.1 Machine learning

Machine learning (ML) is a field of artificial intelligence where algorithms and computers are employed to model and adapt output functions based on a specific input. Usually machine learning concerns itself with either learning an algorithm to recognize a pattern or modelling a certain set of input data. This is called supervised learning. In other cases machine learning deals with clustering or grouping of data irrespective of a predefined sets of classes, to which these data should be assigned. This is known as unsupervised learning. The branch of machine learning has evolved from artificial intelligence, and at the heart of ML lies the principle that the algorithm should learn, or be able to generalize, based on past experience which has been fed into the algorithm [CDM08]

4.1.1 Unsupervised learning

Unsupervised learning covers algorithms that do not use preclassified data sets to train the classification algorithm. These methods usually use the inherent structure of the data to classify the documents. These measures are usually based on some form of similarity measure, where the algorithm allocates documents to classes based on some criterion. This in turn also means that this type of classification cannot be tested in a general sense to ascertain if the results are accurate as per the class some data point is classified into. One can still evaluate the "relatedness" of documents when studying the final results and by human evaluation one can relate the results and study their coherence.

For document clustering unsupervised learning algorithms use measures such as semantic similarity, word co-occurence and similar methodologies to sort documents into similar clusters. Some methods are based on separating each document from all others, and then grouping the documents that are furthest away from others. Furthermore it is evaluated so that the groups of documents in turn are as near to each other as possible within the group. This type of clustering algorithm is a centroid based clustering and can be varied to provide more or less weights to the points closest to the document in the space where they are being analyzed.

Since unsupervised learning does not use any pre-classified examples and actually does not fit elements of data (in our case documents) into a predefined classification structure, these methods are not relevant for the work in this thesis. These methods all have merits on their own and are used in for instance Yippy's online clustering-based searching system [Yip]. Regardless these will not be elaborated on further in this thesis.

4.1.2 Supervised learning

In supervised learning we have the situation that given a set of pre-classified instances we need to learn a classifier to assign previously unseen documents to the right class. This means that first we need to learn the classifier, then use this to model how the attributes of the documents combine to distinguish which class they belong to. In mathematical terms this can be termed as a text classification problem, and following the definition in Bekkerman [Bek03], we have:

Given a training set $\mathcal{D}_{train} = [(d_1, l_1), (d_2, l_2), \ldots, (d_n, l_n)]$ of n labeled text documents, each of which are from a full set of documents $\mathcal{D}$ where each label $l(d_i) = l_i$ belongs to one of m predefined categories $\mathcal{C} = [c_1, c_2, \ldots, c_m]$.
Each document is thus represented by the vector of weights corresponding to the r (total number of) terms in the dataset being analyzed.

The goal of text classification is then to create a classifier, or a hypothesis, $h : \mathcal{D} \to \mathcal{C}$ that will be as accurate as possible in classifying unseen documents to classes from $\mathcal{C}$.

4.2 Automatic text classfication

There are many different classfication algorithms that may be used to classify text automatically. As a preprocessing of the algorithm, input documents are parsed into some form of text-document association scheme. In this thesis we focus on analyzing the text within documents based on their semantic content, and will use the *bag-of-words* (BOW) model. The BOW model classifies the terms in the documents creating an allocation corresponding to what is called a *term-document* (TD) matrix. Using this model we describe the document collection that is to be classified as a set of vectors consisting of document terms representing each document, which will be further explained below, see figure 4.2.1, table 4.2.1 and Kumar [PNT05].

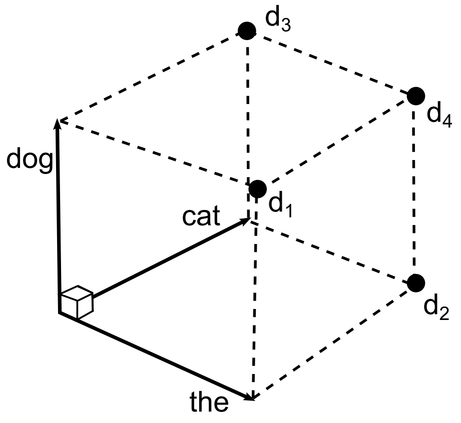


Figure 4.2.1: *Example of how the vector space model works for a document collection with a total of three terms (*the (1), cat(2) *and* dog (3)*). In this example doc. 1 has vector representation* $d_1 = (1, 0, 1)$*, doc. 2:* $d_2 = (1, 1, 0)$*, doc. 3:* $d_3 = (0, 1, 1)$ *and doc. 4:* $d_4 = (1, 1, 1)$*.*

Text classification can be viewed as an automatic classification problem. This consists of a number of given steps to create a classfier. Given a set of training data, where the data contains some types of attributes, one of these attributes is which class the data belong to. Given this we wish to find a model for the class attribute by using the values of the other attributes in the data. Once we have the training set and we have created a way of mapping the attributes to data, we finally use the created classifier to classify unseen documents, based on what it has learned. In most cases a fully labelled (or classified) collection is split to construct one set of training data, one set of validation data and one set of test data to learn the algorithm. The validation set is used to ensure the functionality of the algorithm. The test set is used to estimate the accuracy of the model.

Some other classification problems are image recognition (finding similarities in image data), automated categorizing of credit card transactions or as in this thesis, classifying documents based on their textual content. Some of the most popular classifiers include but are not limited to: rule-based methods, k-nearest neighbours, artificial neural networks, naïve Bayes and support vector machines [PNT05], [CDM08].

4.2.1 The classification task

The assignment for this thesis was to develop and design an algorithm for automatically classifying text documents based on their textual content. This was to be done using a number of pre-labeled documents to train the classifier, with the final goal of being able to classify new documents that were unlabelled based on their textual content.

As described in table 3.3.1 there are multiple types of information that may be used depending on the data at hand. One can have unstructured, structured and semi-structured data corresponding to the different sources of information. In the thesis we have attempted to implement the semi-structured information, but have left the structured information, due to time limitations. The assignment was also specifically designed not to focus on the specifics of the different possible configurations, but to develop a process for classifying any document based on its textual content into the predefined set of classes (the SI).

Categorization can be done into a hierarchical set of categories or into a flat set categories. The first is a hierarchical structure, similar to that of folders in a computer, see figure 4.2.2. The other is to organize it as a number of classes, where no class is a sub-class of another, see figure 4.2.3. Thus we have two general frameworks, where the hierarchical corresponds to the TT described in section 3.2.3 and the second corresponds to the allocation to SI discussed in section 3.2.3. Since the assignment only concerns itself with flat classification we will only use this type to classify documents for the purposes of this thesis.

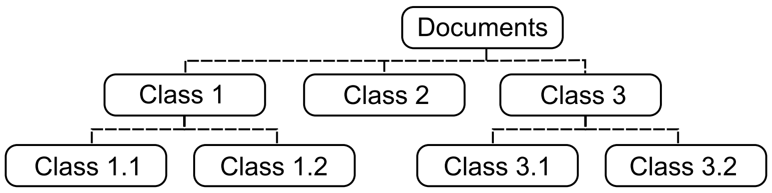


Figure 4.2.2: *Example of hierarchical document categorization (as in tech. tree approach).*

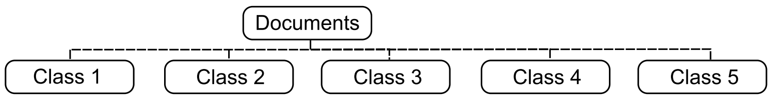


Figure 4.2.3: *Example of flat document categorization (as in this classification task).*

Finally the general process of creating a classifier for automatically classifying documents can be stated in the following steps:

1. Extract the relevant documents to be classified, together with the pre-classified ones
2. Construct a set of features that are a representation of each document (the attributes)
3. Define a sub-problem of finding the relevant features and constructing the classifier
4. Use the created classifier to classify previously unseen documents
5. Evaluate the accuracy of the classifier using both labeled documents and expert feedback

We will address each of these steps, starting with the challenge of feature selection. But first we will give some background theory on a couple of issues within automatic analysis of textual data.

4.2.2 The data classes

As mentioned above the internal classification scheme SKF has developed is known as SI. This thesis will not investigate all the possible allocation paradigms that may have been used for classifying data for the purposes of this thesis, but will comment on the nature of this classification scheme. It is important to realize that the classficiation into SI is somewhat arbitrary.
The arbitrary nature comes both from the definition of the classes and the overlap between different classes. For instance one document may contribute to intent A) but may be intimately connected with intent B) as well. This means that the data classes, as far as allocation is concerned can be termed *siblings* as defined in figure 3.2.3. This type of non-disjoint nature of the distribution of the SI is something that will not be addressed further in the thesis, other than the comment that this may have a diminishing effect on the accuracy of the classifier.

The created classifier cannot perform better than the quality of the inputs provided for learning it. In this case the type of allocation that a TT provides, see section 3.2.3, is better suited for achieving a disjoint set of both attributes and classifications. This however was not a part of the specific task for this thesis work and was therefore not elaborated on further.

4.2.3 Some characteristics of text classification

Some problems that arise in automated text classification are *high dimensionality of data*, *sparseness of data* and *unclear labelling*. Usually textual documents cannot be represented perfectly when the dimensionality of the document is reduced. Furthermore the sparseness of terms over the whole corpus or the problem of too few documents in some classes that are accurately labelled may create issues when creating the classifier. As mentioned the unclear labelling of documents is another problem. The effect of this can be reduced by limiting the problem above to only deal with labelling each document into one class (binary classification) at a time. In this case we are dealing with a *uni-labelling* text classification problem.

Another type of issue that presents itself is that many terms may deal with multiple topics, which is known as *polysemy*. Such words as Jaguar (a car, a computer OS and an animal) can give problems when simply recognizing terms in documents. This issue is usually addressed by making use of dimension modifying techniques such as latent semantic indexing, which is discussed below in section 4.3.6. Another way of dealing with this problem is to reduce the number of analyzed dimensions, to focus on the main principal directions in the matrix describing the data.

4.2.4 Dimensionality reduction

Different methods for reducing the dimensionality of text data exist. All of which concern themselves with finding a more compact way of representing each document. One efficient way of doing this is by reducing the number of words (and thus the dimensionality of the representation space) that are used for the document representation. This can be employed by using for instance, *stop-words*, *stemming*, topic specific *thesauri*, extracting *keywords*, *n-grams* of words, principal components of documents *SVD/Latent semantic analysis*, filtering of very high and/or low frequency words *tf-idf*, etc. For the interested reader this is discussed further in appendix E.

For the purposes of this thesis work we will reduce the dimensionality using stemming, stop-words, feature modification (weighting schemes) and SVD/Latent Semantic Indexing to represent the analyzed documents.

4.2.5 Efficient representation

Representing information as efficiently as possible is important when dealing with a large corpus of data. The methods used for data classification and the algorithms chosen should be able to handle an increasingly large corpora. These corpora also grow with the number of sources being analyzed and with the number of documents available in machine readable form. This means that efficient representation of each document is a prerequisite for enabling efficient document classification on a larger scale. For representing the documents analyzed in this thesis we made use of the BOW model, see section 4.2.8.

One reason for choosing the BOW model for representing the terms appearing in documents, is that besides being intuitive, it is easy to interpreted and explain. The *term-document matrix* (TDM), which is the essential part of the analysis is also easy to compute using the BOW model. The BOW model represents each document as a simple collection of terms, and does not distinguish between word order, or other possible representations. This representation is very efficient, since we only store the non-zero elements of the TDM A, which will be presented below. This means that it becomes even less memory-consuming to represent the full text corpora being analyzed.

One explanation for the efficiency of the BOW model, where each document is represented by the terms appearing in that document, is that each document vector will be very sparse. This sparseness in the vector (i.e. that most terms do not appear in the document), is most clearly explained by *Zipf's law.*

4.2.6 Zipf's law

In general when studying the distribution of words in languages, there is a recurrent phenomenon known as Zipf's law. Zipf's law states that in a language, the frequency of words used is distributed so that the most frequently used word occurs twice as often as the second most frequent. Furthermore the second most frequent occurs twice as often as the third most frequent, and so on. This law thus dictates that the sparsity in the space needed to represent the data is natural. Zipf's law gives a distribution to consider when ranking words in documents, since there is a clear deviance from a uniform distribution for the words in any textual corpus [Zip35].

This also has clear consequences for text classification problems. Zipf's law should be considered when analyzing text, and one of the most important considerations is the weighting of terms. The weighting of terms dictates how much emphasis is given to different terms when analyzing a text. The findings of Salton et al. in 1975 and 1988, [GS75], [SB88] show that the most frequent words in any document collection say the least about any one of the distinct documents. Furthermore they also find that the least frequent words also add the least information when classifying documents. This is a natural part of reshaping the *term-document vector* (TDV) and will be elaborated on further in section 4.2.9.

We now shift our focus back to the main topic of automatic text classification, and more specifically what features of a document that should be included in the analysis of documents.

4.2.7 Features

Feature selection

Feature selection is a vital part of the data representation, which must be done before classifying the data. Given a corpus of documents there are a multitude of different features that may be analyzed. One example of this is the case of search engines such as Google and Bing. Some of the research performed at Microsoft gives an indication of how complex the feature selection may be. One article published by one Microsoft research team defines 137 different features based on different scoring methods and document relevance models to create a meta-score that defines the query-document pair. This query-document pair is a linear combination of all the implemented classifying models and defines how well one document fits a given query [Mic11]. By using this approach the research team achieves a monotonically decreasing ranking of all documents based on their respective relevance to the query.

One feature in documents to analyze would be to look at phrases of words, and extract n-grams: two-word phrases (2-grams) or three word phrases (3-grams) and so on. However to parse out this data and represent it quickly becomes computationally heavy and is not feasible for larger collections of data. This of course is an improvement upon the simple BOW model, but an associated problem with using the n-grams for analysis is that they strongly increase the dimensionality of the data. For the purposes of this thesis we will only use the simple BOW model to represent the terms in the documents being analyzed.
Another feature to use for representing documents more completely is to use the different meta-tags that may be associated to some words. Google, for instance, uses this in their PageRank algorithm to use anchor text and headings to better represent the content of a webpage. This is of course also a good source for a more complete description of the document, but as in the case of n-grams these will not be addressed in this thesis.

Feature selection is a rich area of research and the choice of features to analyze is an important part of the text classification task. Unfortunately there are no simple methodologies for choosing which features to analyze, or which ones are more relevant for a specific type of corpus. Some indications of this can be found in Bekkerman [Bek03] and also in Chakrabati et al. [SC98]. They discuss the probabilistic distribution of single terms over topics, or using the structured information, such as table and image captions and headings to extract a more structured set of data.

For a more comprehensive introduction to possible features that may be included in text classification or similar tasks, such as document retrieval, we refer the reader to the book by Raghavan [CDM08].

Feature generation

The problem of choosing features is not only concerned with analyzing the given documents as they are. As noted by, among others, Bekkerman [Bek03] there is another aspect to features used to describe documents, other than the selection of features to analyze. For analysis we may create, or generate, features to be analyzed. We could for instance extract part-of-speech tags, extract key n-grams and then check the correlation of different n-grams in section or parts of the document. Another feature that may be generated may be links between different documents, or references from different sections to other sections. This can then add data into the given document based on for instance the links contained within that document.

4.2.8 Vector space models

One of the classic models for document retrieval was proposed in the 1970s when Salton et al. [GS75] suggested the vector space model for automatic indexing. This model uses the basic notion that if one knows all the terms in all documents within a collection one can represent each document d as a collection of terms in the form of a vector describing which terms appear in that specific document. This means that if there are r distinct terms in the whole corpus and D documents we have that each document d_j may be represented by a r-dimensional TDV: $d_j = (w_{1j}, w_{2j}, \ldots, w_{rj})$. Here w_{ij} represents the weight of the ith term and is equal to 0 if the term does not occur in document j and > 0 if it appears in that document. Thus we can represent each document d_j in the collection as a r-dimensional weight vector $w_j = (w_{1j}, w_{2j}, \ldots, w_{rj})$.

The reason for changing from term to weight, is that there are many different ways of representing each terms importance in any one document. This is reflected in a weighted approach for describing the prevalence of that specific term in a given document, as described in section 4.2.9.

In the vector space model suggested by Salton the method is to first calculate the TDV for each document and then project each document d_j onto the r-dimensional unit sphere (i.e. normalize the document vectors length) to 1, so here the weights are regulated to be normalized by the total length of the vector d_j, see figure 4.2.1. This gives a simple way of comparing the similarities between documents by, for instance, using the cosine similarity measure in equation 4.2.3. We describe some different weighting measures extensively in section 4.2.9.

To give a short example of how the vector space model works, we assume the three following documents:

d_1: The dog ate the grass.

d_2: The grass blew in the wind.

d_3: The dog ate in the wind.

Assigning them to a TDM can be done simply by adding 1 each time term i appears in document j, giving the following TDM, which will be denoted as A in this thesis, for the three documents above as:

	d_1	d_2	d_3
t_1 = the	2	2	2
t_2 = dog	1	0	1
t_3 = ate	1	0	1
t_4 = grass	1	1	0
t_5 = blew	0	1	0
t_6 = in	0	1	1
t_7 = wind	0	1	1

Table 4.2.1: Example term-document matrix A (using the Bag-of-words model) with $r = 7$ and $n = 3$.

As can be seen the model gives a simple and intuitive approach to describing documents. However the model is limited in the sense that for instance t_1 occurs with the same frequency in each of the documents, meaning that it adds no information for distinguishing between the different documents. This way of analyzing the terms, or more generally, the features of any document is the standard course of action. This is done to represent each document as a r-dimensional vector, so that each document is comparable to the other documents in the corpus.

4.2.9 Weighting schemes

Once the features that represent each document have been chosen we have a representation of the document using a vector of features. In this thesis however the only features that have been analyzed are the words and their occurence within any document in the analyzed corpora. This means that the only features that have been analyzed in the suggested model for our experiments are the terms occuring in the different documents. The different ways of representing this TDV will be discussed in this section.

There are a multitude of weighting schemes that can be used to weight the TDV. Among the different versions we have binary, frequency, log-frequency, and many other possible weight schemes. For a comprehensive survey we refer the reader to Raghavan et al. [CDM08], Tan et al. [ML05] and especially Salton and Buckley [SB88].

If we simply assume a BOW model for the terms occuring in a document, that document may be represented using a (sparse) vector containing values $w_{ij} > 0$ for the terms (i) that occur in that document (j). There are many different choices in how you represent the document vector, such as the weighting scheme applied and if weights should be binary, integer or ratios.
Following Salton and Buckley [SB88] we will divide the weighting of any TDV into a weighting based on three components. These components are the term frequency component, the collection frequency component, and the normalization component. A summary of the different schemes discussed in Salton and Buckley's article is given in table 4.2.2.

Term Frequency Component		Description
b	1.0	binary weight 1 if term occurs, 0 otherwise
t	tf	term frequency (# of times term occurs in doc.), 0 otherwise
n	$0.5 + 0.5\frac{tf}{\max(tf)}$	augmented normalized term frequency (tf factor normalized by the maximum tf in the vector and limited to 0.5-1)
Collection Frequency Component		
x	1.0	no weigthing
f	$\log\left(\frac{N}{n}\right)$	inverse collection frequency, i.e. the logarithm of the inverse of the document frequency (idf) N is the total number of documents in corpora, n is the # of doc. in which the term appears
p	$\max\left(0, \log\left(\frac{N-n}{n}\right)\right)$	probabilistic weighting (tf weighting with the inverse document collection frequency)
Normalization Component		
x	1.0	no weighting
c	$1/\sqrt{\sum w_i^2}$	normalized by weight vector length

Table 4.2.2: Some possible weighting schemes for term-document representation, when using term-document incidence as the basic measure. w_{ij} is the weight of term i in document j. Source: Salton and Buckley [SB88] and Raghavan et al. [CDM08]

Some of the conclusions reached in Salton [SB88] discuss the appropriate weighting scheme to use for collections from different sources. In their results the term frequency component should be n for technical collections with varied and specialized vocabularies. For the collection frequency component there are similar results using both f and p and thus the standard idf f is chosen for technical collections. In the case that there is a strong deviation in document length, the recommendation for normalizing is to use c. This gives similar weights to documents of different length. Thus using a triplet we can describe different weighting schemes, in the case of simple incidence without any normalization we have bxx. Blom investigates other possible weighting schemes [Blo04], and similarly to Salton [SB88], the best weighting pair when dealing with document query matching. The findings are consistent with each other and also with other experiments done by for instance Dumais [Dum91].

In our case for the topic of automatically categorizing technical abstracts and patents we chose to use nfc. If there is a very varied vocabulary (for instance in larger collections with mixed sources) the recommendations would be to choose tfc or(tpc) [SB88].

Now we have defined how we represent the documents and how these representations are weighted to give a more balanced description of the given data. Thus we have the input for the comparison between two distinct documents. For these purposes an important aspect are the similarity measures, which are discussed in the next subsection.

4.2.10 Similarity measures

When grouping or clustering data an important measure is the similarity between the sets of evaluated features. Measures of similarity are dependent on the type of feature being analyzed. Given the features mentioned above, we give an introduction to a number of similarity measures below, as some of these are used in the clustering algorithms considered. For different attributes some possible similarity measures may be defined as:

Type of attribute	Dissimilarity	Similarity
Nominal	$D = \begin{cases} 0 \text{ if } d_i = d_j \\ 1 \text{ if } d_i \neq d_j \end{cases}$	$S = \begin{cases} 1 \text{ if } d_i = d_j \\ 0 \text{ if } d_i \neq d_j \end{cases}$
Ordinal	$D = \frac{\lvert d_i - d_j \rvert}{n-1}$	$S = 1 - \frac{\lvert d_i - d_j \rvert}{n-1}$
Interval or Ratio	$D = \lvert d_i - d_j \rvert$	$S = \frac{1}{1+D}, \quad S = 1 - \frac{D - \min_D}{\max_D - \min_D}$

Table 4.2.3: Different measures of similarity for different types of data. Source: Kumar [PNT05]

where D defines the dissimilarity between documents and S denotes the similarity. Furthermore d_i and d_j represent the vector of attributes for two different documents. n is the number of documents considered. As can be seen there are a multitude of similarity measures and each is dependent on the characteristics of the underlying attribute that we are measuring. One typical measure when we are dealing with multi-dimensional data is the euclidean distance. The euclidean distance between two vectors $d_i \in \mathcal{R}^r$ and $d_j \in \mathcal{R}^r$ is defined as:

$$dist = \sqrt{\sum_{k=1}^{r}(d_{ki} - d_{kj})^2} \tag{4.2.1}$$

where the vectors can be normalized prior to measuring the distance, if the scales differ greatly. Other measures of distance include the Minkowski (or taxi-cab) distance.

A distance has well defined properties. All distances has following three properties,

- positive definiteness $dist(d_i, d_j) \geq 0$, $\forall i, j$,
- symmetry $dist(d_i, d_j) = dist(d_j, d_i)$ and
- satisfies the triangle equality $dist(d_i, d_k) \leq dist(d_i, d_j) + dist(d_j, d_k)$, $\forall i, j, k$.

This means that all measures that satisfy these three properties is a distance. In the same sense a similarity is also defined using the following two properties: (1) $S(d_i, d_j) = 1$ iff $i = j$ and (2) symmetry.

Other measures of similarity when studying term vectors are the Jaccard coefficient, Cosines similarity and the Tanimoto (Extended Jaccard) coefficient. These are defined as follows, Jaccard (for binomial vectors):

$$Jaccard(d_i, d_j) = \frac{|d_i \cap d_j|}{|d_i \cup d_j|} \tag{4.2.2}$$

The Jaccard coefficient does not consider how many times a term occurs in a document. Terms in the collection that are rare provide a lot of information regarding the specific nature of the document. This is the case when we represent each document as a binary vector over the terms, which of course only indicates if a term occurs in a document or not [CDM08]. If we instead consider the *count* of the terms in each document we get a better representation, as described in 4.2.1.

Cosine similarity is defined as:

$$cos(d_i, d_j) = \frac{d_i \cdot d_j}{||d_i||||d_j||} \tag{4.2.3}$$

If we use the vector space model described in figure 4.2.1 and represent each word and project these documents onto the unit sphere, we have a length normalized measure for comparing documents (confer with equation 4.2.3). The two-dimensional case of this comparison is presented in figure 4.2.4.

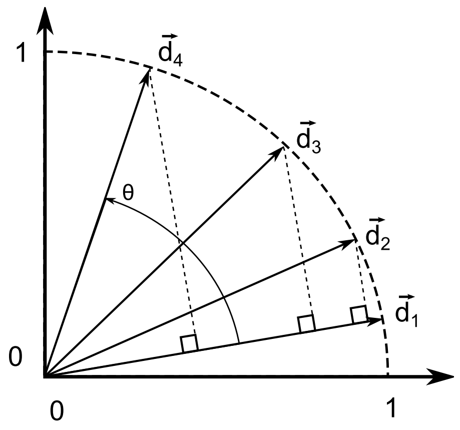


Figure 4.2.4: *Two-dimensional document comparison using the cosine similarity measure. This measure quantifies the r-dimensional "angle" between two documents as a measure of their similarity. In combination with the unit sphere projection this is a consistent measure for comparing documents based on the bag-of-words model (see figure 4.2.1). In the figure the two-dimensional case is presented, with an angle θ between documents d_1 and d_4 which indicates that they are less similar than d_1 and d_2 or d_3.*

Finally the Tanimoto coefficient is defined as:

$$T(d_i, d_j) = \frac{d_i \cdot d_j}{||d_i||^2 + ||d_j||^2 - d_i \cdot d_j} \tag{4.2.4}$$

which reduces to the Jaccard coefficient for binary vectors.

4.3 Classification algorithms

There are a number of different clustering methods to use in supervised learning. In supervised learning the problem is to classify unlabeled data, by training a classifier on prelabeled data. Clustering methods can be based on classifying different statistical properties, grouping objects based on similar charateristics or simply recognizing patterns in the features that are used to represent the documents.

In the classification task that was provided by SKF the goal was to use a set of prelabeled documents to produce a classifier that can be used for classifying previously unseen documents. For this to be done the terms in the corpus of documents being analyzed were chosen as the relevant features. These TDV are the full description of each document, and by using the prelabeled documents we investigate what methods may be used to generate a classifier using these vectors.
The principles for analyzing these vectors are sometimes connected to the similarity measures discussed in section 4.2.10. In other situations the importance of the patterns in the given TDV are the basis for allocation. Other methods use the (spatially) neareast documents in the feature space. Lastly another type of models that we have considered are probabilistic models, that base themselves on the analysis of the random distribution of words across topics or documents.

Below we will describe some different clustering methods and we finally give a short list of pros and cons of the different models.

4.3.1 k-nearest neighbours

The first type of supervised classifier that will be presented is example based classifiers. *k-nearest neighbours* (k-NN), is one of these example-based classifiers. k-NN as the name implies, considers the characteristics of the k "closest" data (in some sense) in the training set, for classifying new data. After choosing which features to analyze, the process suggested by Yang and Chute [YY94] is the following. For classifying document $d_j \in c_k$ k-NN looks at the k nearest neighbours. Given that enough data points are positive the document is classified as belonging to the dominant class. When constructing the classifier one important aspect is to decide the appropriate number (k) of neighbours to consider to get good results on the validation set of documents. Some results indicate that values of $30 \leq k \leq 45$ should provide the best results, as described by Yang et al. in [YY94] and [YY99].

For a mathematical description of the k-NN algorithm, see equation 4.3.1. If we denote the similarity $sim(d_x, c_k)$ between document d_x and class k as the distance between two document vectors in some multidimensional space, in which we measure the similarity, we have the following equation.

$$sim(d_x, c_k) = \frac{\sum_{d_j \in k\text{-NN}}(sim(d_x, d_j) I_{d_j \in c_k}(d_j, c_k))}{\sum_{d_j \in k\text{-NN}} sim(d_x, d_j)} - C_k \tag{4.3.1}$$

where $I_{d_j \in c_k}(d_j, c_k)$ is the indicator function which is $= 1$ if the prelabeled document j belongs to class k and 0 otherwise. C_k is a constant that is a threshold for classifying a document into category k. This can also be thought of as the required summed similarity over all the pretrained documents for classifying that data point to class k. An illustration of the principle of k-NN can be seen in figure 4.3.1.

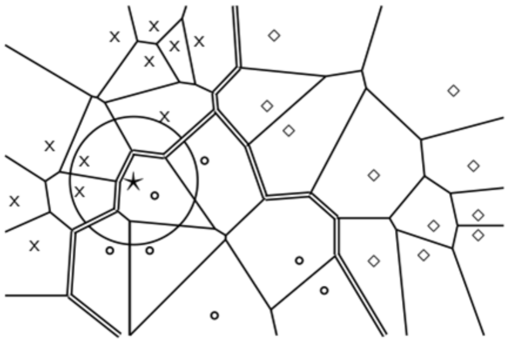


Figure 4.3.1: *Example of k-NN and how they group data ($\diamond$, $\times$, $\star$, $\circ$ ($m = 4$ classes)), based on the elements closest in the feature space. Source: Raghavan et al. [CDM08]*

k-NN is not a linear classifier, as can be seen in figure 4.3.1 however, it is piecewise linear, in distinguishing between the different classes. As can be seen in figure 4.3.1 the k-NN algorithm assigns each of the different data to the nearest neighbors class, to reach the final allocation, shown in figure 4.3.1. For a deeper discussion regarding the model we refer the reader to the book by Raghavan [CDM08].

4.3.2 Rocchio method

The Rocchio method is another classifier that looks at the surrounding prelabeled data points for classifying new data. The method computes a linear regression based on a set preclassified documents and uses these to assign new documents. The resulting clustering is based on computing the classifier for class k as $c_k(d_1, d_2, \ldots, d_n)$ where $1 \leq k \leq m$ and m is the number of classes describing the corpus using the formula:

$$c_k(\vec{d}) = \beta \sum_{\{d_j \in [+]_k\}} \frac{d_j}{\#_{[+]_k}} - \gamma \sum_{\{d_j \in [-]_k\}} \frac{d_j}{\#_{[-]_k}} \tag{4.3.2}$$

where β and γ are the enforcement parameters. These are set to reward being close to the typical "representative" test documents in the class $[+]_k$ (with similar term content) and punish $(-\gamma)$ being near documents that are *not* in the class $[-]_k$. Some typical values for β and γ that have been used are $\beta = 16, \gamma = 4$ which more strongly emphasizes the reward of being correct, and deemphasizes the punishment for being incorrect, see for instance Cohen and Singer [WWC99]. In essence the Rocchio classifier computes the centroid of all data from one class, adds the document if it is closest to that centroid, and then recalculates the weights based on the new set of documents. This means that the centroid in the end will converge to a position "centered between" each of the documents that represent that class, see figure 4.3.2

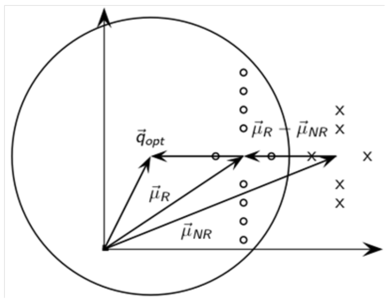


Figure 4.3.2: *Example of Rocchio classifier and how it groups data, based on elements closest in the feature space in the class and discriminating the documents not in the class. In this image the $\beta = 1, \gamma = 1$ results in that both the "correct" vector $\vec{\mu}_R$ and the correction vector for "not correct" $\vec{\mu}_R - \vec{\mu}_{NR}$ combine to give the final resulting optimal classifier as $\vec{q}_{opt}$. Source: Raghavan et al. [CDM08]*

One problem of this seemingly efficient and simple approach is that when computing the training set and finding the characteristic data, it may be the case that the data that should be categorized does not fall near the centroid of the correct class. The Rocchio classifier will then fail to classify these documents correctly due to their non-similarity with the training set. An example of this may be the semantic content in two news articles on foreign policy which may have very disjoint sets of terms. This could be two articles where one deals with the foreign fiscal policy of the US towards the euro and another concerning their ongoing operations in Afghanistan.

4.3.3 Support vector machines

Support vector machines (SVM) is a statistical learning scheme proposed by Vapnik [Vap95]. In the context of text categorization there have been many studies to support the usage of SVM, see for instance [Joa01]. The SVM is a model that posits that the data are separable by inducing a linear split between two classes of data, analogously in a multidimensional case one can separate the data using the corresponding separating hyperplane. The principle behind SVM is to separate data into groups, by placing a linear split between the data groups that *maximizes the margin* between the two, see figure 4.3.3.

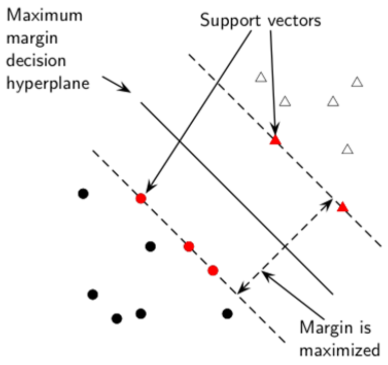


Figure 4.3.3: *Example of support vector machines for separating data from coming from a data set consisting of two separate classes: ● and △ in the 2-dimensional case. Source: Raghavan et al. [CDM08]*

As can be seen the optimum corresponds to a vector perpendicular to the separating support vectors that has the maximum length possible. In some cases the data may not be perfectly separable in the dimensions, and this is usually solved by employing slack variables in the optimization problem of finding the optimum hyperplane. This allows for some deviations, and by using a fixed cost function one may optimize to find the best possible hyperplane using the slack to accomodate for data which is not linearly separable.

The optimization problem in SVM can be described as an optimization problem and following Raghavan et al. [CDM08] we have (with y as the scaling of the hyperplane, $\vec{w}^T\vec{x}+b$ is a hyperplane (b is an arbitrary constant) and r is the geometric margin):

1. Define the geometric margin (the maximum width of the band that the separating support vectors between the two classes can have) as:

$$r = y\frac{\vec{w}^T\vec{x}+b}{|\vec{w}|} \tag{4.3.3}$$

2. Then assuming that all data are at least one distance unit from the hyperplane describing the split: $y_i(\vec{w}^T\vec{x}_i+b) \geq 1$

3. Given this, find the vector that maximizes the geometric margin (which is $\sim 1/|\vec{w}|$), i.e. minimizes $|\vec{w}|$:

$$\text{Find } \vec{w} \text{ and } b \text{ such that:} \tag{4.3.4}$$
$$\frac{1}{2}\vec{w}^T\vec{w} \text{ is minimized,} \tag{4.3.5}$$
$$\forall\{\vec{x}_i, y_i\}, \text{ where } y_i : y_i(\vec{w}^T\vec{x}_i+b) \geq 1 \tag{4.3.6}$$

4.3.4 Artificial neural networks

Artificial neural networks or ANN are a type of supervised learner algorithm that takes inputs and maps them to outputs, through elements called neurons. These neurons are simply transfer functions that (usually) assign a value between 0 and 1 to the output. Thus when combining these neurons outputs these give a description of how much each input contributes to the given output. An ANN consists of an input layer, a hidden layer and an output layer. There can also be designs with multiple hidden layers, but these are not addressed in this thesis. The different layers consist of neurons, there can be any number of neurons in any layer, in figure 4.3.4 we see an example ANN.

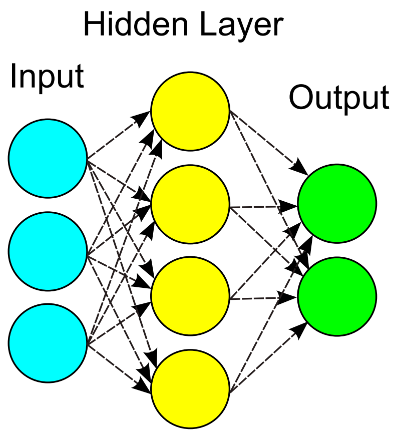


Figure 4.3.4: *An example neural network with three layers, input, hidden and output. The input layer has 3 nodes (or neurons), the hidden layer 4 neurons and the output layer has 2 neurons.*

The simplest ANN model is called a perceptron and was first conceived of in 1957 by Frank Rosenblatt [Ros57]. It was suggested for use in text categorization in 1997 by Dagan [ID97] and Ng [HTN97]. The perceptron functions as a simple linear transfer function that maps an inputs to an output, weighting the input in some way, therefore it has 1 input neuron, 1 neuron in the hidden layer and 1 output neuron. The use of a sigmoid activation functions in the neurons in the hidden (and output) layer has also shown itself valuable, and is presently a go-to choice in text categorization tasks, see for instance Yu [BY09], [AJCT05] and [AJCT06].

ANNs have successfully been implemented in a multitude of tasks, among which are text categorization, email spam filtering, grouping of financial data, and many other clustering applications [PNT05], [CDM08], [BY09].

4.3.5 Latent semantic analysis

In the article by Dumais et al. [STD90] the approach is to use a vector space model as the basic concept for representing documents. The principle from Salton's work [GS75] on the vector space model is used to establish a document representation by using the TDM. Using this concept as the underlying structure, the purpose of Dumais et al.'s work was to find a more efficient conceptual representation than the original document vectors to match these with query vectors described in the same way. These more representative vectors are then used for matching documents and their contents to queries represented in the same space.

The idea is to represent the TDM A by using the k-rank approximation A_k of the matrix which is created using the SVD of A, see appendix F.1.2 for a description of SVD. Given this approach, and as described in appendix F.1.2, the approximation of the true TDM, is the best possible approximation to the true TD distribution in $A \in \mathcal{R}^{m \times n}$ using k dimensions, where $k < l$, and $l = \min(m, n)$. Thus using this matrix, we find the best possible, dimensionality-reduced representation of the term-document space, see figure 4.3.5.

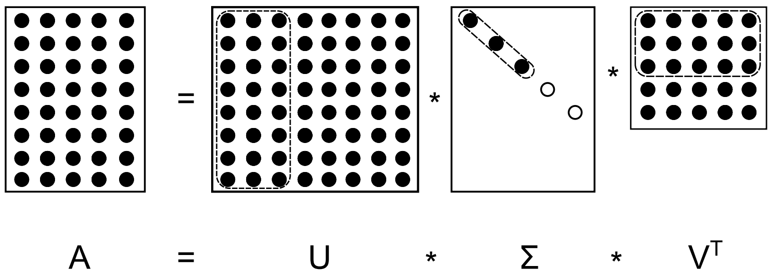


Figure 4.3.5: *Example of the structure for creating a lower-rank approximation of a matrix A. In this case using $k = 3$. Based on the elements marked with dashed lines, we create the lower-rank approximation A_k of A.*

This will be the fundamental concept for analysis in the suggested model. It is a useful representation, that gives a dimensionality-reduced description of the whole document set. This is more easily interpreted by pattern recognition methods [BY09] and thus makes the pattern recognition classifiers function better. Using this methodology Dumais et al. [STD90] suggest many possible ways of comparing different document or term properties, for instance for comparing two documents for their similarity (using the notation in appendix F.1.2) we have: $A_k^T A_k = V\Sigma^2 V^T$, where ^{T} denotes the transpose. Or for comparing two terms and their incidence: $A_k A_k^T = U\Sigma^2 U^T$. This is essentially equivalent to the similarity measures of using a scalar product of two documents as the most appropriate comparison measure. When they are also normalized by their respective lengths this is analogous to the cosine similarity measure which was discussed in section 4.2.10.

4.3.6 Latent semantic indexing

LSI is an information retrieval method suggested by Deerwester in his 1988 paper Improving information retrieval with Latent Semantic Indexing [Dee88]. Latent semantic indexing is closely related to the concepts from latent semantic analysis mentioned in section 4.3.5. In an article by Papadimitrou et al. [CHP00] they suggest a technique of random projection to speed up the process used in LSA for computing the k-rank approximation A_k of the term-document matrix A.

LSA is based on a probabilistic model of the corpus, i.e. the documents being analyzed. To do this one wishes to find a model of the distribution of topics over documents. This concept is also used in BM25 [RZ09] and latent dirichlet analysis [TLG04] to describe the underlying document structure. In the work by Papadimitrou et al. the assumption is a corpus model that combines a probabilistic distribution over four different properties. The four properties being modelled are the set of terms, the set of topics, the set of styles, and the probability distribution of the combinations of topics over all sets of styles.

Using this model and assuming a simple structure of the corpus, for details see [CHP00], it is possible to prove that LSI and LSA can describe the corpus well and that it can handle synonymy. Thus the underlying structure of documents, in that simple case at least, is indeed extractable using the dimensionality reduced representation of the original term-document matrix A.

4.3.7 Naïve Bayes

Using the basic concepts from Bayesian statistics, and employing Bayes' theorem, probabilistic classifiers have a natural place among text retrieval or classification tasks. In these models the use of equation 4.3.7 is the basic building block, where the probability $P(c_k|\vec{d_j})$ is interpreted as the probability that a document d_j randomly belongs to class c_k given its random document vector $\vec{d_j}$:

$$P(c_k|\vec{d_j}) = \frac{P(c_k)P(\vec{d_j}|c_k)}{P(\vec{d_j})} \tag{4.3.7}$$

One natural drawback of the model is the independence assumption that is necessary for Bayes' theorem to hold. In equation 4.3.7 we see that Bayes' theorem applied to documents and classes gives the statement: the probability of the class k multiplied by the probabillity of document vector $j = 1, \ldots, n$ conditional on class k, normalized by the probability of document j. This means that given the independence we get the proper estimate of the probability $P(c_k)$ that the class is k given the probability $P(\vec{d_j})$.

There are some complications to this approach, among others it is very hard to estimate the vector $\vec{d_j}$, since there are a very high number of possible random document vectors. This problem is alleviated when using some algebra and rewriting, see for instance [RZ09]. By assuming a binary independence classifier as in Robertson and Sparck Jones [SER76] we reduce the problem to estimating the probability:

$$P(w_{kj}|c_k) = P(w_{kx} = 1|c_k)^{w_{kx}}(1 - P(w_{kx} = 1|c_k))^{1-w_{kx}} \tag{4.3.8}$$

Using this classifier to describe documents the problem is simplified and reduces to estimating the $2n$ parameters $P(w_{kx}|c_k)$ and $P(w_{kx}|\bar{c}_k), k = 1, \ldots, 2n$ in:

$$\begin{aligned}\log\left(\frac{P(c_k|\vec{d_j})}{1 - P(c_k|\vec{d_j})}\right) = \log\left(\frac{P(c_k|\vec{d_j})}{P(\bar{c}_k|\vec{d_j})}\right) = \\ \underbrace{\frac{P(c_k)}{1 - P(c_k)}}_{\text{constant}} + \sum_{k=1}^{n} \log\left(\frac{1 - P(w_{kx} = 1|c_k)}{1 - P(w_{kx} = 1|\bar{c}_k)}\right) \\ + \sum_{k=1}^{n} w_{kj} \log\left(\frac{P(w_{kx} = 1|c_k)(1 - P(w_{kx} = 1|\bar{c}_k))}{P(w_{kx} = 1|\bar{c}_k)(1 - P(w_{kx} = 1|c_k))}\right)\end{aligned}$$

Thereby the problem of probability estimation is relieved, since these computations are not as complex. Especially there are a significant amount of the $2n$ elements that are 0, and thus the calculation time is further reduced. Indeed one of the best characteristics with the naïve Bayes approach is its efficiency and that it has had some success in certain classficiation tasks, see for instance Yang and Liu [YY99].

4.3.8 Latent Dirichlet allocation

LDA is a generative probabilistic model that models discrete data collections. It was first presented in an article by Blei, Ng and Jordan [Jor03]. The model is built as a hierchical Bayesian model with each word occuring in documents being modelled as a finite mixture over a given set of topics, which the documents can be about. Each topic is then a distribution over topic probabilities, which for text modelling becomes an explicit distribution of topics over documents. Latent Dirichlet Allocation has been used with success for text classification tasks as described in the article by Griffiths [TLG04].

4.4 Issues in creating a classifier

Independent of the choice of classifier for performing automated text classification, there are some underlying problems when creating classifiers. The issues presented below are thus common for all classification tasks, where some are also present in unsupervised classification.

4.4.1 Overfitting

Irrespective of the model chosen there is a chance that when modeling and learning the classifier there will be problems with overfitting of the classifier. This means that the classfier becomes a too complex model of the connections between inputs and outputs. This will induce the model to fit smaller variations that are normally noise, as being parts of the underlying structure that is being modeled. This in turn also leads to that the classifier will perform worse. This problem is similar to that of aliasing, when a curve is fitted (too well) to a given set of data points, see for instance Mitchell's article [DPM88].

4.4.2 Lack of data in different classes

A second issue that may induce problems is that there may be a lack of data for some classes in the corpora. This is a specific problem for text classification according to non-standard allocations where the manual work of allocating documents to classes is time consuming and requires in depth knowledge of the given area. This is the case in the given assignment, and the situation at SKF is no exception. The need for a sufficient number of training documents is a prerequisite for most, if not all, of the classifiers discussed. If there is insufficient data in some classes then the classifier will simply not be able to classify, or generalize, well for that specific class. This problem may be alleviated by searching for documents from that class in the data, and then prelabelling these. This is generally a heavy work, especially in the case of a large corpus of documents.

4.4.3 Uneven distribution of data

A third issue that may present itself is the distribution of data points. In the case that the classifier is to be used on previously unseen documents it needs to be trained to distinguish each of the possible classes. If the distribution of the dataset, and especially the training data, is distributed so that some classes are represented by only one or two documents in a corpora of thousands of documents, the classifier cannot learn to distinguish these documents from others.

This means that when dividing a pre-labelled set of documents into a training set and a test set, there is a higher chance of not having any of the documents at all in the training set from the least represented classes. This also shows some of the problems with the standard approach of dividing data randomly into training and test sets. If this allocation is done randomly, then the experimental performance of the classifier will be dependent on the given distribution, or even worse the classifier may be learned to classify incorrectly, confer section 4.6 on classifier performance. A way to alleviate this problem is to force the learner to include a certain number of training samples from each class, i.e. to enforce class-based random sampling. This can then be extended so that the training sample should consist of randomly chosen documents, but sampled as proportional stratas from each class. Thus the classes of the sample will match the prelabelled portion of documents and correspond to the distribution of classes in the prelabelled set.

This, however, creates the problem of having a biased sample for training. In the case that one forces the randomly chosen documents for the training to be sampled from a certain subset of the distribution this will not only bias the sampling, but also bias the classifier. Since we, by definition, do not know the distribution of the unlabelled data points, this effect may be very harmful on the performance of the classifier. Due to these considerations the approach of using a random allocation to training and test set was used, with multiple restarts to guard against misallocations.

4.4.4 Non-disjoint data spaces

A fourth issue that may present itself is the type of classification used. If a binary classification is used then fully disjoint data classes can be guaranteed. However in the case of a pre-labelled document corpus, represented by a number of (partially) overlapping classes as in the classification task in this thesis, the classification problem becomes harder. Not only is it harder to learn the classifier, but when the data is distributed on the boundary between two groups, or in multiple groups, it is hard to guarantee a proper final classification. The classification problem can then be extended into a multi-labelling problem over multiple classes, or finding the distribution over the set of classes of each document.

This was not the case in the task for this thesis, since the problem was defined as a one-against-all categorization problem, i.e. a binary classification task (and in some cases a multinomial classifier). In this case the classifier is designed to model a disjoint classification paradigm, even if the training data do not come from a perfectly disjoint set of classes, as described in section 3.2.3, [CDM08].
In this case the classifier may become less precise due to the possible multi-classification of a document. The problem of synonymy and polysemy is also a part of this issue, but on the term level, where the terms in the corpus cannot be perfectly distinguished from each other. This in turn creates the same issues as with the multi-topic nature of a document. This was considered and will be discussed further when presenting the suggested framework in chapter 5.

4.5 Pros and cons of different clustering methods

For the type of classification task being pursued in this thesis there are many possible algorithms that can be used. As described in the previous sections one may use vector space models, probabilistic models, similarity based classifiers, etc. For a summary of the basic characteristics of the different investigated models see table 4.5.1.

Model	Pros	Cons
VSM	Simple model, easy to implement, fast classification	Low accuracy
LSI/LSA	Efficient representation (approximation of the term space)	Transparency, distorts original data
kNN/Rocchio	Good accuracy with less training data	Increase in training data decreases efficiency
Naïve Bayes/ Latent Dirichlet	Easy calculation, fast classification	Too simple assumptions, low accuracy
ANN	Approaches a very good classification	Requires a sufficient (larger amount) of training data

Table 4.5.1: Some pros and cons of the different models which were investigated.

This table is of course only a very limited description of the different possible methods. It still gives an indication of the important aspects of the different models and how these compare with each other. Of course the two example-based classifiers have the same issues, as is the case with the probabilistic models.

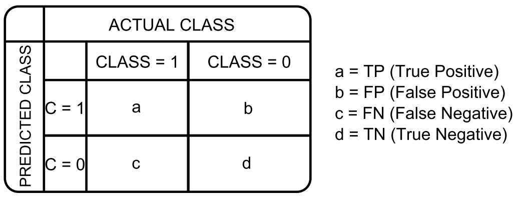
4.6 Performance evaluation

One of the most important parts in creating a classifier is to evaluate its performance. Without a generalizable measure of performance it is not possible to improve upon any classifier or compare different classifiers qualitatively. Therefore we will here present the basic measures for evaluating classifiers, and the intimately connected issues when evaluating a classifiers performance.

4.6.1 Describing the accuracy of a classifier

Confusion matrix

In the case that the classifier is allowed to be computationally heavy to create, and speed is not an issue, the main measure of performance is the number of accurate classifications in the final results. In this context it can be valuable to make use of the confusion matrix, see figure 4.6.1.



		ACTUAL CLASS	
PREDICTED CLASS		CLASS = 1	CLASS = 0
	C = 1	a	b
	C = 0	c	d

a = TP (True Positive)
b = FP (False Positive)
c = FN (False Negative)
d = TN (True Negative)

Figure 4.6.1: *Confusion matrix for a two-class problem.*

The most classic measure of accuracy is the percentage number of actual correct answers: $(a+d)/(a+b+c+d)$. There is however a major caveat with this approach, consider the following example: a problem where 9990 data points belong to class 0, and 10 belong to class 1. If the model predicts everything to be class 0 we will have an accuracy equivalent to 99.9%.
Accuracy is misleading since it does not consider the full extent of the classification problem. The problem in this example is that the classifier does not detect any class 1 example, and never could. A more suitable approach is to consider measures that reward correct classifications and imposes costs on mis-classifications. Some of these approaches are Precision, Recall and F-measure.

4.6.2 Measures of classifier performance

To evaluate classifier performance in a more suitable manner one should consider all aspects of the classification problem. For instance the difference between predicting the right class given that the document belongs to that class (also known as precision). Or the amount of correct classified document, out of the ones classified into the given class (also known as recall). Together these can give a more complete and fair measure of the classifier performance, the *harmonic mean* of precision and recall is known as the F-measure. These different performance measures are defined below.

Precision

The precision of an algorithm (using the notation in figure 4.6.1) is defined as:

$$p = \frac{a}{a+c} \tag{4.6.1}$$

The precision measures how well the classifier classifies documents belonging class one as such. Precision is biased towards the number of positives in the predicted values.

Recall

Recall is defined as (again using the notation in figure 4.6.1):

$$r = \frac{a}{a+b} \tag{4.6.2}$$

Recall measures how well the classifier classifies documents, when they do not belong to that specific class. Recall is biased towards the number of positives in the actual values.

F-measure

A less biased way to measure the performance of a classification algorithm is to use the F-measure, which is defined in terms of p and r as:

$$F = \frac{2pr}{p+r} = \frac{2a}{2a+b+c} \tag{4.6.3}$$

The F-measure is better than both Accuracy, Precision and Recall since it gives a balanced weight to both Precision and Recall. It weights them together so that neither is less important, resulting in the *harmonic mean* between the two. In the case that either precision or recall tend to a smaller value this will give a strong indication in the F-measure. For instance in the case that we have very low values in either p or r this will result in low F-measures, whereas a $p = r = 100\%$ gives a F-measure of $F = 1$. Thus the F-measure is bounded from below at 0 (the worst possible result) and has a maximum of 1, corresponding to a perfect classifier.

4.6.3 Considerations in performance evaluation

There are other aspects which are important to consider when evaluating the performance of an algorithm. Some of the main issues, besides using an appropriate measure such as the F-measure will be described. One of these is the distribution of data into different classes.

The two-class classification problem example in figure 4.6.1 shows that we need to consider the distribution of data in the classes. If there are 9990 examples in class 0 and 10 in class 1 then we simply may have to consider skipping the use of data from class 0 in the training. This gives other considerations such as the appropriate size of training sets as discussed in section 4.4. How many examples that are needed from each class to reach a high-performance classifier is thus dependent on more than the classifier itself or the method used to train it [PNT05]. These types of issues are similar to the ones discussed in section 4.4 and will be elaborated on in chapter 5 where we present the suggested framework. We will present the results from each experiment using precision, recall and the F-measure, since giving all three will give a more complete view of the performance of the given methodology.

4.7 Representation of data for analysis

The data that will be analyzed in each experiment is thus the terms occuring in the whole set of documents in the corpus being analyzed (and classified) for that experiment. The documents are then represented by the term-document vectors. These are normalized using some of the weighting schemes discussed in section

4.2.9. When this weighting has been done, both based on documents and based on terms we have a clean representation of each of the documents in the corpus being analyzed.

This clean representation is then approximated using a k-dimensional approximation to the normalized term-document matrix A. After this approximation has been done, we use the prelabeled documents to train the classifier, which then is used to classify the rest of the approximated document vectors from the corpus. Thus the data that is used as input, both for training the classifier and for classifying the unlabeled documents will be the k-dimensional approximation of the clean term-document vector representation.

We will now turn to describing the suggested framework for automatically allocating documents based on strategic intents.

5 The suggested framework

The task of this thesis was to develop an algorithm for automatically classifying documents into a predefined set of classes. This was done to support the technology intelligence team at SKF in making trend analyses of competitor behavior. To structure the following description we will once again present the general process of creating a classifier for automatically classifying documents. This process can be stated in the following steps:

1. Extract the relevant documents to be classified, together with the prelabeled ones
2. Construct a set of features that are a representation of each document (the attributes)
3. Define a sub-problem of finding the relevant features and constructing the classifier
4. Use the created classifier to classify previously unseen documents
5. Evaluate the accuracy of the classifier using both labeled documents and expert feedback

All these steps taken together define the parts of the suggested framework and will be presented below. Finally we present how the process of the document classification framework will function.

5.1 Data analyzed

The data analyzed for this thesis were sets of documents, consisting of patents and publications from SKF and some of their key competitors in specific business areas. These documents are all publicly available and accessible through the Thomson Innovation database [Tho11a]. This data was downloaded for each experiment set (see section 6.1 for which sets) to be analyzed.

For each of the experiments we combine the relevant prelabeled documents together with the documents that will be classified in the final step. This full set of documents is what has been referred to as the corpus in each of the experiments. This corpus is then the basis for creating the term-document matrix of the terms that appear in each of the documents.

5.2 Preprocessing of data for input

The data was downloaded and accessed based on the requests from the technology intelligence team. It was then cleaned and preprocessed using stemming, stop-words and a clean-up of field values within Thomson Data Analyzer [Tho11b].

For each experiment set, the full set to be analyzed and classified was combined (both labeled and unlabeled examples) and then fed into the TMG toolbox in Matlab [DZ11] developed by Zeimpekis and Gallopoulos at the University of Patras. This is a toolbox for Matlab that contains basic and advanced functionalities for dealing with text categorization. For more information we refer the reader to appendix E.2 and the webpage of TMG which can be found in the references [DZ11].

Using the TMG (text-to-matrix-generator) toolbox we extract the terms from each document, which are fed to the algorithm by means of a raw textfile with the text fields of the abstracts from the documents to be analyzed. This data is then used to create the term-document matrix A, which is the basis for our analysis. The TMG toolbox import allows for usage of many different weighting schemes, but more importantly Matlab has the functionality of making the simple calculations for re-weighting the term-vectors, if this is needed.

The features that we chose to analyze were the words in the document abstracts. We used the term-frequency-inverse-document-frequency (tf-idf) for weighting the term-document vectors. This results in the weight-based term-document matrix, A that is the original representation of the data.

Once the matrix A has been constructed we move on to considering what dimensionality reduced representation will be used for the given set of data. Amongst the different dimensionality reducing measures that could be used we have employed stemming, stop-words, and data cleaning.
To simplify the learning of the classifier we move on to create the k-rank approximation of the matrix A to be used as input to the clustering algorithm, as described in section 4.3.5. The LSA principle described in section 4.3.5 was used to generate this k-rank approximation to the term-document of the full corpus of documents.

This was done to have a systematic way of creating the analyzed term-document representation, both for the prelabeled and unlabeled documents.

The final document representation that was analyzed was thus a k-rank approximation of the original raw term-document matrix, after applying a tf-idf weighting and normalizing the term-document vectors. This in effect results in that the weighting scheme used in our analysis is the tfc scheme, as described in section 4.2.9.

5.3 Classification algorithm

The classification algorithm to be used depends on the input data. As described in the previous section we use the tf-idf weighting with normalized TDV and then compute the k-rank approximation of TDM describing the corpus. Thereafter we also apply the k-rank approximation from LSA on the original tf-idf weighted matrix to generate our input for analysis.

The choice of clustering algorithm was a balanced decision between functionality, performance, and ease of implementation. From the discussion regarding pros and cons in section 4.5 we know that there are some better and some less good aspects of each algorithm.
Among the different methodologies available the artificial neural network approach was chosen. Partially because it easily handles high-dimensional input and gives a good output. Another reason for this choice is that the algorithm approaches expert judgements with a very large number of iterations, as described in the article by Trappey et al. [AJCT06].

Therefore the clustering algorithm that we chose for training using the given k-rank approximation of the corpus was the ANN using LSA and the tf-idf weighting scheme on the cleaned data. The algorithm that was chosen was the back-propagation neural network algorithm. We now give some details of the employed algorithm for the interested reader.

5.3.1 The back-propagation neural network algorithm

A *back-propagation neural network* (BPNN) is a generalized version of a multi-layer feedforward neural network. More precisely it is a learning method that minimizes the mean squared error of the output produced by the network. This error is evaluated by comparing against some prespecified target values, which in our case are the set of classes (either 10 different for the multi-labelling case or 2 different for the binary labelling case) of the documents analyzed. First we will give a short introduction of the network details and then we will proceed with the specific implementation of the algorithm.

An artificial neural network consists of a number of layers. Each layer has a number of neurons and each neuron represents an output value from the given neuron. Each neuron may contain a simple, or complex, linear, or non-linear, transfer function which transforms the inputs coming in to the neuron to the output from the same. This mapping is done according to the given transfer function. For a general description of an ANN with an input layer, 1 hidden layer and an output layer, see figure 5.3.1.

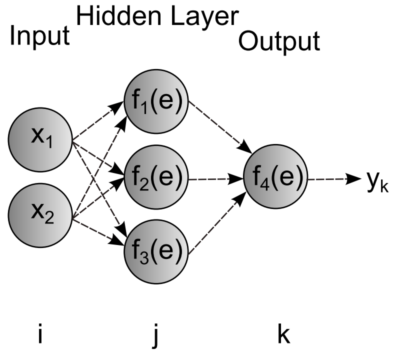
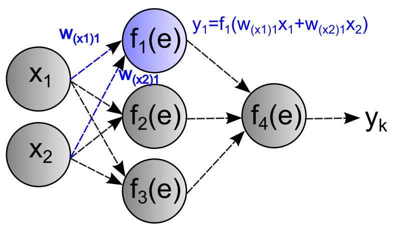
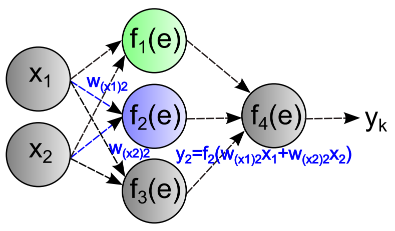


Figure 5.3.1: *An example ANN with* $i = 2$ *neurons in the input layer,* $j = 3$ *neurons in the hidden layer and* $k = 1$ *neuron in the output layer.*

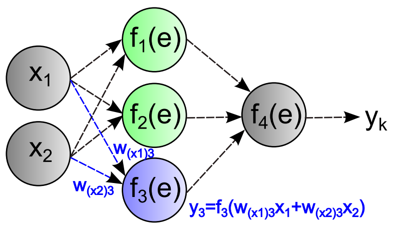
In this case the input layer consists of a vector $\mathbf{x}$ of two values $\mathbf{x} = (x_1, x_2)$ corresponding to $i = 1, 2$ input neurons. Each of these values should be in the range 0 to 1, i.e. $0 \leq x_1 \leq 1$ and $0 \leq x_2 \leq 1$ [CDM08]. These inputs are then propagated forward in the network based on weights associated with each neuron in the next layer, see figure 5.3.2a, 5.3.2b and 5.3.2c. The weights are initialized randomly for each of the steps in the BPNN method.



(a) *Feed forward for neuron 1*



(b) *Feed forward for neuron 2*

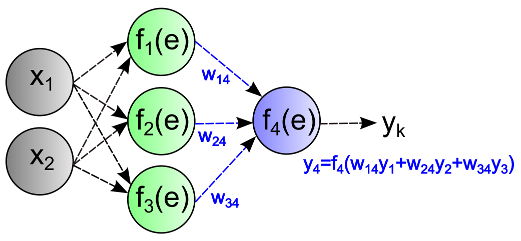


(c) *Feed forward for neuron 3*

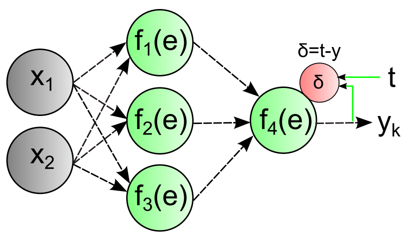
Figure 5.3.2: *The feed forward step in the ANN*

Each of the $j = 1, 2, 3$ neurons in the hidden layer then evaluates the sum of the weighted inputs according to the transfer function in the neuron (in this example f_1, f_2, f_3). These functions are chosen to be the same function, but could also be chosen as different transfer functions. An example transfer function is presented below after this description. Usually the transfer function in each neuron is a monotonically increasing function of the input value e, that starts at -1 and grows to $+1$ with a value of 0 at $e = 0$. This is based on the desired behaviour of the given transfer function, and many possible functions exist. For a thorough investigation on different transfer functions we refer the reader to chapter 7 in the book by Rojas [Roj].

The final value y_k ($k = 1$) is then calculated in the same manner. In the output layer the transfer function (f_4) is usually a weighted sum of the inputs (a simple addition), see figure 5.3.3a. In other cases however it can be chosen to have a sigmoid output or any similar transfer function, as in the case of the neurons in the hidden layer. The resulting value is then compared to the target value for the training sample and the error δ at the output layer is calculated, see figure 5.3.3b.



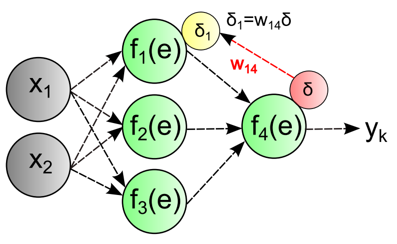
(a) *Resulting value at neuron 4*



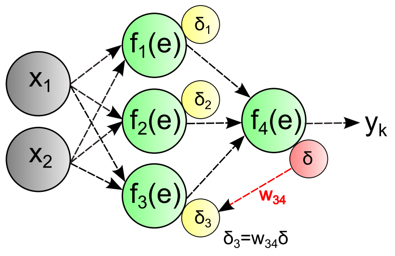
(b) *Error calculation and feedback based on the target values*

Figure 5.3.3: *The resulting value at neuron 4 in the ANN, and error calculation.*

When the value of the error has been calculated we can start to *back-propagate* the error to the preceeding layers. This is done by propagating the error, weighted according to the same feedforward weights used previously, to each neuron in the preceeding layers, see figure 5.3.4a and 5.3.4b.



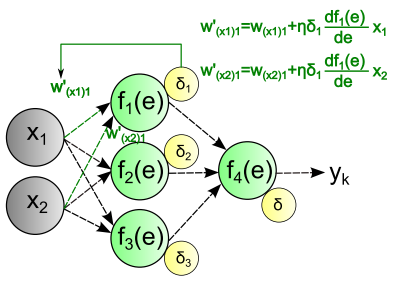
(a) *Feedback step for neuron 1*



(b) *Feedback step for neuron 3*

Figure 5.3.4: *The feedback step for the neurons in the hidden layer (neuron 2 is calculated analogously whereby it is omitted).*

When finally all possible back-propagated errors have been calculated the next step is the correction step of the algorithm. In this step we correct the feedforward weights according to the linear approximation of the previous weights, based on the transfer function in the corresponding neuron, see figure 5.3.5a. This is done for all weights, until all corrected weights between the input layer and the first hidden layer have been calculated.


$w'_{(x1)1}=w_{(x1)1}+\eta\delta_1\frac{df_1(e)}{de}x_1$
$w'_{(x2)1}=w_{(x2)1}+\eta\delta_1\frac{df_1(e)}{de}x_2$
w'(x1)1
w'(x2)1
x1
x2
f1(e)
f2(e)
f3(e)
f4(e)
yk
δ1
δ2
δ3
δ
(a) *Weight correction of input weights for neuron 1*

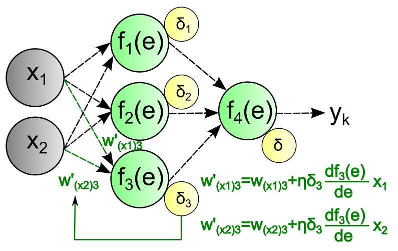
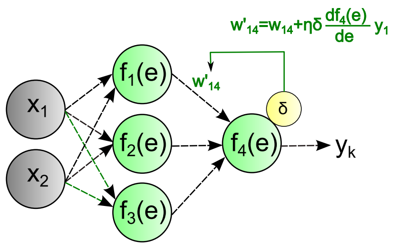
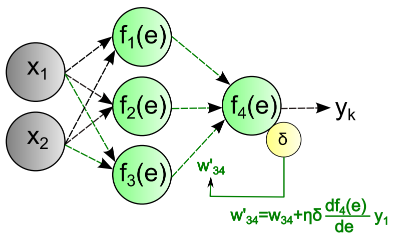

x1
x2
f1(e)
f2(e)
f3(e)
f4(e)
yk
δ1
δ2
δ
δ3
w'(x1)3
w'(x2)3
$w'_{(x1)3}=w_{(x1)3}+\eta\delta_3\frac{df_3(e)}{de}x_1$
$w'_{(x2)3}=w_{(x2)3}+\eta\delta_3\frac{df_3(e)}{de}x_2$
(b) *Weight correction of input weights for neuron 3*

Figure 5.3.5: *The final weight correction for the neurons in the hidden layer (the weights for neuron 2 are calculated analogously whereby they are omitted).*

When the corrected weights between the first and second layer have been calculated the algorithm again continues forward through the network. This is called the weight correcting forward pass. This correcting step is then continued forward in the network, calculating the corrected weigths at each layer. This is repeated until the final output layer has been reached. In our example we only have one hidden layer, so the weight correcting forward pass only needs to calculate the weights between the input and the hidden layer, and the weights between the hidden and the output layer.



(a) *Weight correction of input weights for neuron 4*



(b) *Weight correction of input weights for neuron 4*

Figure 5.3.6: *The final weight correction for the neurons in the output layer. This concludes* one iteration *in the back-propagation neural network model*

In the correction pass, the parameter η is called the learning speed. This can be changed to influence the amount of change that the derivative should have on the weights. In our implementation we simply set $\eta = 1$

and use normal standard-paced learning without any speed-up. This is due to the possible problems a sped up approach might run into. When the weight correcting forward pass has been completed, see figure 5.3.6a and 5.3.6b, we have completed the first iteration of the back-propagation neural network algorithm.

The algorithm is repeated a certain number of iterations, a specific duration of time, or until some form of convergence criterion has been reached. The last of these is the most common framework for concluding the algorithm, and usually the *mean squared error* (MSE) is used to evaluate when the ANN has reached a sufficient level of accuracy. MSE will be discussed and described below, together with the implementation specific details.

Example transfer functions

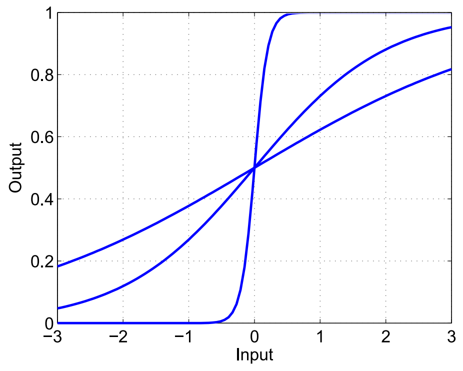
Some example transfer functions are the sigmoid transfer function $s(x)$ (figure 5.3.7a) that has a parameter c and the variable x as input:

$$s(x) = \frac{1}{1 + \exp(-cx)} \tag{5.3.1}$$

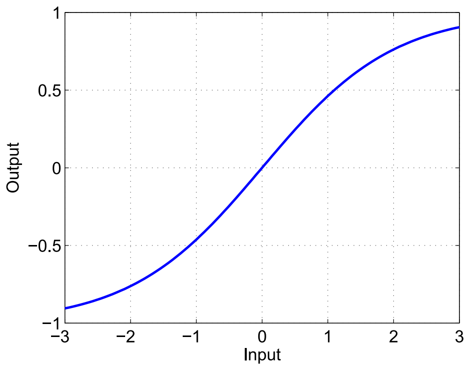
where a higher value of c gives a sharper increase in function value around the point of symmetry ($x = 0$), see figure 5.3.7a. This function has the drawback that it never outputs a negative value. For a more versatile description of the input-to-output allocation one can employ a output-symmetrical version of the sigmoid $S(x)$:

$$S(x) = \frac{1 - \exp(-x)}{1 + \exp(-x)} \tag{5.3.2}$$

which then evaluates negative input values as negative outputs. This function can of course also be appended with the scaling parameter c as in the sigmoid activation function, if one wishes to have a more direct response.



(a) *Sigmoid transfer function with 3 different values of c*



(b) *Symmetrical sigmoid transfer function*

Figure 5.3.7: *Some example transfer functions*

The specific implementation

Looking at figure 5.3.1 we describe the network as the following. The network takes inputs placed on the input neurons, and multiplies each by a weight (w_{ij}) between the *input layer* and the *hidden layer*. Thus each node j in the hidden layer gets the following input $f_{in}(j)$

$$f_{in}(j) = b_j + \sum_{i=1}^{n} w_{ij} x_i \tag{5.3.3}$$

where b_j is a bias function that induces a bias on that node. These inputs $f_{in}(j)$ are equivalent to the value e in figure 5.3.1, and represent the inputs to neurons 1, 2 and 3 in the hidden layer. These values are then transfered in the hidden layer neurons according to the tan-sigmoid activation function:

$$g_{trans}(x) = tanh(x) = \frac{exp(x) - exp(-x)}{exp(x) + exp(-x)} \tag{5.3.4}$$

which serves as an activation function. Thus in our implementation the functions f_1, f_2 and f_3 in figure 5.3.1 are g_{trans}. This transfer function saturates the input when the sum of inputs to that node is sufficiently high, producing an output from that node in the hidden layer. The reason for using the tansig transfer function is that it easily deals with negative inputs, and since our k-rank approximation of the term-document matrix may produce negative entries, this was chosen as the transfer function.

When all the outputs from the hidden layer have been calculated they are fed into the *output layer.* There is another (different) bias (c_k) and weights (w_{jk}) between the hidden layer and the output layer. Finally the resulting outputs at node k in the output layer become:

$$f_{out}(k) = y_k = c_k + \sum_{j=1}^{M} w_{jk} x_i \tag{5.3.5}$$

Thus the purpose of the BPNN is to minimize the error when comparing the outputs y_k with the targets t_k which are the classes it should specify documents into, see figure 5.3.3b. Using this notation the error (denoted as δ in figure 5.3.3b, 5.3.4a, 5.3.4b, 5.3.5a, 5.3.5b, 5.3.6a, 5.3.6b) at each output neuron k can be denoted as:

$$e_k = y_k - t_k \qquad MSE = \frac{1}{M} \sum_{k=1}^{M} e_k^2 \tag{5.3.6}$$

where *MSE* denotes the mean square error. In our implementation the BPNN is trained using a *scaled conjugate gradient* (SCG) or the *Levenberg-Marquardt method* to minimize the *MSE* for the whole network. For an explanation of these methods, see appendix F.2.2 and F.2.3.

Network outputs and class allocations

For most typical cases the outputs in pattern recognition are a set of binary values, 0 or 1, for the case that the document is in the respective class, or not, see equation 5.3.7.

$$\left[\begin{array}{c|cccccc} \text{Doc.} & 1 & 2 & 3 & \dots & n \\ \hline \in A & 1 & 1 & 0 & \dots & 1 \\ \notin A & 0 & 0 & 1 & \dots & 0 \end{array}\right] \tag{5.3.7}$$

In the case that we have a multi-class classifier then we will have the corresponding number of possible output neurons (in the binary case $k = 2$ in out setting the multi-class case has $k = 10$), see equation 5.3.8.

$$\left[\begin{array}{c|ccccccccccc} \text{Doc.} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & \dots & n \\ \hline \in A & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ \in B & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ \in C & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ \in D & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \dots & 0 \\ \in E & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \dots & 0 \\ \in F & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ \in G & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 1 \\ \in H & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \dots & 0 \\ \in I & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \dots & 0 \\ \in J & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \end{array}\right] \tag{5.3.8}$$

5.4 Evaluating classifier performance

The different measures of performance that will be used are described in section 4.6. The different performance measures precision, recall and the F-measure will be calculated for each of the binary classifiers. In the multinomial allocation case we will make use of the same measures for the results for each class, and combine these into a mean average precision, recall and F-measure. In the case of evaluating across different sets, we will use the mean, best and worst for each of the three measures as results.

5.5 Post-experiment expert validation

After the classification of documents has been done, on a larger set of data, the resulting assigned classes will be evaluated. For each of the binary classifiers we will take a random sample of about 20% of the automatically labeled documents, to evaluate the performance of the algorithm. The reason for choosing a smaller subset is simply that timewise the validation would take too long.
This sample of the results will be given to a domain expert for validation of the results, based on their knowledge of the specific strategic intent. Based on their classification a different evaluation of the classifier performance will be done. This evaluation will concern itself with the performance of allocating previously unseen documents, and the resulting performance is based on the expert evaluations.

5.6 The suggested process

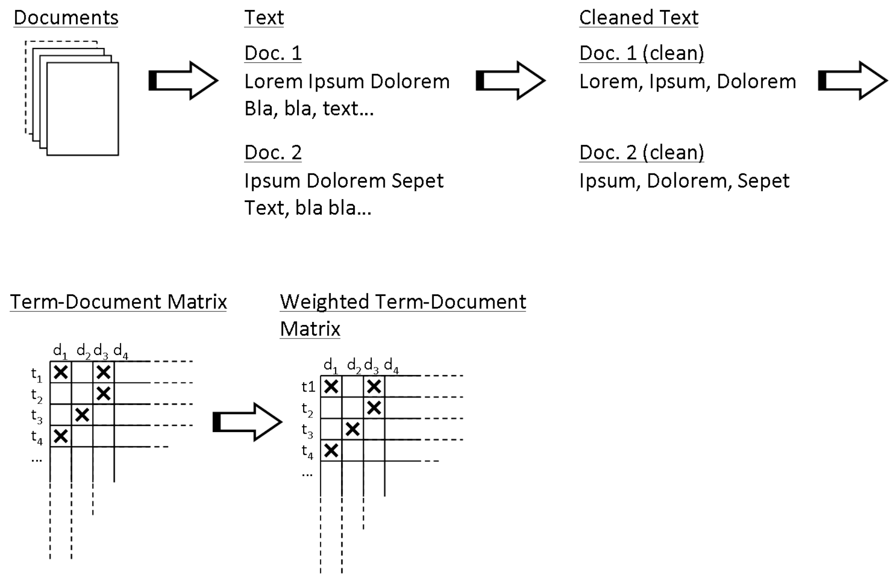


Figure 5.6.1: *Suggested process, part 1.*

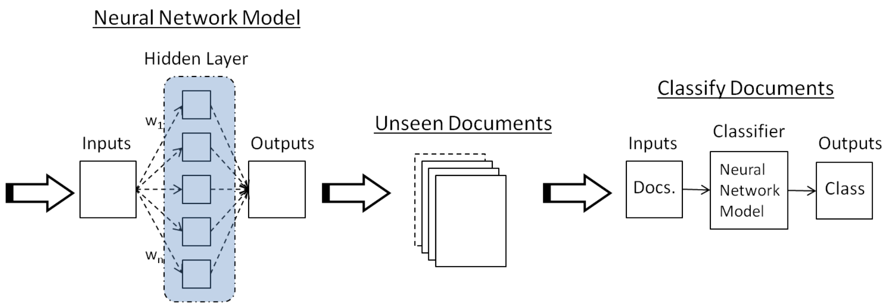


Figure 5.6.2: *Suggested process, part 2.*

6 Experimental setup

The thesis will look into a number of possible experimental setups for assessing the effectiveness of the suggested framework. More importantly the aim is to assess whether the framework could perform well for automated document classification in the way intended at the technology intelligence department of SKF.

The main questions to be answered by the experiments are:

- Can the suggested framework multi-label documents succesfully?
- Can the suggested framework binary-label documents succesfully?
- Are there significant differences in results when using a more limited number of training documents, and if so how few are too few?
- Can the suggested framework perform better for one specific strategic intent, when the number of prelabelled documents is larger?

The different experiments are going to give an indication as to whether there is a significant difference in performance of the classification algorithm, when adding or removing different sources of training data and when analyzing different types of classification tasks.

6.1 Data Sets Used for Experiments

The different sets of training, testing, and basic data were the following (censored, encoded as per request from SKF):

a) Prelabeled training set with multiple strategic intents consisting of all SKFs publications

b) Prelabeled training set with multiple strategic intents consisting of all of competitor A's publications

c) Prelabeled training set with multiple strategic intents consisting of all of competitor B's publications

d) Prelabeled training set consisting of all patents from SKF within strategic intent E

e) Prelabeled training set consisting of all patents from competitor C within strategic intent E

f) Prelabeled training set consisting of all patents from competitor D within strategic intent E

g) Prelabeled training set consisting of all patents from competitor E within strategic intent E

h) Unlabeled set of patents from SKF

i) Unlabeled set of patents from competitor A

j) Unlabeled set of patents from competitor B

The experimental setup will be based on a two-step approach. The first step will be to train the ANN for identifying the given set of data, either in the binary case or in the multinomial case (if applicable). The next step is to evaluate the performance of the network based on the prelabeled documents. To do this the algorithm splits the whole prelabeled set into a training part and a testing part. In the suggested algorithm we randomly assign 70% of the prelabeled documents for training and use the remaining 30% for testing the created network. This evaluates the performance of the ANN based on unseen (but classified) documents, to validate the performance of the ANN. This approach is used in both the multiclass and the binary classification case, and finally it is applied to a broader set of documents, mixing both publications and patents, some of which are labeled.
This will give some indications how the algorithm is performing based on the training set used and give an indication of how well we can expect the algorithm to perform for specific strategic intents in comparison with the full set. When analyzing the full set we will set up the term-document matrix and the subsequent analyses by first parsing all of the corpus that is to be analyzed later. This is needed to have a full representation of the term-space before any approximation or training is done.

6.2 Performed Experiments

The eight experiments that will be performed based on the data listed above are the following:

1) Using set a) to train a multiclass ANN classifier, then evaluate the performance of this network. If performance is acceptible apply it to set h), i) and set j) to evaluate the distribution of the portfolios of SKF (set h)), competitor A (set i)) and competitor B (set j)) over time, for each of the strategic intents.

2) Using set a) to train 10 different binary ANN classifiers to classify if a document belongs to a single strategic intent or not. Then evaluate the performance of this network, and finally if performance is acceptible apply it to set h), i), and j) to evaluate the distribution of these competitors portfolio over time, for each of the strategic intents.

3) Using set b) to train a multiclass ANN classifier, then evaluate the performance of this network. If performance is acceptible apply it to set h), i) and set j) to evaluate the distribution of the portfolios of SKF (set h)), competitor A (set i)) and competitor B (set j)) over time, for each of the strategic intents.

4) Using set b) to train 10 different binary ANN classifiers to classify if a document belongs to a single strategic intent or not. Then evaluate the performance of this network, and finally if performance is acceptible apply it to set d), e), and f) to evaluate the distribution of these competitors portfolio over time, for each of the strategic intents.

5) Using set c) to train a multiclass ANN classifier, then evaluate the performance of this network. If performance is acceptible apply it to set h), i) and set j) to evaluate the distribution of the portfolios of SKF (set h)), competitor A (set i)) and competitor B (set j)) over time, for each of the strategic intents.

6) Using set c) to train 10 different binary ANN classifiers to classify if a document belongs to a single strategic intent or not. Then evaluate the performance of this network, and finally if performance is acceptible apply it to set h), i), and j) to evaluate the distribution of these competitors portfolio over time, for each of the strategic intents.

7) Using set a), b) and c) to train a multiclass ANN classifier to classify. Then evaluate the performance of this network, and finally if it performance is acceptible apply it to set h), i), and set j) to evaluate the distribution of SKF (set h)), competitor A (set i)) and competitor B (set j)) portfolio over time, for each of the strategic intents.

8) Using set a), b) and c) to train 10 different binary ANN classifiers to classify if a document belongs to a single strategic intent or not. Then evaluate the performance of this network, and finally if performance is acceptible apply it to set h), i), and j) to evaluate the distribution of these competitors portfolio over time, for each of the strategic intents.

When the experiments have been done the first publication date of each source document from each competitor will be used to create a chart of the distribution of strategic intents for that competitor over time. This will also be the final results presented from the thesis.

6.3 Data features used

The features that were used for training the algorithm and analysing the documents was the process described in chapter 5. We preprocess the data, and then construct the term-document matrix, based on the tfc weighting scheme. After this we apply the k-rank approximation to the matrix using LSI to get the final training matrix. From this matrix we take all labeled documents, and use as input to the ANN classifier, to generate the results. In each experiment we will test different number of neurons in the hidden layer. Preliminary results give some indication that a number closer to the number k from the k-rank approximation may function better. Finally the ANN is trained using these, and then evaluated based on the part of the prelabeled data that was not used.

6.4 Reporting Results

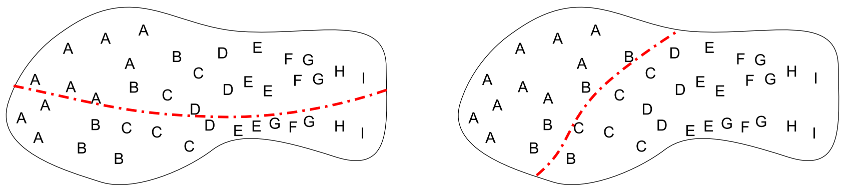
The results will be reported based on performance, and if performance was good enough we will also give the resulting temporal distributions of strategic intents for the competitors. The performance measures described in section 4.6 will be used for each experiment i.e. Precision, Recall and the F-measure. If the performance is good enough, we will also report the results for the broader analysis of non-labeled sets (set d), e), and f)).

7 Results & analysis

In this section we will present the results from the experiments that were performed. We will start by presenting the data that was analyzed and will then move on to the results using the suggested methodology. Based on the findings some modifications will be made to the framework, to attempt to improve the performance of the ANN. Finally we will present the summary of the results for the different datasets, and display the temporal distribution of the resulting classifications.

7.1 Data

The data sets that were studied was divided into a number of sets for analysis. These sets were all preprocessed in the same way. The first is to remove empty documents, the second is to remove documents that are not in English, after this all documents are organized sorted into coherent subsets from each source of information. After this was done the documents where parsed for creating the document-term matrix representing the collection being analyzed, and subsequently it was weighted according to the specified weighting scheme.
When the final TDM is ready, we apply the SVD k-rank approximation of the matrix to generate the final matrix representation of the set, that will be used for the analysis and the training of the ANN text classifier. After the matrix has been generated a number of runs of simulations are done to ensure a good ”starting division of documents”, see figure 7.1.1a. Since we randomly allocate documents to the different test and training sets there is a chance that the training set that has been generated does not contain any elements from the group which we are trying to train it to recognize, see figure 7.1.1b. This clearly creates a problem when creating the classifier, since it cannot recognize any elements it has not seen before.



(a) *A ”good” split into training and test* (b) *A ”bad” split into training and test*

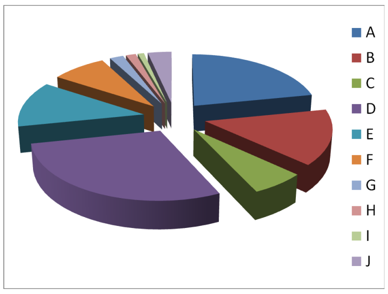
Figure 7.1.1: *The distribution of SI for two different splits of the datasets across all SI*

As described in the figures the importance of a ”well representing” document set split is not a trivial task, and this may have strong effects on the resulting classifier accuracy. For instance any classifier built on the split in figure 7.1.1b will not be able to accurately classify classes C through J if the left part is used for training, and vice versa for classes A and B if the right half of the split is used. Thereby it is clear that the imporance is to get a good and ”representative” split of the dataset, and preferably the set that is to be classified, before creating the classifier. This is however not possible to control or correct for, which is why multiple runs were used to assure that the probability of a ”good split” would be increased.

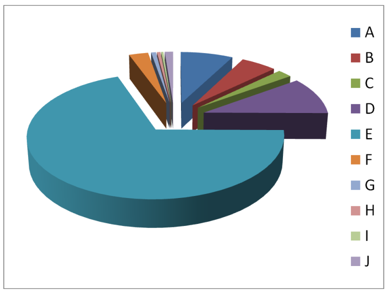
7.1.1 Data distribution and classifier evaluation

An important concern when evaluating the performance of the classification algorithm is to consider the training data distribution. This is important due to the same considerations as the ones raised in section 4.6. There we showed that a trivial classification that classifies all documents as not belonging to class X can yield a 99.9% accuracy in the algorithm, given that class X only represents 0.1% of the data. This type of caveat is important to consider since we do not want a biased result due to any controllable factor within the experiments.

To give some consideration of these aspects we will here present the distribution of classes in the different prelabelled data sets that were considered.



(a) *sets a)-c)*



(b) *All sets a)-g)*

Figure 7.1.2: *The distribution across SI for the two different sets across all SI, data from table 7.1.1*

Set	# Docs.	Class A [%]	B [%]	C [%]	D [%]	E [%]	F [%]	G [%]	H [%]	I [%]	J [%]
a)	316	19	16	7	20	16	11	3	2	0 (0.3)	5
b)	44	20	27	0	30	2	20	0	0	0	0
c)	156	27	10	6	44	9	0	1	1	2	1
a)-c)	516	22	15	6	29	12	8	2	1	1	3
d)	205	0	0	0	0	100	0	0	0	0	0
e)	361	0	0	0	0	100	0	0	0	0	0
f)	351	0	0	0	0	100	0	0	0	0	0
g)	102	0	0	0	0	100	0	0	0	0	0
d)-g)	1019	0	0	0	0	100	0	0	0	0	0
All	1535	8	5	2	10	69	3	1	1	0 (0.3)	1

Table 7.1.1: Percentage distribution of SI over the different analyzed data sets. Percentages may sum to more than 100% due to rounding.

From table 7.1.1 we see that the distribution over the different SI differs strongly between the data sets. Since we analyze set a), b) and c) both separately and together, we see that there is a strong bias towards SI A)-E). For the case of the SI E specific documents, there is a large number of documents available. This should be possible to use to create a fairly well performing classifier for class E. Furthermore the other documents in sets a)-c) can also contribute to a SI E specific classifier. For a more visual description of the distributions of documents, see figure 7.1.2, corresponding to sets a)-c) and All as described in table 7.1.1.

One important distinction is that the documents are of two different types, set a)-c) are publications, while d)-g) are patents. This may influence their compatibility in using the documents for automatic categorization. Based on the discussions regarding the different types of data in section 3.3 we expect differing formats in the two different types of documents. To analyze this performance, we will test to successively add a number of the prelabelled SI E patents to the a)-c) set of publications to evaluate the improvement (if any) for creating a binary classifier for SI E.
This means that the only classifiers that may be sufficiently analyzed and trained (in a multi-classifier setting) will be the classifiers for the SI A-F. This will be true unless there are strongly deviating characteristics for the other classes (causing the very few documents to strongly influence the classifier training). We will assume this is not the case, since we are only dealing with technical data in this application. Thus the results will be limited to the main groups of SI A-F. Also the group of companies are from a fairly coherent business area, being direct competitors to SKF. This means that we can assume that the publications should, in some respect, consist of similar term-document representations.

Another important distinction that one may make between the sets is the following. For creating a classifier for SI E) one can choose to either use only sets d)-g), or one can choose to use all the documents in the full set, a)-g), or finally only use set a)-c). These three options give somewhat different aspects of the analysis. For creating a classifier only for that class it is important to remember that if all documents come from one class, the algorithm has nothing to distinguish them from any other class when creating the classifier. This means that the classifier will consistently overclassify documents into that class, since no other characteristics are analyzed. On the other hand, if a too broad set is used for creating a classifier for, lets say, class E, this will result in poorer performance as well since there is insufficient data for the classifier to learn from.

With these things in mind, we attempt to create the classifiers, either in binary or multi-class form, to establish if there are qualitative aspects that may be found regarding the suggested framework. Since we are interested in evaluating the methodology rather than finding the best possible classifier, due to the fact that this is a fairly comprehensive task. To evaluate classifier performance there are many measures that may be used, but for consistency we need to have a common measure for evaluating the performance of the classifier. This finally results in that we will consistently use the F-measure as the "real" measure for comparsion between the different classifiers. So when we will express that one classifier is better than another, we refer to that the previous has a higher F-measure.

7.1.2 Details for each dataset

In this section we give some of the general characteristics of each of the 7 datasets that were considered for the thesis work. For each dataset we present the key characteristics describing that set, and finally we state a summary of all the datasets, when combined as one complete.

Set	# Docs.	# Terms	Ave. # Terms/Doc.	Ave. # Indexing Terms/Doc.	TDM Sparsity
a)+h)	2178	8729	122.573	55.4807	0.391626%
b)+i)	1531	5629	121.496	53.8125	0.583244%
c)+j)	849	5314	144.159	72.4417	0.799032%

Table 7.1.2: Properties of the different analyzed data sets. The qualitative properties of the sets are similar across the main sets analyzed.

7.2 Results using suggested methodology

In the following section we present the primary results of each experiment performed, in chronological order. The full results are presented in appendix A.1. The separation of the results is partially to include the main findings, but also to reflect the main results here, and the full results of the experiments in the mentioned appendix. This will give the reader a insight into the process of evaluating the performance of the framework. Furthermore it will give the process of zooming in on the topic of interest, which was the best implementation, using the suggested framework.

7.2.1 SKF publications

The first experiment that was carried out was an analysis using only a set of 316 prelabelled SKF patents. The suggested framework was applied for the complete dataset. The first classifier was a multi-class neural network (with 10 classes) which was trained to allocate the documents according to the 10 relevant SI.

This experiment however yielded very poor results, which is why these will not be elaborated on further. Instead the focus was shifted to making use of binary (one-against-all) classifiers. As discussed above this classifier classifies the documents as being of class X or not. Therefore it is only useful for distinguishing for one class, and a separate binary classifier must be produced for classifying each class.
Given this situation 10 binary classifiers were created. One for class A, one for class B, etc.. The performance of the classifiers is heavily dependent on the original distribution of the training and test documents. Our 95%-5% split for each SI classifier gives a higher probability of having the relevant class (X) in the training set. This is since we will most probably succeed in having a random split that generates a "good" split, as described in figure 7.1.1.

However even though we use a 95%-5% split there is a high probability that for the smallest classes we will get poor results. Therefore 10 different runs were performed for creating each binary classifier (where these were evalulated based on their F-measure, precision and recall performance). The first results for this dataset indicated that approximately 100-200 neurons in the hidden layer were sufficient for increasing performance. Using a hidden layer with 175 neurons and restarting 10 times for creating the classifier the following resulting F-measure, precision and recall values were the best performance for the algorithm:

Binary Classifier	Precision	Recall	F-measure
$\in A$ vs. $\notin A$	0.8	0.85246	**0.8254**
$\in B$ vs. $\notin B$	0.88235	0.88235	**0.88235**
$\in C$ vs. $\notin C$	0.81818	0.81818	**0.81818**
$\in D$ vs. $\notin D$	0.84375	0.87097	**0.85714**
$\in E$ vs. $\notin E$	0.81034	0.95918	**0.8785**
$\in F$ vs. $\notin F$	1	0.77143	**0.87097**
$\in G$ vs. $\notin G$	0.76923	1	**0.86957**
$\in H$ vs. $\notin H$	1	0.85714	**0.92308**
$\in I$ vs. $\notin I$	1	1	**1**
$\in J$ vs. $\notin J$	0.8	0.88889	**0.84211**

Table 7.2.1: Results when analyzing prelabelled publications from only SKF (set a)). Best performance (measured by precision, recall and F-measure) out of 10 random restarts for each classifier using 2000 iterations.

As can be seen in table 7.2.1 very high values for both precision and recall are attained. This of course yields very high F-measures as well, meaning that the overall classifier performance is good. A limitation of this approach is of course that the given experiments and tests are performed on prelabelled data. For evaluation purposes this is the only way to evaluate the performance of the classifier, but for the resulting classification we will make use of expert validation to ensure the performance of the algorithm.

In this experiment only SKF publications were used to train and test the classifiers, which leads to the question of how the suggested framework performs for a more mixed dataset. This was the focus of the next experiment discussed in section 7.2.4, where we apply the suggested framework to the set a)-c), i.e. for SKF publications combined with the publications from competitor A and competitor B.

7.2.2 Competitor A publications

When studying the results for the set of publications by competitor A, the results differ strongly. In this case we only used approximately 5000 iterations, but the difference in iterations did not account for a major difference in performance. In this case we run the experiments using only the mentioned set and the results presented in table A.1.1 should qualitatively be compared to the results in table 7.2.1. However note that in this experiment a 70% − 30% split was used instead of a 95% split. We choose to present the best, worst, mean and median values for the precision, recall and F-measure, to give an indication of the performance across the 10 random starts.

Bin. Class. (#**)	Pre. (M)	(B)	(W)	Rec. (M)	(B)	(W)	F (M)	(B)	(W)
$\in A$ vs. $\notin A$ **(200)**	0.62979	1	0.51351	0.61739	0.91304	0.13043	0.57196	**0.70175**	0.23077
$\in B$ vs. $\notin B$ **(200)**	0.55287	0.625	0.48387	0.66667	0.95238	0.2381	0.57866	**0.69091**	0.33333
$\in C$ vs. $\notin C$ **(20)**	0.57699	0.64706	0.54839	0.69545	0.90909	0.5	0.62303	**0.70175**	0.52381
$\in D$ vs. $\notin D$ **(150)**	0.49358	0.6	0	0.60909	0.90909	0	0.58481	**0.7037**	0.2069

Table 7.2.2: Primary results when analyzing prelabelled publications from only competitor A (set b)). Best performance (measured by precision, recall and F-measure) out of 10 random restarts for each classifier using 500 iterations.

As can be seen we have a strong deviation in performance for the 4 SI that have been analyzed in this experiment. In order of magnitude the differences in F-measure are about 0.10-0.15, which in this context is a large difference in performance. We move on by considering the publications from competitor B.

7.2.3 Competitor B publications

When analyzing the publications from competitor B we find that in this case the results deviate even more than in the case of competitor A. For this situation the resulting performance (once again using a 70% − 30% split of the data and 5000 iterations) is even poorer than for competitor A's publications. This indicates that the performance is not only dependent on number of iterations and split sizes but also on the source of the data. We choose to present the best, worst, mean and median values for the precision, recall and F-measure, to give an indication of the performance across the 10 random starts.

Bin. Class. (#**)	Pre. (M)	(B)	(W)	Rec. (M)	(B)	(W)	F (M)	(B)	(W)
$\in A$ vs. $\notin A$ **(200)**	0.61771	0.69388	0.5463	0.46947	0.65263	0.25263	0.52119	**0.61084**	0.36923
$\in B$ vs. $\notin B$ **(20)**	0.58268	0.66667	0.53488	0.47684	0.77895	0.33684	0.51451	**0.64069**	0.44755
$\in C$ vs. $\notin C$ **(150)**	0.5412	0.58333	0.49432	0.58947	0.91579	0.36842	0.54658	**0.64207**	0.45161
$\in D$ vs. $\notin D$ **(20)**	0.54368	0.58824	0.51351	0.57579	0.89474	0.14737	0.53614	**0.66667**	0.22951

Table 7.2.3: Primary results when analyzing prelabelled publications from only competitor B (set c)). Best performance (measured by precision, recall and F-measure) out of 10 random restarts for each classifier using 500 iterations.

In table A.1.2 we have the final results when analyzing only competitor B's publications. A strong decrease in performance compared with the first and second experiment can be seen, with differeneces in F-measure of magnitude 0.20-0.25. This is a fairly large deviation, and the next step was to check the performance when mixing the documents, and when applying stemming to the document data sets.

7.2.4 Results for set a)-c) with stemming

Based on the high performance in the experiment using only SKF publications, the expectation is that increasing the number of training samples should improve upon the performance of the classifiers. The dataset considered in this experiment was the combination of set a), b) and c). This was the total number of prelabelled documents from all three sets of publications. These sets were analyzed according to the framework using stemming, stop-words and normalization to generate the TDM.

Once the TDM was generated we created a number of multi-class classifiers for the 10-class case. As in the previous experiment the results were poor, whereby we chose to once again focus on creating the best performing one-against-all binary classifier given the data.

Using the suggested framework we prepared the data and ran initial tests for all SI with very poor performance, with F-measures ranging from 0.1 to 0.6, but never higher. Further experiments using a higher number of neurons in the hidden layer also did not contribute to improved performance. Attempts were also made to increase the number of back-propagation iterations (from 500 iterations as before to 2500 iterations), but with little results. Thus the focus was to test the performance of the binary classifiers across the full set, even though it was poor. The resulting values for the classifiers are presented in table A.1.3 below. We choose to present the best, worst, mean and median values for the precision, recall and F-measure, to give an indication of the performance across the 10 random starts.

Bin. Class. (#**)	Pre. (M)	(B)	(W)	Rec. (M)	(B)	(W)	F (M)	(B)	(W)
$\in A$ vs. $\notin A$ **(20)**	0.5537	0.57789	0.5102	0.484	0.90909	0.32	0.50464	**0.65359**	0.4093
$\in B$ vs. $\notin B$ **(50)**	0.5502	0.58	0.51025	0.51709	0.90545	0.26182	0.51954	**0.65269**	0.35644
$\in C$ vs. $\notin C$ **(50)**	0.53706	0.57237	0.50091	0.59964	1	0.31636	0.54635	**0.66748**	0.40749
$\in D$ vs. $\notin D$ **(50)**	0.55132	0.58974	0.52941	0.51564	0.67273	0.36364	0.52695	**0.59581**	0.43956

Table 7.2.4: Primary results when analyzing prelabelled publications from set a)-c). Mean (M), best (B) and worst (W) performance (measured by precision, recall and F-measure) out of 10 random restarts for each classifier using 2500 iterations. **(Using 5 different number of neurons (# of neurons in parenthesis in the leftmost column)). For full results (for all classes) we refer the reader to appendix A

As can be seen in table A.1.3 we have a (in comparison with the "only SKF publications"-case in table 7.2.1) very poor performance across all binary classifiers. Note that the 10-class multiclass classifier performed even worse, whereby these results are omitted.

The results suggest that since the subset that we used from the start (the set of SKF publications a)) is contained within this dataset, the poor performance can be due to the inclusion of the other sets (b) and c) with corresponding results in table A.1.1 and A.1.2). This also suggests that to test this result we should investigate if the suggested framework performs well when analyzing the SKF subset within this complete dataset. This may seem very similar to the first experiment performed, but in this case we are applying the k-rank approximation to the full matrix of documents, i.e. sets a)-c) combined. This matrix and the k-rank approximation of it will differ from the matrix created when analyzing only SKFs publications, since the k-rank approximation is based on a different starting matrix in the different cases.

The reason this makes sense is that the SVD approximation of the three sets together is different from the SVD approximation of any of the three separately. This also means that when we have analyzed the three sets alone, the sets are more coherent "in-set" than "across-sets". The results indicate that the documents coming from the same company are more consistent in terms of their textual style and topical nature. This makes sense because the different companies, while operating in the same market and making similar products, do not focus on the same things when doing R&D. This is exemplified well in figure 7.1.2.

The major difference between the two datasets used is the differing amount of words used in the document descriptions. This corresponds to the rows of the TDM. The number of words analyzed in the SKF only case was 5966 terms, after dimensionality reduction. In the case analyzed here (when using stemming) the dictionary consisted of only 3929 terms. This means that the major difference between the two experiments, may be due to the amount of input variables to each neuron in the hidden layer. When the number of inputs goes from ≈ 6000 terms to ≈ 4000 terms, one could expect a worsening of performance since this less diverse description may lessen the performance of the suggested framework.

The major source of this limitation of words is due to stemming of words, which accounts for removing approximately 1400 words from the given set. Thus the question of if the stemming of words should be applied is raised. Theory ([CDM08]) suggests that stemming should not decresase the accuracy, since the words that are combined into single terms all are connected to the same topic. There are however other findings that suggest that stemming may cause a decrease in effectiveness, see for instance the results from Baker and McCallum [BM98].

Another aspect that was somewhat surprising is the limited improvement with an increase in the number of neurons used in the hidden layer. This may be consistent with problems due to overfitting discussed in section 4.4. This may also indicate that the suggested number of neurons in the hidden layer, as mentioned in section 7.2.1, may not be strongly relevant for improving classifier performance.

The general behaviour of the classifiers is that they tend towards classifying all documents as not belonging to any class (i.e. all are not in the class), and for some cases, all documents were in that class (for SI D). These results were not elaborated upon further, since the deviating nature and poor results did not contribute to the analysis.

We now turn to the three possible subsets within the superset considered above (with a), b) and c)), within the full dataset of publications (set a)-c)). These will all be analyzed based on the same SI as above, furthermore we will also investigate the case of using stemming, versus not using stemming.

Set a) within a)-c) with stemming

When studying the algorithm performance for set a) within the dataset consisting of the three sets of publications alone, performance increases. The performance is not comparable to the performance achieved when only creating the SVD approximation for set a) as presented in table 7.2.1, but it is sometimes better (for SI C and D) than the average performance for the set of all publications (a)-c)) seen in table A.1.3.

Bin. Class. (#**)	Pre. (M)	(B)	(W)	Rec. (M)	(B)	(W)	F (M)	(B)	(W)
$\in A$ vs. $\notin A$ **(2000)**	0.54675	0.6	0.50804	0.52215	1	0	0.5195	**0.67377**	0.24876
$\in B$ vs. $\notin B$ **(20)**	0.55877	0.59286	0.51282	0.48671	0.71519	0.20253	0.51127	**0.62088**	0.29224
$\in C$ vs. $\notin C$ **(400)**	0.52636	0.60294	0.49107	0.53671	1	0.11392	0.49099	**0.66667**	0.19048
$\in D$ vs. $\notin D$ **(100)**	0.60165	0.67568	0.54592	0.44304	0.67722	0.29747	0.49769	**0.60452**	0.4

Table 7.2.5: Primary results when analyzing only subset a) of prelabelled publications from set a)-c). Mean (M), best (B) and worst (W) performance (measured by precision, recall and F-measure) out of 10 random restarts for each classifier using 2500 iterations. **(Using 5 or 6 different number of neurons (# of neurons in parenthesis in the leftmost column)). For full results (for all classes) we refer the reader to appendix A

Since the results indicate that the improvement may be due to the stemming or simply random effects due to the initial allocation, any conclusions based on these results would only be spurious. One important observation however is that there still seems to be indications that a higher number of neurons in the hidden layer does not necessarily improve the classifier performance.

Set b) within a)-c) with stemming

When studying the algorithm performance for set b) within the dataset consisting of the three sets of publications alone, performance increases. The performance is considerably better compared to the performance achieved when creating the SVD approximation for only set b) as presented in table A.1.1. It is also significantly better than the average performance for the set of all publications (a)-c)) seen in table A.1.3.

Bin. Class. (#**)	Pre. (M)	(B)	(W)	Rec. (M)	(B)	(W)	F (M)	(B)	(W)
$\in A$ vs. $\notin A$ **(50)**	0.64711	0.81818	0.53659	0.73043	0.95652	0.3913	0.66967	**0.72727**	0.52941
$\in B$ vs. $\notin B$ **(20)**	0.65015	0.76471	0.6	0.68696	0.91304	0.47826	0.6566	**0.73684**	0.5641
$\in C$ vs. $\notin C$ **(100)**	0.73195	0.9	0.57692	0.54545	0.77273	0.31818	0.60481	**0.75556**	0.45161
$\in D$ vs. $\notin D$ **(50)**	0.72471	1	0.57895	0.58182	0.86364	0.18182	0.60952	**0.8**	0.30769

Table 7.2.6: Primary results when analyzing only subset b) of prelabelled publications from set a)-c). Mean (M), best (B) and worst (W) performance (measured by precision, recall and F-measure) out of 10 random restarts for each classifier using 2500 iterations. **(Using 5 or 6 different number of neurons (# of neurons in parenthesis in the leftmost column)). For full results (for all classes) we refer the reader to appendix A

These results indicate that there are company specific effects. These effects may give some indication that a smaller set (set b) consists of only 44 documents) set may have improved performance by being included a larger more complete sample, before creating the classifier.

The results are different based on if set b) is contained within a larger set when creating the k-rank approximation or not. An important distinction is that in the experiment using only competitor A's publications (set b)) we have a dictionary of only approx 1600 terms, as compared to a dictionary of approximately 4000 words in the experiment presented here. This indicates, as before, that an increase in the number of terms used to describe the set improves performance.

The significant difference in the results is not likely to be due to the small increase in the number of iterations, since the results in table A.1.4 do not show any effects similar to this.

Set c) within a)-c) with stemming

The same results as for set a) and b) are found when analyzing set c) within the dataset consisting of the three sets of publications alone, performance increases. The performance is better when compared to the performance achieved when only creating the SVD approximation for set c). It is also significantly better than the average performance for the set of all publications (a)-c)) seen in table A.1.3.

Bin. Class. (#**)	Pre. (M)	(B)	(W)	Rec. (M)	(B)	(W)	F (M)	(B)	(W)
$\in A$ vs. $\notin A$ **(400)**	0.59535	0.66667	0.5461	0.6383	0.85106	0.45745	0.60623	**0.67511**	0.53416
$\in B$ vs. $\notin B$ **(20)**	0.61717	0.66265	0.53896	0.59158	0.87368	0.41053	0.59478	**0.66667**	0.46988
$\in C$ vs. $\notin C$ **(20)**	0.59916	0.63158	0.56296	0.67474	0.8	0.56842	0.6323	**0.689**	0.58378
$\in D$ vs. $\notin D$ **(50)**	0.58529	0.62727	0.55319	0.67684	0.82105	0.55789	0.62486	**0.67568**	0.57297

Table 7.2.7: Primary results when analyzing only subset c) of prelabelled publications from set a)-c). Mean (M), best (B) and worst (W) performance (measured by precision, recall and F-measure) out of 10 random restarts for each classifier using 2500 iterations. **(Using 5 different number of neurons (# of neurons in parenthesis in the leftmost column)). For full results (for all classes) we refer the reader to appendix A

When analyzing the results in table A.1.6 we see that again as in the case of set b) within the a)-c) set, the set c) also has a better performance. Thus it seems that company specific effects are relevant when analyzing documents. This is clearly consistent with intuition that different companies with different areas of focus also publish documents with differing textual content. Furthermore the results once again show that the $70\% - 30\%$ split may be insufficient to create a high-performing classifier for a given dataset.

The results once again validate the results from before that the number of neurons seems to have a smaller effect on improving the classifier performance. The effect is most clear on the best F-measures for the different tests, which also is the most important measure of classifier performance.

7.2.5 Results for set a)-c) without stemming

If we consider the limited dictionary that describes the documents in the previous experiments, one explanation to the poorer performance may be a lack of terms. This "lack of terms" is consistent with our intuition regarding the problem - that a pattern-recognizing algorithm should perform better for a larger set of inputs. Our attention is therefore focused on what parts of the suggested framework that limit the dictionary size. The largest dictionary reducing factor in our analysis is the stemming of words, for more on stemming see appendix E.1. It has also been shown that stopword lists, see appendix E.3, when chosen correctly improve the dimensionality of the data as well as the performance of the classifier. Therefore we here focus on repeating the four experiment we have just described across the sets of publications (a), b), c) and a)-c)).

The comparative analysis between the results due to differing SVD factorizations was considered by considering the publications (sets a), b) and c) alone), and then considering the same set as a subset within a larger set of publications (a)-c) together). In other words: when we have a TDM, created from the full set of 551 documents over three different companies and only use the columns (documents) from set a), b) or c) and compare to the previous results, we should see the SVD-dependent change.

Besides this the results below are from when not applying stemming, to account for the effects of a more limited dictionary. These effects are important to evaluate if performance changes due to the inclusion of stemming or not in the preprocessing of data.

Bin. Class. (#**)	Pre. (M)	(B)	(W)	Rec. (M)	(B)	(W)	F (M)	(B)	(W)
$\in A$ vs. $\notin A$ **(200)**	0.5453	0.5988	0.5	0.61055	1	0.36364	0.56382	**0.66667**	0.45249
$\in B$ vs. $\notin B$ **(150)**	0.53099	0.58898	0.5	0.60545	1	0.40727	0.55434	**0.66667**	0.4489
$\in B$ vs. $\notin B$ **(200)**	0.55428	0.58261	0.5	0.544	1	0.24364	0.53144	**0.66667**	0.34359
$\in C$ vs. $\notin C$ **(100)**	0.52667	0.575	0.33333	0.592	0.85455	0.0072727	0.53512	**0.64384**	0.014235
$\in D$ vs. $\notin D$ **(150)**	0.52822	0.55372	0.51071	0.53455	0.76364	0.35636	0.52562	**0.62222**	0.43267

Table 7.2.8: Primary results when analyzing prelabelled publications from set a)-c) without applying stemming. Mean (M), best (B) and worst (W) performance (measured by precision, recall and F-measure) out of 10 random restarts for each classifier using 2500 iterations. **(Using 5 or 6 different number of neurons (# of neurons in parenthesis in the leftmost column)). For full results (for all classes) we refer the reader to appendix A

The resulting values presented in table A.1.7 indicate that there are some limitations when using stemming. If we study the results from the first experiment for the full set, presented in table A.1.3, we see that the best F-measures are achieved in the absence of stemming. The best F-measure values, given a fixed number of neurons in the hidden layer, are also better overall when not using stemming.

In general the results are still much poorer than the results from for instance only set a) table 7.2.1. This still indicates that the $70\% - 30\%$ split has a very strong influence on the results, as is expected. This is due to that if the test set only represents 5% of the full set (as in the 95%-5% split used here), there is a very high chance that the few documents on which the algorithm is tested will be easy to classify. This is especially true when evaluating a strongly coherent set of data, that each come from the same source (company).

Moving on we present the results when not applying the stemming and then evaluate the performance of each subset as a part within the larger set.

Set a) within a)-c) without stemming

To continue the analysis of the influence of stemming we evaluate the different subsets a), b) and c) within the superset of a)-c) considered as a whole. When we do not apply the stemming the results in a larger context hint that stemming lessens the best possible performance of the algorithm, within the experiments we have considered. In table A.1.8 we present the results for subset a) within the superset.

Bin. Class. (#**)	Pre. (M)	(B)	(W)	Rec. (M)	(B)	(W)	F (M)	(B)	(W)
$\in A$ vs. $\notin A$ **(50)**	0.56585	0.59804	0.5445	0.5962	0.77848	0.38608	0.57332	**0.65252**	0.46923
$\in B$ vs. $\notin B$ **(100)**	0.53223	0.56863	0.50185	0.69623	0.98113	0.53459	0.59591	**0.66953**	0.53583
$\in C$ vs. $\notin C$ **(200)**	0.548	0.57297	0.51685	0.67089	0.87975	0.41139	0.59703	**0.65566**	0.47445
$\in D$ vs. $\notin D$ **(100)**	0.538	0.57407	0.50211	0.50696	0.90506	0.29114	0.50189	**0.65297**	0.38017

Table 7.2.9: Primary results when analyzing only subset a) of prelabelled publications from set a)-c) without stemming. Mean (M), best (B) and worst (W) performance (measured by precision, recall and F-measure) out of 10 random restarts for each classifier using 2500 iterations. **(Using 5 different number of neurons (# of neurons in parenthesis in the leftmost column)). For full results (for all classes) we refer the reader to appendix A

As can be seen when comparing these results (table A.1.8) with the results when applying stemming (table A.1.4) we observe that there is a consistently better result when evaluating the F-measure, except for the case of SI A. This may be explained by many different things, but mainly it points to the necessity to check the results when using stemming. For all other cases and classes the results are better, with higher average F-measures for all classes. The method is thus more consistent for this dataset when we exclude stemming from the preprocessing of data.

Set b) within a)-c) without stemming

Analyzing of the influence of stemming we evaluate the different subsets a), b) and c) within the superset of a)-c) considered as a whole. When we do not apply the stemming the results in a larger context hint that stemming lessens the best possible performance of the algorithm, within the experiments we have considered. In table A.1.8 we present the results for subset b) within the superset.

Bin. Class. (#**)	Pre. (M)	(B)	(W)	Rec. (M)	(B)	(W)	F (M)	(B)	(W)
$\in A$ vs. $\notin A$ **(200)**	0.66158	0.8	0.57895	0.64783	0.95652	0.43478	0.63939	**0.72131**	0.51282
$\in B$ vs. $\notin B$ **(150)**	0.53829	0.61111	0.47368	0.55238	0.80952	0.33333	0.54	**0.66667**	0.4
$\in C$ vs. $\notin C$ **(20)**	0.6296	0.70833	0.57143	0.59091	0.77273	0.18182	0.59495	**0.73913**	0.27586
$\in D$ vs. $\notin D$ **(150)**	0.61811	0.72727	0.56522	0.55909	0.81818	0.36364	0.57545	**0.7234**	0.45714

Table 7.2.10: Primary results when analyzing only subset b) of prelabelled publications from set a)-c) without stemming. Mean (M), best (B) and worst (W) performance (measured by precision, recall and F-measure) out of 10 random restarts for each classifier using 2500 iterations. **(Using 5 different number of neurons (# of neurons in parenthesis in the leftmost column)). For full results (for all classes) we refer the reader to appendix A

In contrast with the results in table A.1.8 the results presented here indicate that stemming does improve performance. When comparing these results (table A.1.9) with the results when applying stemming (table A.1.5) the improvement observed previously is not noticeable. These results indicate that the stemming is good for the performance of the classifier in the case of set b) within the whole set (a)-c)).

Set c) within a)-c) without stemming

To evaluate the influence of stemming we analyze the case where we do not apply the stemming to the data. The results in a larger context (table A.1.7) hint that stemming lessens the best possible performance of the algorithm, within the experiments we have considered. In table A.1.8 we present the results for subset b) within the superset.

Bin. Class. (#**)	Pre. (M)	(B)	(W)	Rec. (M)	(B)	(W)	F (M)	(B)	(W)
$\in A$ vs. $\notin A$ **(200)**	0.57334	0.58696	0.55357	0.6117	0.7766	0.41489	0.58798	**0.65766**	0.48447
$\in B$ vs. $\notin B$ **(100)**	0.55566	0.60606	0.49091	0.63158	0.93684	0.21053	0.56828	**0.66667**	0.3125
$\in C$ vs. $\notin C$ **(20)**	0.56664	0.59756	0.52727	0.66105	0.91579	0.50526	0.60185	**0.66923**	0.52747
$\in D$ vs. $\notin D$ **(20)**	0.60294	0.66667	0.55372	0.64737	0.74737	0.48421	0.62043	**0.66667**	0.56098
$\in D$ vs. $\notin D$ **(200)**	0.61036	0.65714	0.5679	0.59368	0.74737	0.46316	0.59634	**0.66667**	0.52273

Table 7.2.11: Primary results when analyzing only subset c) of prelabelled publications from set a)-c) without stemming. Mean (M), best (B) and worst (W) performance (measured by precision, recall and F-measure) out of 10 random restarts for each classifier. **(Using 5 different number of neurons (# of neurons in parenthesis in the leftmost column)). For full results (for all classes) we refer the reader to appendix A

In this case the results in table A.1.10 when compared with table A.1.6 show that for set c) applying stemming improves results. This once again emphasizes the case-to-case difference when applying stemming and that it can both increase and decrease performance.

Out of the total set of a)-c) (516 documents) the different sets differ in size, set a) is 316 documents, set b) is 44 documents and set c) is 156 documents. This may also have an influence in the improvement due to stemming. Since for the two smaller subsamples it improves and for the larger subset it does not improve. The thesis will not elaborate further on this but suggests that future work is still needed to evaluate the use of stemming.

7.2.6 Results for set a), b) and c) without stemming (large number of iterations)

In the next experiment we apply the suggested framework to the three mentioned subsets without applying stemming, but running the network training for a large number of iterations. Since the number of neurons has been found to not be a strongly significant variable, the number of neurons was chosen to 20 due to computational speed. Also the choice was to emphasize the results that can be achieved given that one uses a strongly limited number of neurons in the ANN. This experiment was also run with a $95\% - 5\%$ split of the data into training and test set, to emphasize the difference, if any, that is due to increasing the number of training samples.

Furthermore these experiments serve to emphasize the quality of splitting the data into coherent subsets. Whereas the previous experiments have started from the perspective of a full superset within which the subsets are extracted. This experiment instead focuses on each of the sets in their own right. This should show the best possible results by means of an increase in the number of iterations in the BPNN algorithm.

We present the results in order, starting from set a) and then moving on to set b) and lastly set c).

Bin. Class. (#**)	Pre. (M)	(B)	(W)	Rec. (M)	(B)	(W)	F (M)	(B)	(W)
$\in A$ vs. $\notin A$ (20)	0.97272	0.9871	0.95625	0.95411	0.96835	0.91772	0.96317	**0.97764**	0.94463
$\in B$ vs. $\notin B$ (20)	0.95908	0.98065	0.93789	0.9557	0.96203	0.94937	0.95729	**0.97125**	0.94671
$\in C$ vs. $\notin C$ (20)	0.96406	0.98667	0.94479	0.96361	0.97468	0.93671	0.9636	**0.97161**	0.9595
$\in D$ vs. $\notin D$ (20)	0.9701	0.97436	0.96835	0.97468	0.98734	0.96203	0.97235	**0.97806**	0.96815

Table 7.2.12: Results when analyzing only subset a) of prelabelled publications from set a)-c) without stemming. Mean (M), best (B) and worst (W) performance (measured by precision, recall and F-measure) out of 4 random restarts for each classifier. **(Using only 20 neurons and 100000 feedback iterations).

The results when using a large number of iterations are very convincing when looking at set a). The results indicate that in the case of set a) the results presented in table 7.2.1 were not coincidental, but that it is possible to improve upon those results by means of using a larger number of iterations (as seen here in table 7.2.12). These results can be compared with the F-measures in table 7.2.1, A.1.4 and A.1.8. Moving on we present the results for set b).

Bin. Class. (#**)	Pre. (M)	(B)	(W)	Rec. (M)	(B)	(W)	F (M)	(B)	(W)
$\in A$ vs. $\notin A$ (20)	0.8503	0.9048	0.7692	0.88	0.9091	0.8636	0.86105	**0.8837**	0.8333
$\in B$ vs. $\notin B$ (20)	0.71571	0.7619	0.68	0.73864	0.77273	0.68182	0.72624	**0.75556**	0.68182
$\in C$ vs. $\notin C$ (20)	0.81745	0.90909	0.7619	0.70455	0.86364	0.45455	0.74185	**0.84444**	0.60606
$\in D$ vs. $\notin D$ (20)	0.97222	1	0.88889	0.67045	0.72727	0.59091	0.79069	**0.84211**	0.74286

Table 7.2.13: Results when analyzing only subset b) of prelabelled publications from set a)-c) without stemming. Mean (M), best (B) and worst (W) performance (measured by precision, recall and F-measure) out of 4 random restarts for each classifier. **(Using only 20 neurons and 100000 feedback iterations).

When studying the results in table 7.2.13 there is a clear improvement, as compared to previous results (table A.1.1, A.1.5 and A.1.9). This once again indicates that the strongest method for improving the results is by increasing the number of iterations or by using a bigger training set. One of the significant cons with the ANN approach, as described in section 4.5, was that the ANN needs a sufficient number of training data. The results from the experiments are clearly consistent with the theory.

Bin. Class. (#**)	Pre. (M)	(B)	(W)	Rec. (M)	(B)	(W)	F (M)	(B)	(W)
$\in A$ vs. $\notin A$ (20)	0.77081	0.82692	0.71579	0.59211	0.71579	0.45263	0.66248	**0.71579**	0.58503
$\in B$ vs. $\notin B$ (20)	0.72075	0.76271	0.6962	0.56842	0.66316	0.47368	0.63215	**0.68478**	0.58442
$\in C$ vs. $\notin C$ (20)	0.67526	0.73134	0.63636	0.58947	0.66316	0.51579	0.62645	**0.64948**	0.60494
$\in D$ vs. $\notin D$ (20)	0.67889	0.74419	0.60959	0.78158	0.93684	0.67368	0.71982	**0.73859**	0.70531

Table 7.2.14: Results when analyzing only subset c) of prelabelled publications from set a)-c) without stemming. Mean (M), best (B) and worst (W) performance (measured by precision, recall and F-measure) out of 4 random restarts for each classifier. **(Using only 20 neurons and 100000 feedback iterations).

The final results from this set of experiments for set c) are no surprise. However the results indicate (when compared to table A.1.2, A.1.6 and A.1.10) that the significant improvement shown in the case of set a) and b) does not show itself as much in the case of set c). This again strongly indicates that the set c) is simply harder to classify into the SI paradigm based on the suggested framework.

Finally it must be stated that the improvement in F-measure is usually due to the improvement in recognizing the pattern in the training set. This means that the mean square error will consistently decrease until it approaches convergence. Using the large number of iterations (100000 iterations) does not ascertain that we will have convergence, but in the cases examined this is clearly sufficient for no further improvements to be observed. This should be contrasted with the resulting convergence if we use a limited number of iterations.

7.2.7 Using a large number of iterations

When analyzing this case figure 7.2.1 gives some indications of the possible caveats when limiting the number of iterations.

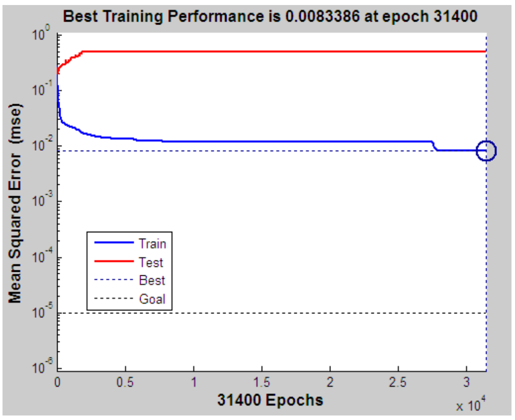


Figure 7.2.1: *Example of late convergence for the BPNN algorithm around iteration 26500 (Epochs ≡ Iterations)*

As can be seen an early stopping condition based on number of iterations would limit the improvement in the classifier. Note the log-scale on the y-axis in figure 7.2.1, which means that the late improvement takes the MSE from $MSE_{26000} \approx 0.015$ down to $MSE_{28000} \approx 0.0083$ which in this context (as in any pattern recognition task) is a major improvement (a factor of 2). Thus we also note the importance of letting the algorithm either converge to a preferred level of the MSE or some other measure of quality, rather than limiting the number of iterations.

The other graph (the resulting MSE for the test part of the data) also warrants a comment. In general the framework and the BPNN will train the algorithm to recognize the patterns it can find in the training data, since this is the only data that is available to learn the algorithm. This means that the pattern recognizer will be improving its classification, but only for classifying the documents in the training set. This means that for the test set, as the case is in figure 7.2.1, there may be a lessening of performance for the test set, since this is not exactly equal to the training set. This is the major limitation of pattern recognition, but as is clear it is impossible to improve the framework since the classes within the test set by definition are not known.

7.3 Summary of results

There seems to be a consistent improvement to be achieved by running the BPNN algorithm with a larger number of iterations. This is clear from the theory presented in the chapter 5 where we presented the framework. The number of corrections in the network is more relevant than the number of neurons in the hidden layer for improving the performance of the network. This is consistent with our intuition as well, since the number of correction feedback loops are more important. In other words we would rather have fewer neurons in the hidden layer that all have learned to be better at recognizing the pattern in the TDV, than to have a large number of neurons that each has learned less well.

This is clearly consistent with the previous experiments as well, and since there are no significant improvements due to the number of neurons we suggest that a smaller number of neurons be used to increase the speed. Furthermore following the principle of *Occam's razor* we use fewer neurons to decrease the complexity of the model and suggest that the number of iterations to learn the algorithm is strongly increased instead to improve performance.

Based on the above presented results the inclusion of stemming or not may or may not improve performance. Including smaller sets in a larger superset, may improve the resulting classification scheme, most probably due to the larger dictionary used to describe the document collection. With regards to the terms describing the documents, the experiments seem to indicate that a larger set of terms improves the performance of the classifier, furthermore the usage of stopwords improves upon the results.

When evaluating the k-rank approximation it is always good to make use of this approximation to the original TDM. Experiments using the suggested framework on the raw TDM gave very poor results in the classification task. On the topic of classification a binary classification scheme improves the results when compared to a multiclass classifier. With regards to the number of iterations a larger number is more appropriate and the number of neurons in the hidden layer is less so. This does not include a very low number, but for the classification task considered the last case using 20 neurons was sufficient to yield the highest F-measures for set a) as well as the highest F-measure for the most classes studied. Thus the recommendation for a given problem is to preferably let the number of iterations have preference over the number of neurons in the hidden layer. Also the number of inputs give better results with a higher number of inputs.

This being said, each classification task should be evaluated on a case by case basis. There is no single framework that can encompass the full extent of complexity that a text categorization task represents. As has been presented many times in the results discussed above, there are some parts of the framework that can be given recommendations, whilst the remaining parts should be evaluated based on the given case.

7.4 Temporal distributions of classes

In this section we will outline the resulting distribution based on the classifiers resulting categorization. When a document is binary classified into multiple classes it is allocated into the classes to which it has the least error. In the case of a multiple allocation it is evenly distributed across the given sets, i.e. if two binary classifiers classifies the document into their respective class we weight the resulting document allocation. In that specific case it is weighted as 0.5 in one class and 0.5 in the other. This is done to ensure that the normalization factor is correct across the sum of all documents. In other words the 100% in the temporal distributions should reflect the sum of all documents published that specific year. Thus we truly get a fair distribution of the topical nature of the documents over time which enables a trend analysis of the topics of publications and patents from the given companies.

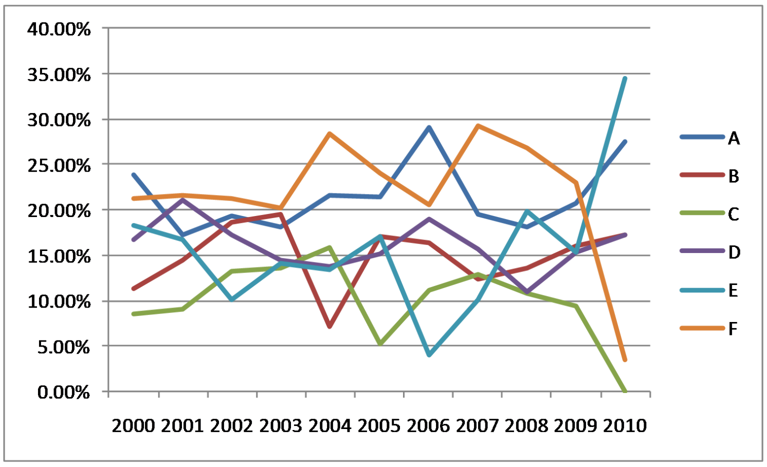


Figure 7.4.1: *The relative distribution (per year) of SKFs patents and publications taken together across SI over the studied period (2000-2010)*

For the case of SKFs publishing behaviour we can use the resulting classification into strategic intents to study the temporal changes in focus over time. This result shows that SKF has focused more toward SI E and less toward SI F during 2009-2010. Furthermore SI C seems also to have achieved less focus whilst SI B and D have gotten a stronger focus over the same period.

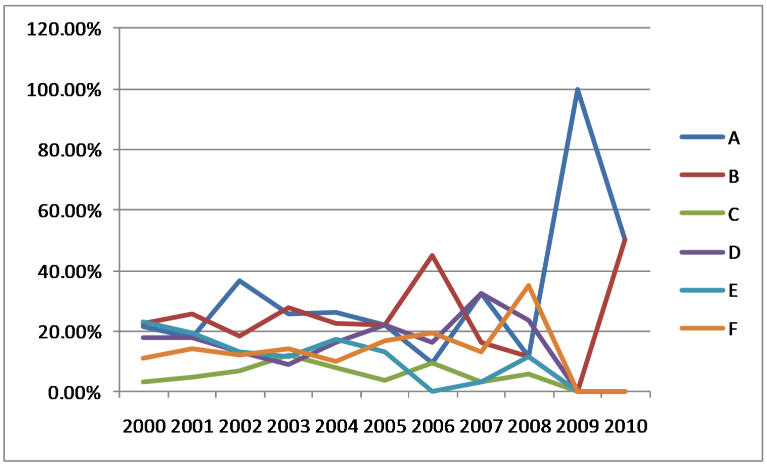


Figure 7.4.2: *The relative distribution (per year) of competitor As patents and publications taken together patents across SI over the studied period (2000-2010)*

For competitor As publications and patents the resulting classification also displays relevant trends. For competitor A the focus in SI A grew to 100% in 2009, but has declined after and is together with SI B the two single areas of focus currently. For the rest of the SI the focus seems to have decreased over time, ending at 0 publications or patents in the corresponding areas in 2010.

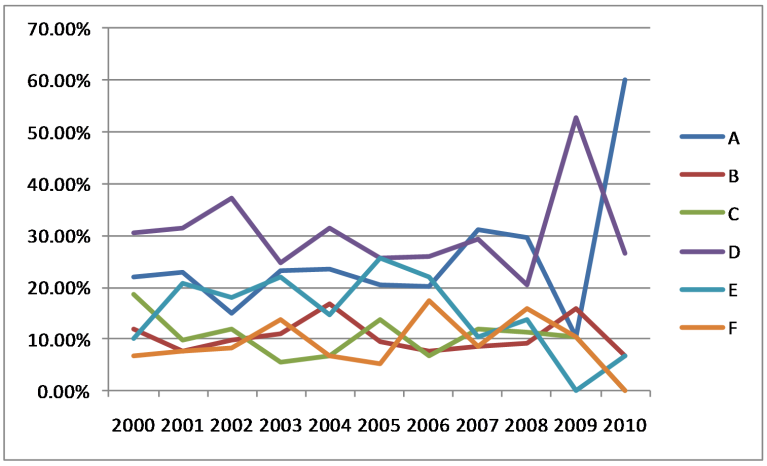


Figure 7.4.3: *The relative distribution (per year) of competitor Bs patents and publications taken together across SI over the studied period (2000-2010)*

For competitor Bs publishing trends we see strong displays of focus over time as well. The focus on SI A and D are strong, while SI B, E and F have less focus toward the end of the period. What is also clear from figure 7.4.3 is the strong profiling (throughout the period studied) on SI A and D.

Finally it can be stated that for all the cases the focus over time has shifted. There seems to be a stronger focus on core over time, specifically during the last years, after the financial crisis. Furthermore the trend analyses generate non trivial bases for comparisons across companies and also within companies across SI.

8 Discussion & conclusions

We will finally turn to the conclusions that can be drawn from the thesis work. The thesis set out to assess whether documents can be grouped together according to a company internal classification scheme, in this case strategic intents at SKF. It investigated what sources can be used, how to model them, the methods to allocate them, and the evaluation of how well the allocation worked. We analyzed patents and publications and described them using the BOW model (see section 4.2.8) together with LSI (see section 4.3.6). Furthermore we used an ANN (see section 4.3.4) and the BPNN algorithm (see section 5.3.1) to train the pattern recognition framework for generating the results according to a binary classification scheme.
The suggested framework can be used to classify documents according to strategic intents. It was also found that the results can be used to evaluate trends in strategic intents on a company level. The resulting classifications are within an acceptable level of accuracy and thus also validates the suggested approach.

When analyzing the results we found many interesting aspects. One aspect that was analyzed was the amount of prelabelled documents necessary for generating a high quality classifier. It was shown that a classifier performs better when increasing the amount of prelabelled documents.
The results also showed that for smaller datasets it may help to include these as part of a larger set of prelabelled documents. This means that if one has many prelabelled documents, it can be useful to add them to the training of the classifier, even if it will be applied to documents from a different class.

With regards to classifier performance, it was tested using different basic assumptions on the training/test split of the data, numbers of neurons in the hidden layer and number of iterations.
For different numbers of neurons in the hidden layer the results showed that a higher number of neurons does not add quality to the classifier. There is of course a lower limit to the number of neurons needed to evaluate the patterns in the documents, but this was not investigated. More specifically it seems that between 20-50 neurons are sufficient for modeling a collection with approximately 5000-8000 features (inputs).
When discussing the number of iterations the principle result is that the higher the number the better. Each iteration represents a correction step which is fed back through the full network, thus for each iteration there is an improvement. This is consistent with intuition and previous results, and the conclusion is to either set a specific number of iterations, or use a convergence criteria based on the MSE.
We also investigated the use of stemming and its effect on classifier performance. The results show that for some datasets it may have a positive effect while in other cases stemming decreases classifier performance. Thus the results are consistent with theory and previous investigations within this topic.

With regards to the size of the specific strategic intent within the prelabelled dataset, and its percentage size we investigated if this may affect classifier performance. The first experiments indicated some effect, but over the whole set of experiments there was no consistently lower "best" performance if the specific strategic intent is underrespresented in the prelabelled data. This is mainly due to the random restarts of the classifier training. This will generally result in that there is a higher probability that there will be one "good" split of the data.

Based on the results we can give some preliminary recommendations when using the suggested framework. It seems that stemming does not add especially to classifier accuracy. Increasing the number of iterations improves the performance, but also increases the risk of overfitting. 20-50 neurons in the hidden layer are a sufficient number and increasing the number is not significant in improving the accuracy of the classifier. The classifier works better when appliead to a binary classification as opposed to a multiclass classification.

Using the strategic intents and the suggested framework for automatically classifying documents gives a methodology for making trend analyses of company behaviour. Based on the concept of the internal classification scheme this approach could be extended and applied to any classification and any text based documents. It could also be extended to evaluate other types of features and other properties of the data. The framework is modular and supports exchangeable sources and classification algorithms.
The framework functions better for a binary classfication than a multiclass classification, and this is an important aspect of the classification problem. Using a too small set of prelabelled documents reduces the performance of the classifier and increasing the ratio of prelabelled to unlabelled documents increases classifier performance. The number of iterations used in the algorithm has a strong influence on the resulting performance, and the number of neurons less so. The results also show that using these automated allocation methods we can assess companies, generating time analyses of their focus areas over time.

The method is by no means perfect and there is much to improve upon. First a more structured set of features, perhaps a set dictionary, would improve the creation and consistency of the classifiers. The method in itself is also very simple, it does not, for instance cover what types of words are being analyzed, such as nouns, verbs, etc. These have been shown to add some information in the modeling of documents. Furthermore the ANN is subject to much criticism, based on its black-box characteristic. This critique can perhaps be addressed by focusing a strong effort on finding the optimal settings of ANNs form specific pattern recognition tasks.

The topic of automaticallly allocating documents using automated algorithms has grown in importance over the last decades and continues to do so [Seb02]. The thesis has shown that automatically allocating documents using ANN and LSI is a viable approach. To improve upon this research there are many areas that could and should be addressed. The first area is the area of modeling a textual data source. How to model documents in a more advanced way than simply using the BOW model is an active area of research. This area should be pursued further to achieve a more accurate description of documents and their content. The second area, which is intimately connected to the first, is the area of what types of features to use when analyzing documents. If there are a multitude of methods to create and analyze the different features of a document, then these could also be included in the modeling of the same. A third area that could be of interest would be to do more qualitative tests on possible performance improvements to the ANN model for pattern recognition. Guidelines for the number of neurons, complexity of the model, number of needed features to analyze and similar aspects that go much further than these first recommendations would be of great value to companies and organizations worldwide.

Finally it can stated that the usage of strategic intents is a powerful tool which enables essential trend analyses. The suggested framework has been shown to have great potential and is generalizable to many types of documents and areas of analysis. More importantly these analyses allow for strategic evaluations of companies both internally and for benchmarking against competitors. The future is clouded in the possibilities it may or may not hold, thus we must do what we can in the present. The suggested framework provides us with one piece of this complex puzzle.

APPENDIX

A Full results

A.1 Full Results using suggested framework

Bin. Class. (#**)	Pre. (M)	(B)	(W)	Rec. (M)	(B)	(W)	F (M)	(B)	(W)
$\in A$ vs. $\notin A$ (20)	0.5842	0.65	0.5	0.60435	0.95652	0.21739	0.5641	**0.6875**	0.32258
$\in A$ vs. $\notin A$ (50)	0.59913	0.66667	0.5625	0.57826	0.78261	0.26087	0.57188	**0.65455**	0.375
$\in A$ vs. $\notin A$ (100)	0.67962	1	0.54054	0.53913	0.86957	0.13043	0.54843	**0.69388**	0.23077
$\in A$ vs. $\notin A$ (150)	0.63062	0.7	0.53333	0.48261	0.65217	0.17391	0.53062	**0.65217**	0.26667
$\in A$ vs. $\notin A$ **(200)**	0.62979	1	0.51351	0.61739	0.91304	0.13043	0.57196	**0.70175**	0.23077
$\in B$ vs. $\notin B$ (20)	0.59032	1	0.5	0.56667	0.90476	0.047619	0.50962	**0.67857**	0.090909
$\in B$ vs. $\notin B$ (50)	0.55038	0.6	0.51724	0.55714	0.90476	0	0.56911	**0.66667**	0.41176
$\in B$ vs. $\notin B$ (100)	0.55732	0.66667	0.375	0.58571	0.80952	0.14286	0.55448	**0.65306**	0.2069
$\in B$ vs. $\notin B$ (150)	0.5458	0.63636	0.33333	0.5619	0.90476	0.047619	0.52622	**0.67925**	0.083333
$\in B$ vs. $\notin B$ **(200)**	0.55287	0.625	0.48387	0.66667	0.95238	0.2381	0.57866	**0.69091**	0.33333
$\in C$ vs. $\notin C$ **(20)**	0.57699	0.64706	0.54839	0.69545	0.90909	0.5	0.62303	**0.70175**	0.52381
$\in C$ vs. $\notin C$ (50)	0.56824	0.63158	0.51429	0.71364	0.95455	0.54545	0.6268	**0.67742**	0.57778
$\in C$ vs. $\notin C$ (100)	0.54008	0.59091	0.42857	0.71364	0.90909	0.13636	0.59663	**0.68966**	0.2069
$\in C$ vs. $\notin C$ (150)	0.59006	0.7	0.54286	0.66818	0.86364	0.31818	0.60767	**0.67925**	0.4375
$\in C$ vs. $\notin C$ (200)	0.58095	0.625	0.52381	0.71364	1	0.45455	0.63067	**0.6875**	0.52632
$\in D$ vs. $\notin D$ (20)	0.60799	0.8	0.54839	0.69545	0.90909	0.36364	0.63109	**0.68966**	0.5
$\in D$ vs. $\notin D$ (50)	0.55838	0.6087	0.46154	0.68182	0.90909	0.045455	0.587	**0.68**	0.083333
$\in D$ vs. $\notin D$ (100)	0.59251	0.64286	0.55556	0.60909	0.81818	0.40909	0.5912	**0.69231**	0.48649
$\in D$ vs. $\notin D$ **(150)**	0.49358	0.6	0	0.60909	0.90909	0	0.58481	**0.7037**	0.2069
$\in D$ vs. $\notin D$ (200)	0.57028	0.64706	0.5	0.69091	0.90909	0.45455	0.61259	**0.68966**	0.47619

Table A.1.1: Results when analyzing prelabelled publications from only competitor A (set b)). Best performance (measured by precision, recall and F-measure) out of 10 random restarts for each classifier using 500 iterations.

Bin. Class. (#**)	Pre. (M)	(B)	(W)	Rec. (M)	(B)	(W)	F (M)	(B)	(W)
$\in A$ vs. $\notin A$ (20)	0.60139	0.67606	0.53846	0.52105	0.58947	0.32632	0.55279	**0.59574**	0.43357
$\in A$ vs. $\notin A$ (50)	0.60453	0.7	0.55405	0.47895	0.61053	0.25263	0.51895	**0.59794**	0.36923
$\in A$ vs. $\notin A$ (100)	0.63297	0.70833	0.53061	0.46526	0.65263	0.2	0.51929	**0.6**	0.31148
$\in A$ vs. $\notin A$ (150)	0.62307	0.79167	0.52577	0.41684	0.62105	0.2	0.48463	**0.60513**	0.31933
$\in A$ vs. $\notin A$ **(200)**	0.61771	0.69388	0.5463	0.46947	0.65263	0.25263	0.52119	**0.61084**	0.36923
$\in B$ vs. $\notin B$ **(20)**	0.58268	0.66667	0.53488	0.47684	0.77895	0.33684	0.51451	**0.64069**	0.44755
$\in B$ vs. $\notin B$ (50)	0.57694	0.625	0.54651	0.49474	0.6	0.37895	0.52899	**0.57868**	0.46452
$\in B$ vs. $\notin B$ (100)	0.57751	0.60938	0.52778	0.51895	0.75789	0.36842	0.53872	**0.63717**	0.45455
$\in B$ vs. $\notin B$ (150)	0.58005	0.625	0.54717	0.49053	0.65263	0.26316	0.52	**0.59903**	0.37037
$\in B$ vs. $\notin B$ (200)	0.58963	0.65385	0.54545	0.49579	0.71579	0.31579	0.52572	**0.62385**	0.41096
$\in C$ vs. $\notin C$ (20)	0.54076	0.5618	0.51685	0.52421	0.70526	0.2	0.5187	**0.6036**	0.29457
$\in C$ vs. $\notin C$ (50)	0.54731	0.59677	0.50993	0.51579	0.81053	0.23158	0.51353	**0.62602**	0.32836
$\in C$ vs. $\notin C$ (100)	0.53923	0.57143	0.51908	0.55263	0.75789	0.27368	0.53338	**0.621**	0.35862
$\in C$ vs. $\notin C$ **(150)**	0.5412	0.58333	0.49432	0.58947	0.91579	0.36842	0.54658	**0.64207**	0.45161
$\in C$ vs. $\notin C$ (200)	0.54303	0.58065	0.52174	0.55368	0.75789	0.36842	0.54033	**0.61803**	0.44586
$\in D$ vs. $\notin D$ **(20)**	0.54368	0.58824	0.51351	0.57579	0.89474	0.14737	0.53614	**0.66667**	0.22951
$\in D$ vs. $\notin D$ (50)	0.55403	0.61039	0.51456	0.59684	0.86316	0.4	0.56605	**0.66129**	0.45238
$\in D$ vs. $\notin D$ (100)	0.55244	0.58654	0.52564	0.59895	0.78947	0.31579	0.56358	**0.64455**	0.39735
$\in D$ vs. $\notin D$ (150)	0.55522	0.57759	0.52041	0.65053	0.8	0.48421	0.5952	**0.65236**	0.52571
$\in D$ vs. $\notin D$ (200)	0.55589	0.58824	0.53077	0.53895	0.72632	0.25263	0.53776	**0.61333**	0.34783

Table A.1.2: Results when analyzing prelabelled publications from only competitor B (set c)). Best performance (measured by precision, recall and F-measure) out of 10 random restarts for each classifier using 500 iterations.

Bin. Class. (#**)	Pre. (M)	(B)	(W)	Rec. (M)	(B)	(W)	F (M)	(B)	(W)
$\in A$ vs. $\notin A$ **(20)**	0.5537	0.57789	0.5102	0.484	0.90909	0.32	0.50464	**0.65359**	0.4093
$\in A$ vs. $\notin A$ (50)	0.54315	0.59	0.52365	0.48873	0.66909	0.21455	0.50459	**0.59547**	0.31467
$\in A$ vs. $\notin A$ (100)	0.54151	0.58382	0.51746	0.51309	0.60727	0.26182	0.51872	**0.56552**	0.35468
$\in A$ vs. $\notin A$ (150)	0.54371	0.57	0.50512	0.50945	0.66545	0.41455	0.52205	**0.59032**	0.47773
$\in A$ vs. $\notin A$ (200)	0.55926	0.58389	0.51759	0.45018	0.74909	0.31636	0.48761	**0.61218**	0.41038
$\in B$ vs. $\notin B$ (20)	0.54818	0.58065	0.49441	0.55927	0.64364	0.45818	0.55154	**0.58904**	0.49802
$\in B$ vs. $\notin B$ **(50)**	0.5502	0.58	0.51025	0.51709	0.90545	0.26182	0.51954	**0.65269**	0.35644
$\in B$ vs. $\notin B$ (100)	0.56853	0.63095	0.54264	0.48145	0.64364	0.19273	0.50764	**0.59197**	0.29526
$\in B$ vs. $\notin B$ (150)	0.55652	0.5885	0.52966	0.50109	0.60364	0.40364	0.52526	**0.56655**	0.47639
$\in B$ vs. $\notin B$ (200)	0.56667	0.59426	0.52895	0.51091	0.73091	0.34545	0.5318	**0.61374**	0.43678
$\in C$ vs. $\notin C$ (20)	0.55535	0.58	0.52888	0.51273	0.63273	0.40364	0.5299	**0.58304**	0.46934
$\in C$ vs. $\notin C$ **(50)**	0.53706	0.57237	0.50091	0.59964	1	0.31636	0.54635	**0.66748**	0.40749
$\in C$ vs. $\notin C$ (100)	0.54474	0.56129	0.53086	0.49745	0.62182	0.31636	0.51396	**0.57868**	0.40465
$\in C$ vs. $\notin C$ (150)	0.54287	0.56654	0.52232	0.47782	0.56727	0.34909	0.50587	**0.5539**	0.42478
$\in C$ vs. $\notin C$ (200)	0.5545	0.57477	0.53465	0.48	0.56364	0.39273	0.51253	**0.54965**	0.45283
$\in D$ vs. $\notin D$ (20)	0.55611	0.58857	0.54167	0.51236	0.61455	0.37455	0.52964	**0.57581**	0.45778
$\in D$ vs. $\notin D$ **(50)**	0.55132	0.58974	0.52941	0.51564	0.67273	0.36364	0.52695	**0.59581**	0.43956
$\in D$ vs. $\notin D$ (100)	0.5604	0.57931	0.53374	0.48109	0.63273	0.30545	0.51075	**0.57903**	0.4
$\in D$ vs. $\notin D$ (150)	0.5501	0.57062	0.52843	0.49891	0.62909	0.36727	0.51862	**0.58151**	0.4469
$\in D$ vs. $\notin D$ (200)	0.55252	0.56388	0.53401	0.49673	0.57091	0.41818	0.52142	**0.55185**	0.47917

Table A.1.3: Results when analyzing prelabelled publications from set a)-c). Mean (M), best (B) and worst (W) performance (measured by precision, recall and F-measure) out of 10 random restarts for each classifier using 2500 iterations. **(Using 5 different number of neurons (# of neurons in parenthesis in the leftmost column)). For full results (for all classes) we refer the reader to appendix A

Bin. Class. (#**)	Pre. (M)	(B)	(W)	Rec. (M)	(B)	(W)	F (M)	(B)	(W)
$\in A$ vs. $\notin A$ (100)	0.56134	0.6	0.5117	0.45443	0.9684	0.1899	0.47454	**0.6696**	0.2885
$\in A$ vs. $\notin A$ (200)	0.54242	0.59701	0.5	0.47975	0.98101	0.20886	0.48189	**0.66955**	0.30415
$\in A$ vs. $\notin A$ (500)	0.55796	0.64815	0.46591	0.37848	0.73418	0	0.45942	**0.61214**	0.33019
$\in A$ vs. $\notin A$ (1000)	0.55925	0.61111	0.50645	0.51013	0.99367	0.18987	0.4951	**0.671**	0.28708
$\in A$ vs. $\notin A$ (1500)	0.53213	0.58824	0.49049	0.48038	1	0.063291	0.45071	**0.66808**	0.11429
$\in A$ vs. $\notin A$ **(2000)**	0.54675	0.6	0.50804	0.52215	1	0	0.5195	**0.67377**	0.24876
$\in B$ vs. $\notin B$ **(20)**	0.55877	0.59286	0.51282	0.48671	0.71519	0.20253	0.51127	**0.62088**	0.29224
$\in B$ vs. $\notin B$ (50)	0.57549	0.64063	0.53788	0.43165	0.62658	0.17722	0.47252	**0.6**	0.27451
$\in B$ vs. $\notin B$ (100)	0.56327	0.6087	0.53179	0.49873	0.70253	0.26582	0.51798	**0.61326**	0.37004
$\in B$ vs. $\notin B$ (200)	0.56892	0.6875	0.52903	0.49494	0.72785	0.06962	0.4981	**0.61497**	0.12644
$\in B$ vs. $\notin B$ (500)	0.58142	0.8125	0.53285	0.44684	0.61392	0.082278	0.47743	**0.57227**	0.14943
$\in B$ vs. $\notin B$ (1000)	0.63892	1	0.5493	0.4038	0.61392	0.018987	0.43915	**0.58788**	0.037267
$\in C$ vs. $\notin C$ (20)	0.53118	0.57823	0.5	0.5038	0.91139	0.21519	0.48867	**0.64574**	0.3105
$\in C$ vs. $\notin C$ (50)	0.54137	0.60417	0.48927	0.42911	0.72152	0.18354	0.46051	**0.58312**	0.28155
$\in C$ vs. $\notin C$ (100)	0.53704	0.575	0.495	0.41899	0.67722	0.29114	0.45827	**0.60452**	0.38655
$\in C$ vs. $\notin C$ **(200)**	0.52213	0.58696	0.47312	0.44747	0.98734	0.17089	0.44954	**0.66667**	0.26471
$\in C$ vs. $\notin C$ **(400)**	0.52636	0.60294	0.49107	0.53671	1	0.11392	0.49099	**0.66667**	0.19048
$\in D$ vs. $\notin D$ (20)	0.63211	0.70423	0.54688	0.3943	0.58861	0.31646	0.47752	**0.58125**	0.43668
$\in D$ vs. $\notin D$ (50)	0.59654	0.63736	0.54595	0.43987	0.63924	0.36709	0.50097	**0.58892**	0.46586
$\in D$ vs. $\notin D$ **(100)**	0.60165	0.67568	0.54592	0.44304	0.67722	0.29747	0.49769	**0.60452**	0.4
$\in D$ vs. $\notin D$ (150)	0.59275	0.63636	0.5625	0.42595	0.53165	0.31013	0.4907	**0.54902**	0.41525
$\in D$ vs. $\notin D$ (200)	0.62366	0.68889	0.52604	0.4	0.63924	0.1962	0.4684	**0.59509**	0.30542

Table A.1.4: Results when analyzing only subset a) of prelabelled publications from set a)-c). Mean (M), best (B) and worst (W) performance (measured by precision, recall and F-measure) out of 10 random restarts for each classifier using 2500 iterations. **(Using 5 or 6 different number of neurons (# of neurons in parenthesis in the leftmost column)). For full results (for all classes) we refer the reader to appendix A

Bin. Class. (#**)	Pre. (M)	(B)	(W)	Rec. (M)	(B)	(W)	F (M)	(B)	(W)
$\in A$ vs. $\notin A$ (20)	0.68429	0.77778	0.6087	0.60435	0.82609	0.30435	0.62346	**0.7037**	0.4375
$\in A$ vs. $\notin A$ **(50)**	0.64711	0.81818	0.53659	0.73043	0.95652	0.3913	0.66967	**0.72727**	0.52941
$\in A$ vs. $\notin A$ (100)	0.64088	0.72727	0.58621	0.62174	0.91304	0.34783	0.61631	**0.72414**	0.44444
$\in A$ vs. $\notin A$ (200)	0.65582	1	0.525	0.5913	0.91304	0.13043	0.56402	**0.71698**	0.23077
$\in A$ vs. $\notin A$ (400)	0.65406	1	0.52381	0.61739	0.95652	0.043478	0.57228	**0.69231**	0.083333
$\in B$ vs. $\notin B$ **(20)**	0.65015	0.76471	0.6	0.68696	0.91304	0.47826	0.6566	**0.73684**	0.5641
$\in B$ vs. $\notin B$ (50)	0.66725	1	0.56	0.56087	0.78261	0.086957	0.56679	**0.69388**	0.16
$\in B$ vs. $\notin B$ (100)	0.70549	1	0.58621	0.52174	0.78261	0.17391	0.56392	**0.69231**	0.2963
$\in B$ vs. $\notin B$ (200)	0.66647	0.90909	0.51351	0.63043	0.86957	0.34783	0.62335	**0.71429**	0.48485
$\in B$ vs. $\notin B$ (400)	0.67023	0.8	0.57143	0.53913	0.78261	0.17391	0.5671	**0.72**	0.28571
$\in B$ vs. $\notin B$ (1000)	0.63892	1	0.5493	0.4038	0.61392	0.018987	0.43915	**0.58788**	0.037267
$\in C$ vs. $\notin C$ (20)	0.7371	0.84615	0.57143	0.62273	0.90909	0.40909	0.65811	**0.73469**	0.52941
$\in C$ vs. $\notin C$ **(50)**	0.69436	0.76471	0.60714	0.65	0.77273	0.40909	0.66279	**0.75556**	0.52941
$\in C$ vs. $\notin C$ **(100)**	0.73195	0.9	0.57692	0.54545	0.77273	0.31818	0.60481	**0.75556**	0.45161
$\in C$ vs. $\notin C$ (200)	0.77207	1	0.58333	0.51818	0.68182	0.18182	0.59575	**0.71429**	0.30769
$\in C$ vs. $\notin C$ (400)	0.73615	0.88889	0.6087	0.54091	0.81818	0.31818	0.59907	**0.75**	0.45161
$\in D$ vs. $\notin D$ (20)	0.72638	0.88889	0.59091	0.57727	0.86364	0.31818	0.61841	**0.71698**	0.46667
$\in D$ vs. $\notin D$ **(50)**	0.72471	1	0.57895	0.58182	0.86364	0.18182	0.60952	**0.8**	0.30769
$\in D$ vs. $\notin D$ (100)	0.71237	0.83333	0.54545	0.63636	0.86364	0.40909	0.65193	**0.76**	0.54545
$\in D$ vs. $\notin D$ (150)	0.69158	0.81818	0.51163	0.60909	1	0.31818	0.62221	**0.76596**	0.4375
$\in D$ vs. $\notin D$ (200)	0.7611	0.88889	0.625	0.52273	0.72727	0.36364	0.60524	**0.74419**	0.51613

Table A.1.5: Results when analyzing only subset b) of prelabelled publications from set a)-c). Mean (M), best (B) and worst (W) performance (measured by precision, recall and F-measure) out of 10 random restarts for each classifier using 2500 iterations. **(Using 5 or 6 different number of neurons (# of neurons in parenthesis in the leftmost column)). For full results (for all classes) we refer the reader to appendix A

Bin. Class. (#**)	Pre. (M)	(B)	(W)	Rec. (M)	(B)	(W)	F (M)	(B)	(W)
$\in A$ vs. $\notin A$ (20)	0.62629	0.66667	0.59322	0.51064	0.64894	0.37234	0.55731	**0.63212**	0.45752
$\in A$ vs. $\notin A$ (50)	0.61842	0.67213	0.5618	0.58511	0.74468	0.43617	0.59679	**0.66038**	0.52903
$\in A$ vs. $\notin A$ (100)	0.598	0.64286	0.57759	0.55426	0.71277	0.42553	0.57178	**0.6381**	0.50633
$\in A$ vs. $\notin A$ (200)	0.59641	0.64865	0.55172	0.57872	0.85106	0.40426	0.57764	**0.66946**	0.49351
$\in A$ vs. $\notin A$ **(400)**	0.59535	0.66667	0.5461	0.6383	0.85106	0.45745	0.60623	**0.67511**	0.53416
$\in B$ vs. $\notin B$ **(20)**	0.61717	0.66265	0.53896	0.59158	0.87368	0.41053	0.59478	**0.66667**	0.46988
$\in B$ vs. $\notin B$ (50)	0.5847	0.6506	0.53543	0.57263	0.71579	0.30526	0.57092	**0.6381**	0.40278
$\in B$ vs. $\notin B$ **(100)**	0.58495	0.65278	0.5274	0.64526	0.87368	0.48421	0.6055	**0.66667**	0.54118
$\in B$ vs. $\notin B$ (200)	0.60603	0.66667	0.57447	0.56737	0.64211	0.44211	0.58432	**0.62887**	0.50909
$\in B$ vs. $\notin B$ (400)	0.58777	0.63218	0.54962	0.58211	0.75789	0.42105	0.579	**0.63717**	0.4878
$\in C$ vs. $\notin C$ **(20)**	0.59916	0.63158	0.56296	0.67474	0.8	0.56842	0.6323	**0.689**	0.58378
$\in C$ vs. $\notin C$ (50)	0.5911	0.6375	0.56897	0.59579	0.75789	0.44211	0.58913	**0.66359**	0.5
$\in C$ vs. $\notin C$ (100)	0.59814	0.65753	0.56667	0.59895	0.72632	0.46316	0.59401	**0.65094**	0.51462
$\in C$ vs. $\notin C$ (200)	0.59869	0.62887	0.56481	0.62	0.72632	0.43158	0.60588	**0.65714**	0.49697
$\in C$ vs. $\notin C$ (400)	0.60477	0.63	0.57732	0.57579	0.67368	0.36842	0.58492	**0.64975**	0.46358
$\in D$ vs. $\notin D$ (20)	0.60675	0.66667	0.52874	0.55684	0.72632	0.32632	0.57127	**0.65714**	0.43056
$\in D$ vs. $\notin D$ **(50)**	0.58529	0.62727	0.55319	0.67684	0.82105	0.55789	0.62486	**0.67568**	0.57297
$\in D$ vs. $\notin D$ (100)	0.57019	0.63953	0.53846	0.65053	0.74737	0.54737	0.60556	**0.64815**	0.55026
$\in D$ vs. $\notin D$ (150)	0.59853	0.65957	0.55147	0.54	0.78947	0.29474	0.54869	**0.64935**	0.39716
$\in D$ vs. $\notin D$ (200)	0.57625	0.63333	0.54082	0.59474	0.73684	0.4	0.57906	**0.64516**	0.49032

Table A.1.6: Results when analyzing only subset c) of prelabelled publications from set a)-c). Mean (M), best (B) and worst (W) performance (measured by precision, recall and F-measure) out of 10 random restarts for each classifier using 2500 iterations. **(Using 5 different number of neurons (# of neurons in parenthesis in the leftmost column)). For full results (for all classes) we refer the reader to appendix A

Bin. Class. (#**)	Pre. (M)	(B)	(W)	Rec. (M)	(B)	(W)	F (M)	(B)	(W)
$\in A$ vs. $\notin A$ (20)	0.55368	0.5814	0.53824	0.55418	0.69455	0.34909	0.54663	**0.60828**	0.42953
$\in A$ vs. $\notin A$ (50)	0.54668	0.55705	0.52229	0.61164	0.75273	0.36	0.57193	**0.62727**	0.43709
$\in A$ vs. $\notin A$ (100)	0.55116	0.57991	0.53846	0.54545	0.68727	0.29455	0.53841	**0.60772**	0.3848
$\in A$ vs. $\notin A$ (150)	0.54559	0.56725	0.52861	0.58509	0.70545	0.35273	0.55846	**0.60883**	0.43498
$\in A$ vs. $\notin A$ **(200)**	0.5453	0.5988	0.5	0.61055	1	0.36364	0.56382	**0.66667**	0.45249
$\in B$ vs. $\notin B$ (20)	0.54675	0.59184	0.49785	0.54545	0.69818	0.42182	0.54346	**0.61244**	0.45669
$\in B$ vs. $\notin B$ (50)	0.54095	0.60106	0.50353	0.58291	0.77818	0.41091	0.55292	**0.61143**	0.47316
$\in B$ vs. $\notin B$ (100)	0.54908	0.58407	0.50562	0.50255	0.72	0.25818	0.51531	**0.60366**	0.355
$\in B$ vs. $\notin B$ **(150)**	0.53099	0.58898	0.5	0.60545	1	0.40727	0.55434	**0.66667**	0.4489
$\in B$ vs. $\notin B$ **(200)**	0.55428	0.58261	0.5	0.544	1	0.24364	0.53144	**0.66667**	0.34359
$\in C$ vs. $\notin C$ (20)	0.55142	0.59799	0.51316	0.58618	0.71273	0.41455	0.56298	**0.61732**	0.46914
$\in C$ vs. $\notin C$ (50)	0.54203	0.56115	0.51527	0.612	0.72727	0.49091	0.57313	**0.6192**	0.50279
$\in C$ vs. $\notin C$ **(100)**	0.52667	0.575	0.33333	0.592	0.85455	0.0072727	0.53512	**0.64384**	0.014235
$\in C$ vs. $\notin C$ (150)	0.5359	0.55844	0.51493	0.59964	0.75273	0.46545	0.56301	**0.61152**	0.49612
$\in C$ vs. $\notin C$ (200)	0.5504	0.56934	0.51636	0.57527	0.64727	0.45818	0.56168	**0.59933**	0.49412
$\in D$ vs. $\notin D$ (20)	0.5364	0.56566	0.51358	0.56836	0.75636	0.37455	0.54447	**0.61176**	0.44397
$\in D$ vs. $\notin D$ (50)	0.52918	0.55276	0.50206	0.54436	0.69818	0.4	0.5328	**0.60377**	0.46414
$\in D$ vs. $\notin D$ (100)	0.52968	0.54658	0.51449	0.56582	0.77455	0.32	0.53907	**0.61829**	0.40367
$\in D$ vs. $\notin D$ **(150)**	0.52822	0.55372	0.51071	0.53455	0.76364	0.35636	0.52562	**0.62222**	0.43267
$\in D$ vs. $\notin D$ (200)	0.53836	0.55814	0.51812	0.46764	0.65091	0.28364	0.49541	**0.58592**	0.36967

Table A.1.7: Results when analyzing prelabelled publications from set a)-c) without applying stemming. Mean (M), best (B) and worst (W) performance (measured by precision, recall and F-measure) out of 10 random restarts for each classifier using 2500 iterations. **(Using 5 or 6 different number of neurons (# of neurons in parenthesis in the leftmost column)). For full results (for all classes) we refer the reader to appendix A

Bin. Class. (#**)	Pre. (M)	(B)	(W)	Rec. (M)	(B)	(W)	F (M)	(B)	(W)
$\in A$ vs. $\notin A$ (20)	0.55732	0.58228	0.53179	0.58038	0.81013	0.38608	0.56027	**0.6514**	0.46212
$\in A$ vs. $\notin A$ **(50)**	0.56585	0.59804	0.5445	0.5962	0.77848	0.38608	0.57332	**0.65252**	0.46923
$\in A$ vs. $\notin A$ (100)	0.56645	0.60417	0.54545	0.57722	0.77215	0.36709	0.56419	**0.64042**	0.45669
$\in A$ vs. $\notin A$ (150)	0.5595	0.59124	0.53555	0.58481	0.77848	0.36709	0.5641	**0.63896**	0.44615
$\in A$ vs. $\notin A$ (200)	0.56446	0.59615	0.53012	0.58544	0.73418	0.38608	0.56752	**0.63043**	0.46212
$\in B$ vs. $\notin B$ (20)	0.55708	0.59859	0.52432	0.59182	0.75472	0.44654	0.5685	**0.63612**	0.49825
$\in B$ vs. $\notin B$ (50)	0.54149	0.57714	0.51931	0.66289	0.76101	0.58491	0.59467	**0.6413**	0.55193
$\in B$ vs. $\notin B$ **(100)**	0.53223	0.56863	0.50185	0.69623	0.98113	0.53459	0.59591	**0.66953**	0.53583
$\in B$ vs. $\notin B$ (150)	0.55213	0.60194	0.50336	0.60943	0.9434	0.26415	0.56106	**0.65904**	0.35146
$\in B$ vs. $\notin B$ (200)	0.55049	0.61261	0.50645	0.63836	0.98742	0.42767	0.57825	**0.66951**	0.4947
$\in C$ vs. $\notin C$ (20)	0.56639	0.60345	0.53299	0.56266	0.66456	0.44304	0.56163	**0.60534**	0.51034
$\in C$ vs. $\notin C$ (50)	0.55357	0.5935	0.52273	0.57595	0.77848	0.22785	0.54985	**0.63361**	0.32
$\in C$ vs. $\notin C$ (100)	0.55572	0.57554	0.47644	0.57152	0.72785	0.3481	0.55821	**0.62842**	0.43307
$\in C$ vs. $\notin C$ (150)	0.54618	0.57576	0.52857	0.53987	0.78481	0.36076	0.53263	**0.6359**	0.43346
$\in C$ vs. $\notin C$ **(200)**	0.548	0.57297	0.51685	0.67089	0.87975	0.41139	0.59703	**0.65566**	0.47445
$\in D$ vs. $\notin D$ (20)	0.5339	0.60606	0.49254	0.53101	0.75949	0.12658	0.51106	**0.62663**	0.20942
$\in D$ vs. $\notin D$ (50)	0.5384	0.56471	0.51064	0.47342	0.75949	0.22152	0.48171	**0.62663**	0.31532
$\in D$ vs. $\notin D$ **(100)**	0.538	0.57407	0.50211	0.50696	0.90506	0.29114	0.50189	**0.65297**	0.38017
$\in D$ vs. $\notin D$ (150)	0.55261	0.58036	0.51613	0.45	0.70886	0.25316	0.4823	**0.59733**	0.34632
$\in D$ vs. $\notin D$ (200)	0.53828	0.56311	0.51462	0.53608	0.75949	0.29747	0.52652	**0.61538**	0.38843

Table A.1.8: Results when analyzing only subset a) of prelabelled publications from set a)-c) without stemming. Mean (M), best (B) and worst (W) performance (measured by precision, recall and F-measure) out of 10 random restarts for each classifier using 2500 iterations. **(Using 5 different number of neurons (# of neurons in parenthesis in the leftmost column)). For full results (for all classes) we refer the reader to appendix A

Bin. Class. (#**)	Pre. (M)	(B)	(W)	Rec. (M)	(B)	(W)	F (M)	(B)	(W)
$\in A$ vs. $\notin A$ (20)	0.6402	0.75	0.55556	0.6	0.86957	0.34783	0.60489	**0.67797**	0.47059
$\in A$ vs. $\notin A$ (50)	0.63036	0.72222	0.53333	0.6087	0.78261	0.43478	0.60955	**0.69388**	0.52632
$\in A$ vs. $\notin A$ (100)	0.65749	0.72222	0.57143	0.6087	0.86957	0.3913	0.62209	**0.68966**	0.5
$\in A$ vs. $\notin A$ (150)	0.64832	0.76923	0.55172	0.62609	0.73913	0.43478	0.62926	**0.70833**	0.55556
$\in A$ vs. $\notin A$ **(200)**	0.66158	0.8	0.57895	0.64783	0.95652	0.43478	0.63939	**0.72131**	0.51282
$\in B$ vs. $\notin B$ (20)	0.55609	0.66667	0.47826	0.55238	0.7619	0.42857	0.54586	**0.61538**	0.48649
$\in B$ vs. $\notin B$ (50)	0.54805	0.6	0.51613	0.58571	0.7619	0.38095	0.55455	**0.64**	0.44444
$\in B$ vs. $\notin B$ (100)	0.53123	0.66667	0.46667	0.52381	0.85714	0.19048	0.50264	**0.64286**	0.2963
$\in B$ vs. $\notin B$ **(150)**	0.53829	0.61111	0.47368	0.55238	0.80952	0.33333	0.54	**0.66667**	0.4
$\in B$ vs. $\notin B$ (200)	0.5465	0.625	0.47826	0.51905	0.66667	0.2381	0.51543	**0.6087**	0.34483
$\in C$ vs. $\notin C$ **(20)**	0.6296	0.70833	0.57143	0.59091	0.77273	0.18182	0.59495	**0.73913**	0.27586
$\in C$ vs. $\notin C$ (50)	0.61496	0.72727	0.57143	0.66364	1	0.40909	0.62386	**0.73333**	0.48649
$\in C$ vs. $\notin C$ (100)	0.61337	0.85714	0.5	0.63636	0.90909	0.27273	0.59542	**0.72727**	0.41379
$\in C$ vs. $\notin C$ (150)	0.63457	0.70588	0.57576	0.60909	0.86364	0.22727	0.60647	**0.7234**	0.33333
$\in C$ vs. $\notin C$ (200)	0.61988	0.75	0.52	0.6	0.77273	0.13636	0.58614	**0.71111**	0.23077
$\in D$ vs. $\notin D$ (20)	0.63454	0.73333	0.55882	0.60455	0.86364	0.36364	0.60256	**0.68**	0.48485
$\in D$ vs. $\notin D$ (50)	0.58599	0.66667	0.5	0.55455	0.72727	0.13636	0.55553	**0.69565**	0.21429
$\in D$ vs. $\notin D$ (100)	0.58307	0.66667	0.5	0.55455	0.81818	0.36364	0.55723	**0.67925**	0.43243
$\in D$ vs. $\notin D$ **(150)**	0.61811	0.72727	0.56522	0.55909	0.81818	0.36364	0.57545	**0.7234**	0.45714
$\in D$ vs. $\notin D$ (200)	0.60465	0.66667	0.5	0.65909	0.81818	0.45455	0.62191	**0.70833**	0.54054

Table A.1.9: Results when analyzing only subset b) of prelabelled publications from set a)-c) without stemming. Mean (M), best (B) and worst (W) performance (measured by precision, recall and F-measure) out of 10 random restarts for each classifier using 2500 iterations. **(Using 5 different number of neurons (# of neurons in parenthesis in the leftmost column)). For full results (for all classes) we refer the reader to appendix A

Bin. Class. (#**)	Pre. (M)	(B)	(W)	Rec. (M)	(B)	(W)	F (M)	(B)	(W)
$\in A$ vs. $\notin A$ (20)	0.57306	0.63889	0.53704	0.54362	0.7234	0.3617	0.55189	**0.62385**	0.4359
$\in A$ vs. $\notin A$ (50)	0.56822	0.60256	0.51304	0.58511	0.7234	0.42553	0.5726	**0.62673**	0.47904
$\in A$ vs. $\notin A$ (100)	0.58437	0.6	0.56604	0.55745	0.76596	0.35106	0.56441	**0.65455**	0.44295
$\in A$ vs. $\notin A$ (150)	0.58619	0.60759	0.55263	0.54574	0.73404	0.21277	0.55331	**0.65403**	0.31496
$\in A$ vs. $\notin A$ **(200)**	0.57334	0.58696	0.55357	0.6117	0.7766	0.41489	0.58798	**0.65766**	0.48447
$\in B$ vs. $\notin B$ (20)	0.54117	0.59677	0.48889	0.59368	0.74737	0.38947	0.55908	**0.63054**	0.47134
$\in B$ vs. $\notin B$ (50)	0.56717	0.66667	0.5119	0.62947	0.90526	0.4	0.58625	**0.65399**	0.5
$\in B$ vs. $\notin B$ **(100)**	0.55566	0.60606	0.49091	0.63158	0.93684	0.21053	0.56828	**0.66667**	0.3125
$\in B$ vs. $\notin B$ (150)	0.55062	0.59794	0.47945	0.60421	0.76842	0.36842	0.5719	**0.64317**	0.41667
$\in B$ vs. $\notin B$ (200)	0.56047	0.59649	0.49495	0.59158	0.71579	0.23158	0.56841	**0.65072**	0.31884
$\in C$ vs. $\notin C$ **(20)**	0.56664	0.59756	0.52727	0.66105	0.91579	0.50526	0.60185	**0.66923**	0.52747
$\in C$ vs. $\notin C$ (50)	0.55703	0.59091	0.5	0.54632	0.70526	0.010526	0.52155	**0.63415**	0.020619
$\in C$ vs. $\notin C$ (100)	0.57401	0.59223	0.55446	0.62	0.76842	0.44211	0.59224	**0.64889**	0.50299
$\in C$ vs. $\notin C$ (150)	0.55652	0.58621	0.53741	0.69579	0.83158	0.53684	0.61398	**0.6556**	0.56044
$\in C$ vs. $\notin C$ (200)	0.56137	0.59048	0.54264	0.61263	0.76842	0.29474	0.57517	**0.64889**	0.38621
$\in D$ vs. $\notin D$ **(20)**	0.60294	0.66667	0.55372	0.64737	0.74737	0.48421	0.62043	**0.66667**	0.56098
$\in D$ vs. $\notin D$ (50)	0.60236	0.65672	0.54839	0.59474	0.71579	0.35789	0.5925	**0.657**	0.43312
$\in D$ vs. $\notin D$ (100)	0.59366	0.62222	0.56557	0.58737	0.72632	0.43158	0.58649	**0.63594**	0.49697
$\in D$ vs. $\notin D$ (150)	0.60487	0.64063	0.5812	0.62737	0.75789	0.43158	0.61081	**0.66055**	0.51572
$\in D$ vs. $\notin D$ **(200)**	0.61036	0.65714	0.5679	0.59368	0.74737	0.46316	0.59634	**0.66667**	0.52273

Table A.1.10: Results when analyzing only subset c) of prelabelled publications from set a)-c) without stemming. Mean (M), best (B) and worst (W) performance (measured by precision, recall and F-measure) out of 10 random restarts for each classifier. **(Using 5 different number of neurons (# of neurons in parenthesis in the leftmost column)). For full results (for all classes) we refer the reader to appendix A

A.2 Temporal distributions

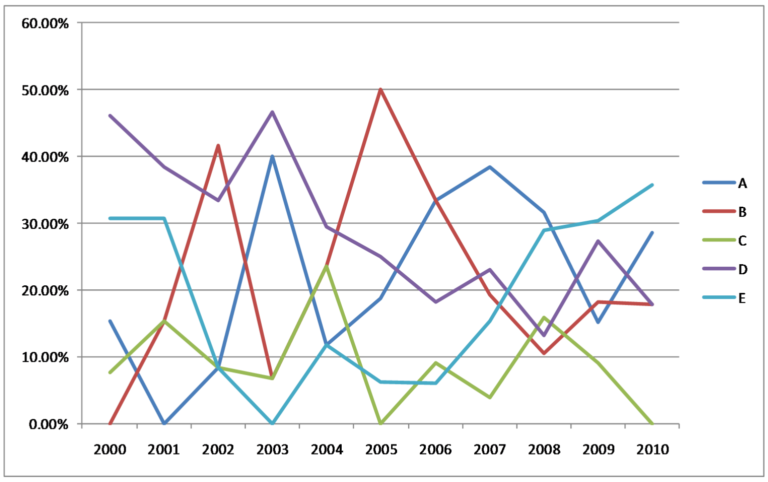


Figure A.2.1: *The relative distribution (per year) of SKFs publications across SI over the studied period (2000-2010)*

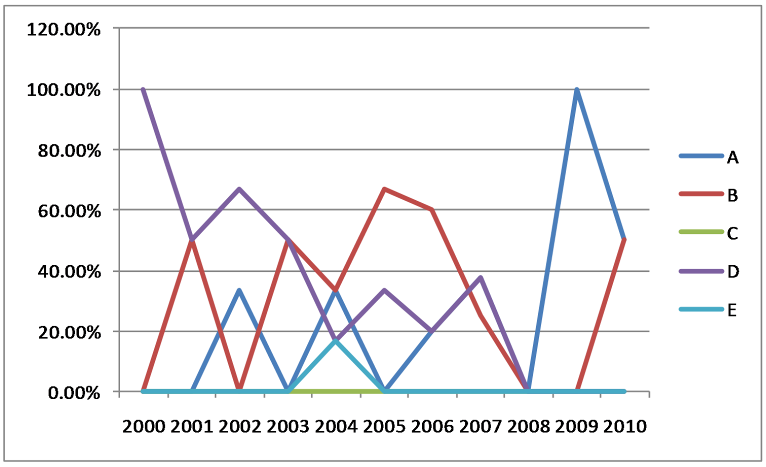


Figure A.2.2: *The relative distribution (per year) of competitor As publications across SI over the studied period (2000-2010)*

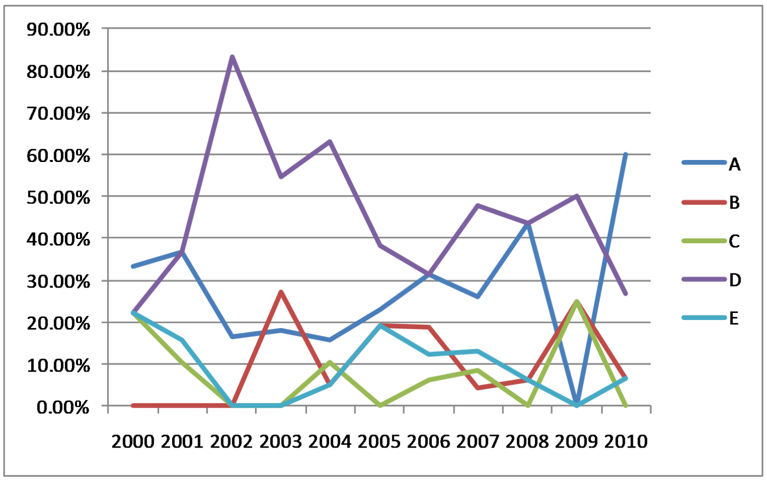


Figure A.2.3: *The relative distribution (per year) of competitor Bs publications across SI over the studied period (2000-2010)*

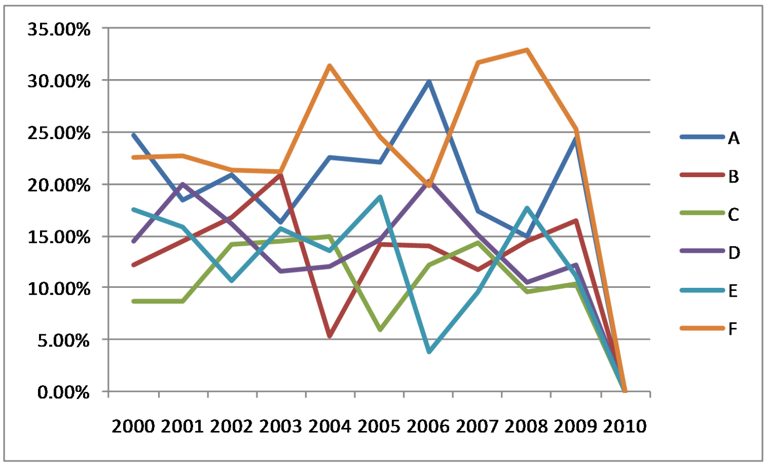


Figure A.2.4: *The relative distribution (per year) of SKFs patents across SI over the studied period (2000-2010)*

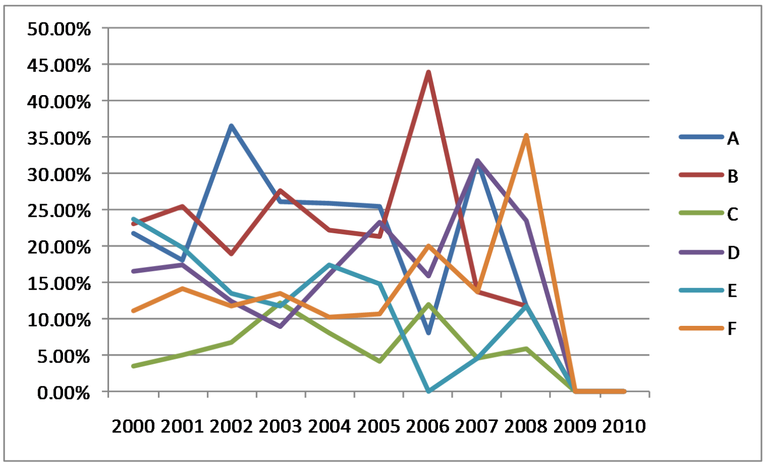


Figure A.2.5: *The relative distribution (per year) of competitor As patents across SI over the studied period (2000-2010)*

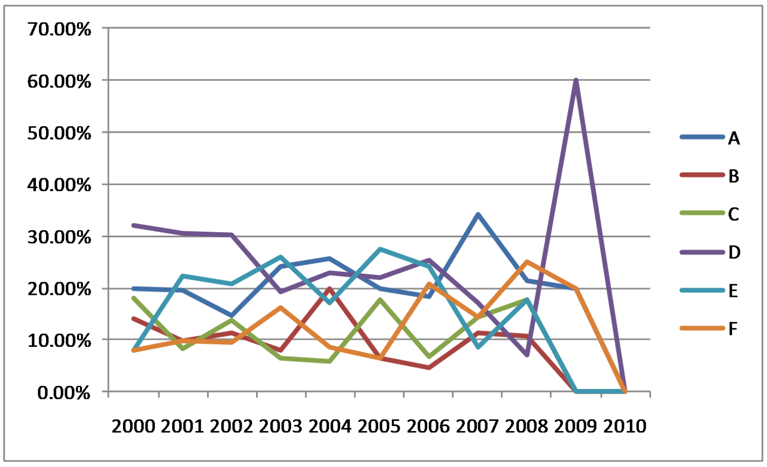


Figure A.2.6: *The relative distribution (per year) of competitor Bs patents across SI over the studied period (2000-2010)*

B Matlab code

The complete uncensored Matlab code is only available internally within SKF, for the importing of text files, see the appendix B.1 below. ... denotes that the line is continued on the next row.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Created by Robin Karmakar, 2011-05-01
%   version 1.0     2011-05-01
%   version 1.1     2011-05-10
%   version 1.2     2011-05-15
%
%   version 2.0     2011-05-25
%   version 2.1     2011-05-31
%% Artificial neural network for text recognition.
% This script preassumes that the loaded data files are available,
%   created using the TMG toolbox. See thesis documentation
%   for explanations.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%%
% Clean up memory
clear

% Otherwise we make use of a loop for each source of data:
for SKF = [0 1 2]%2]%; 0,1,2

    % In the case of SKF
    if SKF == 0
        %2 skf :317
        % Load saved dataset
        load all_SI_5_SKF_A500.mat
        A300 = A500;

        % Load saved pre-classifications
        load classification4_SKF.mat

        % Set the corresponding data ranges
        start=2;
        final=317;

    % In the case of competitor A
    elseif SKF == 1
        %318 fag :361
        % Load saved dataset
        load all_SI_5_FAG_A500.mat
        A300 = A500;

        % Load saved pre-classifications
        load classification4_FAG.mat

        % Set the corresponding data ranges
        start=1;
        final=44;

    % In the case of competitor B
    elseif SKF == 2
        %362 timken :end
        % Load saved dataset
        load all_SI_5_Timken_A500.mat
        A300 = A500;

        % Load saved pre-classifications
        load classification4_Timken.mat

        % Set the corresponding data ranges
        start=1;
        final=190;

    % In the case of using the full set
    elseif SKF == -1
        start=2;
        final=551;
    end

    % Set the analyzed matrix to the appropriate pre-classified documents
    inputs = A300(:,start:final);

    % Loop for different classes (binary classification case)
    %   1=A, 2=B, etc.
    for k=[1 2 3 4 5 6] %7 8 9 10]
        % Check progress
        k

        % Set the ANN targets as the pre-classifications for the set
        if k==1
            targets = classificationA(start:final);
        elseif k==2
            targets = classificationB(start:final);
        elseif k==3
            targets = classificationC(start:final);
        elseif k==4
            targets = classificationD(start:final);
        elseif k==5
            targets = classificationE(start:final);
        elseif k==6
            targets = classificationF(start:final);
        elseif k==7
            targets = classificationG(start:final);
        elseif k==8
            targets = classificationH(start:final);
        elseif k==9
            targets = classificationI(start:final);
        elseif k==10
            targets = classificationJ(start:final);
        end

        % Prepare Output,
        %   in the case of multiple numbers of hidden neurons
        %   or simply the case with 20 neurons in the hidden layer
        Results{2,1} = '20';
        %Results{3,1} = '50';
        %Results{4,1} = '100';
```

```
%Results{5,1} = '150';
%Results{6,1} = '200';

% Create the set of # of neurons to iterate through
num=[1,20,50,100,150,200];

% Create Network
for j=3:3 % For which #of neurons in 'num' do we do the experiments
    fBest = 0;  % Reset checker for the best performing ANN
    for i=1:5   % Set number of restarts (to correct for
                %   distributional differences

        % Set the number of hidden neurons
        numHiddenNeurons = num(j)

        % Check progress when running experiments
        i
        %% Create artificial neural network
        net = patternnet(numHiddenNeurons,'trainscg');
        net.divideParam.trainRatio = 100/100;   % Adjust as desired
        net.divideParam.valRatio = 0/100;       % Adjust as desired
        net.divideParam.testRatio = 0/100;      % Adjust as desired

        % Training parameters
        net.trainParam.epochs = 10000;  % 500, 2500, 5000, 100000
        net.trainParam.goal = 0.001;    % 0.000001

        % Train and Apply Network
        [net,tr] = train(net,inputs,targets);
        outputs = net(inputs);

        % Plot resulting values (uncomment if needed)
        %plotperf(tr)
        %plotconfusion(targets,outputs)

        %% Performance measurements

        % Uses the confusion matrix parameters to calculate
        %   performance measures
        [c,cm,ind,per] = confusion(targets,outputs);
        precision = cm(1,1)/(cm(1,1)+cm(2,1))
        recall = cm(1,1)/(cm(1,1)+cm(1,2))
        F = 2*precision*recall/(precision + recall)

        % Check the performance of the network and save the best
        %   performing ANN produced.
        if fBest<F
            fBest = max(fBest,F)
            netBest = net;
        end

        % Prepare and save cell-matrix structure with results
        Results{j,2+(i-1)*3} = precision;
        Results{j,3+(i-1)*3} = recall;
        Results{j,4+(i-1)*3} = F;
        Results{1,2+(i-1)*3} = 'pre';
```

```
            Results{1,3+(i-1)*3} = 'rec';
            Results{1,4+(i-1)*3} = 'F';
        end
    end

    % Save the corresponding results for SKF
    if SKF == 0
        if k==1
            Results6A_SKF = Results;
            save('Results6A_SKF','Results6A_SKF');
            save('SKFnetBestA','netBest');
        elseif k==2
            Results6B_SKF = Results;
            save('Results6B_SKF','Results6B_SKF');
            save('SKFnetBestB','netBest');
        elseif k==3
            Results6C_SKF = Results;
            save('Results6C_SKF','Results6C_SKF');
            save('SKFnetBestC','netBest');
        elseif k==4
            Results6D_SKF = Results;
            save('Results6D_SKF','Results6D_SKF');
            save('SKFnetBestD','netBest');
        elseif k==5
            Results6E_SKF = Results;
            save('Results6E_SKF','Results6E_SKF');
            save('SKFnetBestE','netBest');
        elseif k==6
            Results6F_SKF = Results;
            save('Results6F_SKF','Results6F_SKF');
            save('SKFnetBestF','netBest');
        elseif k==7
            Results6G_SKF = Results;
            save('Results6G_SKF','Results6G_SKF');
            save('SKFnetBestG','netBest');
        elseif k==8
            Results6H_SKF = Results;
            save('Results6H_SKF','Results6H_SKF');
            save('SKFnetBestH','netBest');
        elseif k==9
            Results6I_SKF = Results;
            save('Results6I_SKF','Results6I_SKF');
            save('SKFnetBestI','netBest');
        elseif k==10
            Results6J_SKF = Results;
            save('Results6J_SKF','Results6J_SKF');
            save('SKFnetBestJ','netBest');
        end
    end

    % Save the corresponding results for competitor A
    if SKF == 1
        if k==1
CENSORED
        end
    end
```

```
        % Save the corresponding results for competitor B
        if SKF == 2
            if k==1
     CENSORED
            end
        end
    end
end
```

B.1 Importing and creating text document matrix using TMG

In this section we present the Matlab script to import and produce the given TDM representations of the documents, which were used for the document analysis and classifications. The documents are fed into the TMG (see appendix E.2) which generates the term-document representation, together with the corresponding weighting as described in chapter 5. ... denotes that the line is continued on the next row. The documentation regarding the TMG function, which is a part of the TMG toolbox, has been sourced from the help section of the TMG Toolbox, see reference [DZ11].

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Created by Robin Karmakar, 2011-04-28
%   version 1.0     2011-04-28
%   version 1.1     2011-05-06
%   version 1.2     2011-05-13
%
%% Import of data from .txt file using the TMG toolbox in Matlab
% This script preassumes that the loaded data files are available,
%   created using the TMG toolbox. See thesis documentation
%   for explanations.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%% TMG Help details:
%  TMG(FILENAME, OPTIONS) defines optional parameters:
%         - OPTIONS.use_mysql: Indicates if results are to be
%           stored in MySQL.
%         - OPTIONS.db_name: The name of the directory where
%           the results are to be saved.
%         - OPTIONS.delimiter: The delimiter between documents
%           within the same file. Possible values are 'emptyline'
%           (default), 'none_delimiter' (treats each file as a
%           single document) or any other string.
%         - OPTIONS.line_delimiter: Defines if the delimiter
%           takes a whole line of text (default, 1) or not.
%         - OPTIONS.stoplist: The filename for the stoplist,
%           i.e. a list of common words that we don't use for
%           the indexing (default no stoplist used).
%         - OPTIONS.stemming: Indicates if the stemming algorithm
%           is used (1) or not (0 - default).
%         - OPTIONS.update_step: The step used for the incremental
%           built of the inverted index (default 10,000).
%         - OPTIONS.min_length: The minimum length for a term
%           (default 3).
%         - OPTIONS.max_length: The maximum length for a term
%           (default 30).
%         - OPTIONS.min_local_freq: The minimum local frequency for
%           a term (default 1).
%         - OPTIONS.max_local_freq: The maximum local frequency for
%           a term (default inf).
%         - OPTIONS.min_global_freq: The minimum global frequency
%           for a term (default 1).
%         - OPTIONS.max_global_freq: The maximum global frequency
%           for a term (default inf).
%         - OPTIONS.local_weight: The local term weighting function
%           (default 't'). Possible values (see [1, 2]):
%                 't': Term Frequency
%                 'b': Binary
%                 'l': Logarithmic
%                 'a': Alternate Log
%                 'n': Augmented Normalized Term Frequency
%         - OPTIONS.global_weight: The global term weighting function
%           (default 'x'). Possible values (see [1, 2]):
%                 'x': None
%                 'e': Entropy
%                 'f': Inverse Document Frequency (IDF)
%                 'g': GfIdf
%                 'n': Normal
%                 'p': Probabilistic Inverse
%         - OPTIONS.normalization: Indicates if we normalize the
%           document vectors (default 'x'). Possible values:
%                 'x': None
%                 'c': Cosine
%         - OPTIONS.dsp: Displays results (default 1) or not (0) to
%           the command window.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Run the TMG document import and create the TMG

% Set options for the TMG parsing
options.stoplist = 'C:\Documents and Settings\wh5464\My Documents\Dropbox\ChalmersThesis\TMG\ ...
common_words_edit_2';
options.stemming = 0;
options.min_length = 0;
options.max_length = 30;
options.min_local_freq = 0;
options.max_local_freq = inf;
options.min_global_freq = 0;
options.max_global_freq = inf;
options.local_weight = 't';
options.global_weight = 'f';
options.normalization = 'c';

[A, dictionary, global_weights, normalization_factors, ...
    words_per_doc, titles, files] = TMG('C:\Documents and Settings\wh5464\My Documents\ ...
Dropbox\ChalmersThesis\Data\Raw Data\Patents\SKF_ready.txt',options);
```

C Implementing neural networks

For the layman neural networks may seem as a black-box system that has no real intuitive explanation. When delving deeper into the theory of NNs the procedure of how the BPNN functions becomes more clear. Given the clear structure of how the BPNN algorithm works, together with its simple description, the algorithm is quite accessible. The need for a consistent result using a neural network may be a concern, but the resulting pattern recognition performance should be the measure against which the paradigm should be evaluated. Artificial neural networks have been and are being used for many interesting tasks, and specifically within pattern recognition, as described in the literature [CDM08].

For an implementation in C++ of a back-propagation neural network, we refer the reader to reference [Chh06]. For details concerning neural networks and similar questions on the details of back-propagation, learning algorithms and similar we refer the reader to [Sar02]. These together serve as a very good basis for anyone interested in implementing a neural network on their own. Since this thesis works focus was on developing the process and not the specific learning algorithm the code for the implementation has been left out, since the ”Neural network toolbox” available in Matlab was used for the purposes of our analysis and experiments.

For the interested reader there are a vast amount of resources concerning neural networks online. The basics for the specific algorithm is most suitably found in a introductory text such as [Roj]. Regarding more in-depth topics we suggest that the reader familiarize themselves with a more advanced textbook on the subject.

D Classification schemes

D.1 Patent classifications

D.1.1 IPC

IPC is an abbreviation for the International Patent Classification. The IPC was developed by the EPO and the *World intellectual property organization* (WIPO) and is used globally to classify patent documents. This classification was created under the Strasbourg Agreement of 1971 [WIP09] and is updated regularly to reflect new areas and technologies not yet classfied. This system consists of ”core” levels and ”advanced” sub-levels that together constitute the full framework. Each document that arrives and is examined by most of the patent offices around the world is given a main class, and furthermore the examiner tries to classify the document as far down in the hierarchical system as possible [WIP09].

Almost all published patents from any patent office has at least one IPC code. There is also the possibility that multiple codes are assigned to one single document that each describe different aspects of the document. When there are multiple IPC codes for one document there is no inherent ranking among the ordering of the IPC codes [WIP09].

D.1.2 ECLA

ECLA is an abbreviation for European Classification. This classification was developed by the EPO to improve upon the IPC classification system. Some of the benefits of the system is that they use experienced patent examiners to classify the patents according to ECLA. In comparison for the IPC documents are sometimes not perfectly classified. In contrast it may be said that ECLA codes are a purely computer based system that is not published together with the documents, since these may be given their code as late as several months after the main publication date [Lan09a].

D.1.3 US codes

US codes are the basic classification codes used by the US patent office. This classification is only done for applications to the US patent office, but still is very important due to the amount of patenting in the US. The US classification has been used since the 19th century and a conversion from US classification into IPC classes is available at the USPTO website, but this should not be relied upon entirely [Lan09e].

D.1.4 F-term codes

F-term codes are the corresponding classification scheme that the Japanese patent office uses. F-terms provide a different view on patent documents. This approach focuses on themes, in the sense that a patent document can and may be related to more than one theme. An example may be sports and material needed for sports, where one document may be about soccer, and the needed gear for soccer, but this gear can also be used in other sports, e.g. land-hockey. Typically one would use the F-terms as a ”descriptive area in which this invention may be used” using general terms [Lan09c].

D.1.5 INPADOC

INPADOC families are a grouping across different countries of the same patent application. INPADOC stands for International Patent Documentation Center, and is an international patent collection. It is managed by the EPO and contains information regarding the which family the patents belong to together with information about the legal status of the documents. Thus when one patent application is filed at different patent offices, it may be granted in more than one country, resulting in mutliple patents. There can be one US patent and one German for instance, each classified by their respective local patent numbers. This is the information that is contained within the INPADOC families, so that each single invention (or application to different countries based on the same invention from the same applicant) is grouped to the same INPADOC family [Lan09b].

D.1.6 The Derwent system

The Derwent system is a "value-adding" information system to restructure and refine the information available in patent documents. The Derwent system aims to rewrite the abstracts of patents to add information and use a common language to describe what the patent actually is about. This added information together with the addition of a classification according to the Derwent system gives two specific fields available in the Thomson Innovation patent database [Lan09d]. An important aspect of the Derwent system is that they have combined information regarding different company names as assignees of the patents into company codes. They also re-write the abstracts of the patent documents to more specifically reflect the invention and supply a separate classification system using so called Derwent manual codes.

Derwent abstracts

Derwent abstracts are rewritten patent abstracts by scientists and analysts working at Thomson Innovation. Derwent abstracts give a more direct explanation of the patent documents that is more understandable than the original texts. These abstracts use a standardized language that is common to the staff, and thus attempts to "clean up" the "lawyer english" that may be prevalent in patent documents upon filing. This system aims to clarify and make clear the invention behind the patent so that novelty and innovation searching is done more easily. A very important characteristic of the Derwent data is that it is only done once per INPADOC family, this information is then added to all patent documents within that INPADOC family. In essence this means that the US and German patent applications will be given a common english-language abstract that is as clear as possible [Lan09d].

Derwent manual codes

Derwent manual codes are a classification system used by the Thomson Derwent patent analyst team to classify patents. The philosophy behind the Derwent system is different from for instance the IPC. Manual codes are assigned based on a hierarchical structure, thus the super-classes will comprise the underlying classes. A benefit, which also could be a problem is that the Manual codes are assigned in conjunction with the abstract rewriting, and thus each patent application in one INPADOC family is only classified once. This means that in other cases, different applications in the same INPADOC family could be classified into different IPC classes by different patent examiners in different patents offices around the world. Manual codes also focus on the "inventive content" in the "basic" document (the first published document in the INPADOC family). Therefore they are not focusing on the main topic of the document as a whole, but rather is only considering the inventive part of the document as such. For novelty searching however and innovation searching this may be a preferable feature [Lan09d].

D.2 Publication classifications

D.2.1 Inspec

Inspec is a database for scientific publications and papers across mainly engineering areas. More specifically computer science, physics, mechanical engineering, communications and manufacturing to mention a few. These are indexed and are searchable by using a classification system that was developed by Inspec. While the Inspec codes are well assigned to the content of the documents that are indexed, the index is very generic and does not provide a very firm basis for classifying documents on a more specific level [The11a].

D.2.2 Compendex

Compendex is a database for scientific publications focusing on the field of engineering. The database has international covers around 2600 journals conference papers, proceedings and technical reports [ULS07]. The Compendex articles are also indexed using classification schemes by some data providers. These classification schemes only concern themselves with the general scientific topic the article or publication is about. This also means that for a more specific classification such as being applied to a pattern recognition framework these codes do not serve a purpose.

E Data cleaning

E.1 Porter's stemming algorithm

The most prominent stemming algorithm used for information retrieval purposes was proposed in 1980 by Martin Porter [Por80] in the article "An algorithm for suffix stripping.". The *Porter stemming algorithm* (PSA) is in his own words a:

> "... is a process for removing the commoner morphological and inflexional endings from words in English. Its main use is as part of a term normalisation process that is usually done when setting up Information Retrieval systems."

The algorithm as the name suggests is a suffix stripper, and is used to combine similar words and terms into one. For English vocabularies this is usually done by cutting the different suffixes. The suffixes present in the English language that do not contribute to the core meaning of a word do not add to its meaning. For instance the basic word CONNECT, is not semantically different from the words CONNECTED, CONNECTING, CONNECTION, or, CONNECTIONS. More generally there are some linguistic aspects of all languages that may be considered when analyzing and parsing text. This can be summarized in some general rules, which were presented in the article by Porter [Por80]. Some examples of these rules are as follows:

Step 1.	SSES $\rightarrow$ SS	caresses $\rightarrow$ caress
	IES $\rightarrow$ I	ponies $\rightarrow$ poni
	SS $\rightarrow$ SS	caress $\rightarrow$ caress
	S $\rightarrow$ " "	cats $\rightarrow$ cat

Step 2.	$(m > 0)$ EED $\rightarrow$ EE	feed $\rightarrow$ feed
	$(*v*)$ ED $\rightarrow$ " "	agreed $\rightarrow$ agree
	$(*v*)$ ING $\rightarrow$ " "	motoring $\rightarrow$ motor

where m is the measure of the word and v denotes a vowel. Stemming is usually used to establish a shorter list of words that each contribute more strongly in information retrieval. This means that our term-document representation becomes more consistent (since similar words are grouped together as the same word) and that the dimensionality of the analyzed data becomes smaller, which is important when dealing with larger scale data. The problem with the approach is that in some cases the stemming algorithm will group words together that are not related, in particular for the case of polysemy.

For implementations of the Porter stemmer in many different progamming languages the reader should consult the reference [Por80] or the code-webpage at URL:

`http://tartarus.org/~martin/PorterStemmer/`

E.2 Text-to-matrix-generator (TMG)

The text-to-matrix-generator toolbox was developed by Zeimpekis and Gallopoulos at the University of Patras. The toolbox contains basic and more advanced tools for performing both text-categorization, classification and similar aspects [DZ11]. More specifically the TMG toolbox implements a text document parser that may be

employed to parse out the relevant document terms that are important for the purposes of this thesis. The TMG gives the user the possibility to perform information retrieval tasks such as indexing, dimensionality reduction, retrieval, clustering, classification, and is freely available at the webpage mentioned in the references [DZ11].
Most importantly for the usage in this thesis work, it supports a well implemented efficient indexing module, which was used to generate the term-document matrix representations from each analyzed dataset. Among other functionalities it has the following:

- removal of stopwords (Used in the thesis), see appendix E.3
- stemming (using Porter's stemming algorithm) (Used in the thesis), see appendix E.1
- removal of short/long terms (Used in the thesis)
- removal of frequent/infrequent terms (locally or globally)
- term weighting and normalization (Used in the thesis), see section 4.2.9
- html filtering, processing of .ps and .pdf files

Thus the framework of the toolbox supports strongly in the preanalysis stage when creating the input, or in other words, the term-document matrix representation of the data. From the possible tools the ones that have been used are marked in the list above. The importance of creating the weighted matrix representation of the data is one of the crucial parts of preparing the datasets, and the TMG has been a good support in this endeavour. Furthermore we remove short and long terms to limit the number of characters in each word to the range 1 to 30 characters. We also remove some highly frequent terms (since these are non-distinguishing for the documents they represent). Furthermore the normalization of the term-document matrix, both locally and globally was used, to achieve a properly weighted term-document matrix.

E.3 Stop-words

Any data cleaning of text documents is not complete without a list of stopwords. The stop-words are a list that represents the words that are to be skipped when parsing the different words contained in the analyzed text. This is done mainly to limit the dictionary used and to remove words that generally give no extra information, such as pronomes or similar. The full list of stop-words used was a combination of different sources for stop-words, combined to give a fairly comprehensive stop-word list. A subset containing the first 4 words beginning with each letter is presented below, for the full list (2377 words) we refer the reader to [Kar].

```
about
above
accordingly
across
 ...
back
be
became
because
 ...
came
can
cannot
cant
 ...
day
described
did
different
 ...
```

```
each
eg
eight
either
 ...
far
few
fifth
first
five
 ...
get
gets
given
gives
 ...
had
hardly
has
have
 ...
ie
if
ignored
immediate
 ...
just
keep
kept
know
last
latter
latterly
least
 ...
made
make
man
many
 ...
name
namely
near
necessary
 ...
of
off
often
oh
 ...
particular
particularly
people
per ...
quite
rather
```

```
really
relatively
respectively
 ...
said
same
second
secondly
 ...
take
taken
than
that
 ...
under
unless
until
unto
 ...
value
various
very
via
 ...
was
way
we
well
 ...
year
years
yet
you
 ...
zero
```

F Mathematics

F.1 Mathematics primer

To give a short introduction to the needed mathematical concepts for understanding the different clustering methods that have been analyzed, we here present a short primer on the background mathematics. The notions below support the description of the clustering models described in section 4.3.

Norm

When speaking of normalizing the vector's we mean to normalize by taking the euclidean norm of the vector. The eucledian norm, also known as the 2-norm, of a vector v is defined as: $\sqrt{v_1^2 + v_2^2 + v_3^2 + \cdots + v_n^2}$ and is usually denoted as $||v||$.

Range

The range of a collection is the subspace that is spanned by the vectors in a given matrix A. So if the matrix A is an $m \times n$ matrix, the column vectors $a_1, \ldots, a_n$ span the *range* of A. The range is denoted by $R(A) \subset \mathcal{R}^m$.

F.1.1 Eigenvalues and eigenvectors

Let A be a matrix, λ be a scalar and x be a non-zero vector such that the following holds:

$$Ax = \lambda x$$

then the vector x is an eigenvector and λ is the corresponding eigenvalue of the matrix A. The collection of eigenvalues of A is known as the spectrum of A.

F.1.2 Singular value decomposition

We follow the description in Golub and van Loan [GHG96] to describe the SVD. Let $A \in \mathcal{R}^{m \times n}$ then there are matrices: $U = [u_1, u_2, \dots, u_p] \in \mathcal{R}^{m \times p}$ and $V = [v_1, v_2, \dots, v_p] \in \mathcal{R}^{n \times p}$ each orthogonal columns such that:

$$A = U \Sigma V^T \text{ or } U^T A V = \Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_p \end{bmatrix} \tag{F.1.1}$$

with $p = min(m, n)$ and where the singular values σ of the matrix A, appear in decreasing order in the diagonal matrix Σ: $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$ for details we refer the reader to Lay [Lay03]. Each of the σ_i are a singular value of the matrix A. and the vectors u_i and v_i are the left and right singular vectors of A. The singular values are the square roots of the squared eigenvalues of $A^T A$, ordered in decreasing order of magnitude. The columns of U and V are the orthonormal eigenvectors of AA^T and $A^T A$ respectively.

Using the SVD we can get approximations to the full matrix A. When constructing an approximation to the matrix A we use only the k largest singular values to reconstruct the matrix A from U, V and Σ. This is called a k-rank approximation A_k to the matrix A and is defined as $A_k = U \Sigma_k V^T$ where Σ_k is:

$$\Sigma_k = \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_k & \\ & & & 0 \end{bmatrix} \text{ and } A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T \tag{F.1.2}$$

in other words the k-rank approximation to the matrix A only uses the k largest singular values when we compute the approximation. For a graphical example of the process see figure 4.3.5. It can be shown that this approximation is the best possible approximation to the matrix A in 2-norm, i.e.:

$$\min_{rank(B)=k} ||A - B||_2 = ||A - A_k||_2 = \sigma_{k+1} \tag{F.1.3}$$

for a proof we refer the reader to Golub [GHG96] or Lay [Lay03].

F.2 Learning algorithms

There exists a large number of numerical methods for updating and correcting the neural network weights. This can be thought of as an iterative process where the goal is to minimize the error of the output from the NN. To minimize this error we compute the partial derivative for each weight on the output error, and thus correct the weights between the input layer and the hidden layer in the ANN.
This minimization and correction of the error is done by updating the weight vector, and by updating it correcting the error to minimize it. Since the error function is the squared errors we have a convex function and thus there must exist a global minimum when examining the function.
The aim of the error minimization in the back-propagation network is not to reach the global optimum (this can almost never be guaranteed since we do not know the exact details of the error function). The goal of the minimization is to reach the lowest possible value for a given number of iterations, or to reach some given value for MSE, described in equation 5.3.6.

F.2.1 Newton method

The Newton (or Newton-Rhapson) method is a method for finding the root to an equation. It makes use of the first and second derivative of the Taylor expansion of a function around a point that is assumed to be close to a root. The Taylor series expansion of a function f around a point $x = x_0 + \epsilon$ is:

$$f(x_0 + \epsilon) = f(x_0) + f'(x_0)\epsilon + \frac{1}{2} f''(x_0)\epsilon^2 + \cdots \Rightarrow f(x_0 + \epsilon) \approx f(x_0) + f'(x_0)\epsilon \tag{F.2.1}$$

where the approximation is the linear expansion representing the tangent of the function at the point x_0. This can then be transformed into the following statement, for describing the error by setting $f(x_0 + \epsilon) = 0$:

$$\epsilon_0 = -\frac{f(x_0)}{f'(x_0)} \tag{F.2.2}$$

which then describes the first adjustment towards the root of the equation. If we let $x_1 = x_0 + \epsilon_0$ we obtain the new x from which the process is repeated *iteratively*:

$$\epsilon_n = -\frac{f(x_n)}{f'(x_n)} \tag{F.2.3}$$

This can then be applied iteratively to reach a local minimum x_n by repeating the steps:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{F.2.4}$$

which finally (given that the function is sufficiently well behaved) will result in a fixed point ($f(x_n) = 0$). The method is fairly simple and is strongly dependent on the choice of initial x_0. If the function is close to a horizontal asymptote or a local extremum it may diverge at these points and have slow convergence toward the true optimum. Thus the Newton method is usually not employed for finding the optimum value of a function, even though it can be shown that for some well-behaved functions it will converge to the global optimum [Weib].

F.2.2 Levenberg-Marquardt method

There are many alternatives to the Newton method that may be employed for finding the optimum value of a function $F(x)$ that is a sum of squares of nonlinear functions:

$$F(x) = \frac{1}{2} \sum_{i=1}^{m} f_i(x)^2 \tag{F.2.5}$$

if we denote the Jacobian of $f_i(x)$ as $\mathbf{J}_i(x)$ then the Levenberg-Marquardt method searches in a direction $\vec{p}$ that is the solution to the system of equations:

$$\left(\mathbf{J}_k^T \mathbf{J}_k + \lambda_k \mathbf{I}\right) p_k = -\mathbf{J}_k^T f_k \tag{F.2.6}$$

where $\lambda_k \in \mathcal{R}^+$ and $\mathbf{I}$ is the identity matrix. A good property of the method is that for some scalar $\delta \sim \lambda_k$, p_k is the solution to the minimization problem $||\mathbf{J}_k p + f_k||_2^2/2 \quad s.t. : ||p||_2 \leq \delta$ [Weia].

F.2.3 Scaled conjugate gradient (SCG)

The scaled conjugate gradient method is one of the many *conjugate gradient methods* (CGM). These are used in our BPNN to learn the network, regarding what direction to optimize in. The principle underlying the SCG is to use the main idea of CGM, which is to use conjugate directions to optimize the performance of a network. Given a weight optimization problem this boils down to taking steps (in the two dimensional case) perpendicular to the previous step. This means that for each step in the optimization we take a step in the weight space, that is conjugate to the previous, towards the optimum (global or local) weight. This involves computing the optimal step length, which also requires computing all network responses to be inputs and computed weights. The SCG was developed to avoid the computationally heavy step of evaluating the network for each line-search.

The general idea of the SCG is to combine Levenberg-Marquardt algorithm with a CGM. As is the case with many other quasi-Newton optimization methods, the SCG was created to approach second-order speed without needing the computation of the Hessian matrix, see [Mol]. As with most optimization algorithms the problems of ending up in a local minimum are present, however this aspect was not elaborated on further in this thesis.

To describe the implementation of the SCG for a BPNN we describe the following. Let x be the input dataset, with dimensionality D. Let $\tilde{x}$ denote the matrix x together with a column of ones, representing the bias. The number of samples is N. Let z be the outputs from the hidden layer and let $\tilde{z}$ denote the matrix z together with a column of ones, representing the bias. Let w_{in} be the hidden units weight set. Let $h(p)$ denote the (possibly non-linear) transformation function in the hidden layer, where p is a linear combination of the weighted inputs xw_{in}. Let α_k be the step size to be taken in the correction direction in each step. Finally let the system have resulting variables $y = [y_1, y_2, \ldots, y_k]$ that also are a weighted linear combination of the outputs from the hidden layer $zw_{out} = y_k$. Thus the weights between the hidden and final layer are w_{out} final output has K different nodes. The targets for the values are denoted as t Then for a backward pass the weights are adjusted according to:

$$w_{in} \leftarrow w_{in} - \alpha_k \frac{1}{N}\frac{1}{K}\tilde{x}^T((y-t)\tilde{w}_{out}^T(1-z^2)) \tag{F.2.7}$$

$$w_{out} \leftarrow w_{out} - \alpha_k \frac{1}{N}\frac{1}{K}\tilde{z}^T(y-t) \tag{F.2.8}$$

Now let $E(w)$ be the error function which should be minimized, and where $w = [w_{in}, w_{out}]$. The SCG is based on the following four steps:

Step 1. Set the start $w_0 = [w_{in,0}, w_{out,0}]$ and the gradient of $E(w)$ to initialize:

$$\begin{aligned} g_0 &= -E(w_0) \\ d_0 &= g_0 \text{ (first search direction)} \\ k &= 0 \end{aligned}$$

Step 2. Evaluate step size α_k and update w_{k+1}

$$\alpha_k = \operatorname{argmin}_\alpha E(w_k + \alpha d_k)$$

Step 3. Calculate the gradient of the new error

$$g_{k+1} = -E'(w_{k+1})$$

Step 4. Find the new conjugate direction d_{k+1} to the former d_k

```
if $k \mod D == 0$
    $d_{k+1} = g_{k+1}$
else
    $\beta_k = \frac{|g_{k+1}|^2 - g_{k+1}^T g_k}{|g_k|^2}$
    $d_{k+1} = g_{k+1} + \beta_k d_k$
end
```

References

[AF10] H. Arman and J. Foden. "Combining methods in the technology intelligence process: application in an aerospace manufacturing firm". In: *R&D Management* 40.2 (2010), pp. 181–194. ISSN: 1467-9310. DOI: 10.1111/j.1467-9310.2010.00599.x. URL: http://dx.doi.org/10.1111/j.1467-9310.2010.00599.x.

[AJCT05] A. C. L. W. A. J. C. Trappey S. C. I. Lin. "Using Neural Network Categorization Method to Develop an Innovative Knowledge Management Technology for Patent Document Classification". In: *The 9th International Conference on Computer Supported Cooperative Work in Design.* 2005, pp. 830–835.

[AJCT06] C. V. T. C.-I. L. A. J. C. Trappey F.-C. Hsu. "Development of a patent document classification and search platform using a back-propagation network". In: *Expert Systems with Applications* 31 (2006), pp. 755–765.

[Bek03] R. Bekkerman. "Distributional Clustering of Words for Text Categorization". Research Thesis. Technion - Israel Institute of Technology, 2003.

[Blo04] K. Blom. "Information Retrieval Using Krylov Subspace Methods". PhD thesis. Chalmers University of Technology and University of Gothenburg, 2004. URL: http://publications.lib.chalmers.se/cpl/record/index.xsql?pubid=1899.

[BM98] L. D. Baker and A. K. McCallum. "Distributional clustering of words for text classification." In: *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval.* Melbourne, Australia, 1998, pp. 96–103.

[BY09] D.-h. Z. Bo Yu. "Combining neural networks and semantic feature space for email classification". In: *Knowledge-Based Systems* 22 (2009), pp. 376–381.

[CDM08] H. S. Christopher D. Manning Prabhakar Raghavan. *Introduction to information Retrieval.* Ed. by C. U. Press. Cambridge University Press, 2008.

[Chh06] T. S. Chhabra. *Back-propagation Neural Net - CodeProject.* English. The Code Project. 2006. URL: http://www.codeproject.com/KB/recipes/BP.aspx.

[CHP00] H. T. S. V. C. H. Papadimitrou P. Raghavan. "Latent Semantic Indexing: A Probabilistic Analysis". In: *JCSS* 61 (2000), pp. 217–235.

[Dee88] e. a. Deerwester S. "Improving Information Retrieval with Latent Semantic Indexing". In: *Proceedings of the 51st Annual Meeting of the American Society for Information Science 25.* 1988.

[DPM88] A. N. N. Don P. Mitchell. "Reconstruction filters in computer-graphics". In: *ACM SIGGRAPH International Conference on Computer Graphics and Interactive Techniques* 22 (1988), pp. 221–228.

[Dum91] S. T. Dumais. "Improving the retrieval of information from external sources". In: *Behavior Research Methods, Instruments & Computers* 23 (1991), pp. 229–236.

[DZ11] E. G. Dimitrios Zeimpekis. *TMG - Text-to-matrix generator.* English. 2011. URL: http://scgroup20.ceid.upatras.gr:8000/tmg/.

[Ern03] H. Ernst. "Patent information for strategic technology management". In: *World Patent Information* 25 (2003), pp. 233–242.

[Gap] *Gapminder: Unveiling the beauty of statistics for a fact based world view. - Gapminder.org.* English. Gapminder Foundation. 2011. URL: http://www.gapminder.org/.

[GHG96] C. F. v. L. G. H. Golub. *Matrix Computations.* Ed. by J. H. U. Press. 3rd. Johns Hopkins University Press, 1996.

[GS75] C. S. Y. G. Salton A. Wong. "A Vector Space Model for Automatic Indexing". In: *Information Retrieval and Language Processing* 18 (1975), pp. 613–620.

[HTN97] K. L. L. H. T. Ng W. B. Goh. "Feature selection, perceptron learning, and a useability case study for text categorization". In: *Proceedings of SIGIR-97, 20th ACM International Conference on Research and Development in Information Retrieval.* 1997, pp. 67–73.

[ID97] D. R. I. Dagan Y. Karov. "Mistake-driven learning in text categorization". In: *Proceedings of EMNLP-97, 2nd Conference on Empirical Methods in Natural Language Processing.* 1997.

[JADP10] M. P. C.-R. Jose Aldo Diaz-Prado Arturo Lopez-Pineda. "Corporate Technology Intelligence Research System through Recycling Public Patent Databases". In: *Communications of the IBIMA* 2010 (2010), p. 10. URL: http://www.ibimapublishing.com/journals/CIBIMA/2010/592641/592641.pdf.

[Joa01] T. Joachims. “A statistical learning model of text classification with support vector machines”. In: *Proceedings of SIGIR-01, 24th ACM International Conference on Research and Development in Information Retrieval.* Ed. by D. H. K. J. Z. W. B. Croft D. J. Harper. New Orleans, US, 2001, pp. 128–136.

[Jor03] B. D. M. N. A. Y. Jordan M. “Latent Dirichlet Allocation”. In: *Journal of Machine Learning Research* 3 (2003), pp. 993–1022.

[Kan05] M. Kanellos. *FAQ: Forty years of Moores Law.* English. CBS Interactive. 2005. URL: `http://news.cnet.com/FAQ-Forty-years-of-Moores-Law/2100-1006_3-5647824.html?tag=nefd.lede`.

[Kar] R. Karmakar. *Combined stop-word list.* English. URL: `http://dl.dropbox.com/u/27897997/common_words_edit_2`.

[Lan09a] Landon IP Inc. *ECLA classification system.* English. Landon IP. 2009. URL: `http://www.intellogist.com/wiki/ECLA_Classification_System`.

[Lan09b] Landon IP Inc. *INPADOC - Intellogist.* English. Landon IP, Inc. 2009. URL: `http://www.intellogist.com/wiki/INPADOC`.

[Lan09c] Landon IP Inc. *Japanese F-Index and F-Terms.* English. Landon IP, Inc. 2009. URL: `http://www.intellogist.com/wiki/Japanese_F-Index_and_F-Terms`.

[Lan09d] Landon IP Inc. *Report:Derwent World Patents Index/Special Indexing/Derwent Manual Codes.* English. Landon IP, Inc. 2009. URL: `http://www.intellogist.com/wiki/Report:Derwent_World_Patents_Index/Special_Indexing/Derwent_Manual_Codes`.

[Lan09e] Landon IP Inc. *US Patent Classification System.* English. Landon IP, Inc. 2009. URL: `http://www.intellogist.com/wiki/US_Patent_Classification_System`.

[Lan94] V. Lange. *Technologische Konkurrenzanalyse: Zur Früherkennung von Wettbewerberinnovationen bei Deutschen Grossunternehmen.* Ed. by D. Universitätsverlag. Wiesbaden, 1994.

[Lay03] D. C. Lay. *Linear Algebra and Its Applications.* Ed. by U. of Maryland College Park. Addison-Wesley, 2003. URL: `http://www.laylinalgebra.com/free_site/index.html`.

[Lic03] E. Lichtenthaler. “Third generation management of technology intelligence processes”. In: *R&D Management* 33.4 (2003), pp. 361–375. ISSN: 1467-9310. DOI: `10.1111/1467-9310.00304`. URL: `http://dx.doi.org/10.1111/1467-9310.00304`.

[Lic07] E. Lichtenthaler. “Managing technology intelligence processes in situations of radical technological change”. In: *Technological Forecasting and Social Change* 74.8 (2007), pp. 1109 –1136. ISSN: 0040-1625. DOI: `DOI:10.1016/j.techfore.2006.10.001`. URL: `http://www.sciencedirect.com/science/article/B6V71-4MFKD0W-2/2/01376bf844f3acd58743bf13a7181c2e`.

[Mic11] Microsoft Corp. *Feature List - Microsoft Research.* English. Microsoft Corp. 2011. URL: `http://research.microsoft.com/en-us/projects/mslr/feature.aspx`.

[ML05] H.-B. L. S.-Y. S. Man Lan Chew-Lim Tan. “A comprehensive comparative study on term weighting schemes for text categorization with support vector machines”. In: *WWW-05: Special interest tracks and posters of the 14th international conference on World Wide Web.* New York, NY, USA: ACM, 2005, pp. 1032–1033.

[Mol] M. F. Moller. *A scaled conjugate gradient algorithm for fast supervised learning.* English. Available electronically. URL: `ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia353_1s07/papers/moller_90.pdf`.

[Net] *Netlib.* English. University of Tennessee Knoxville and Oak Ridge National Laboratory. URL: `http://www.netlib.org/`.

[Non91] I. Nonaka. “The knowledge-creating company.” In: *Harvard Business Review* 69 (1991), pp. 96–104.

[Non94] I. Nonaka. “A dynamic theory of organizational knowledge creation.” In: *Organization Science* 5 (1994), pp. 14–37.

[PNT05] V. K. P-N. Tan M. Steinbeck. *Introduction to Data Mining.* Ed. by A. Wesley. Addison Wesley; US ed edition, 2005.

[Por05] C. S. Porter A. *Tech Mining.* Ed. by W. Inter-Science. John Wiley & Sons, Inc., Hoboken, New Jersey, 2005.

[Por80] M. Porter. “An algorithm for suffix stripping”. In: *Program* 14.3 (1980). Matlab implementation available at URL., pp. 130–137. URL: `http://tartarus.org/~martin/PorterStemmer/matlab.txt`.

[Roj] R. Rojas. *Neural Networks - A Systematic Introduction.* Ed. by Springer-Verlag. Springer-Verlag, p. 502. URL: `http://page.mi.fu-berlin.de/rojas/neural/index.html.html`.

[Ros57] F. Rosenblatt. *The Perceptron - a perceiving and recognizing automaton.* Tech. rep. 85-460-1. Cornell Aeronautical Laboratory, 1957.

[RZ09] S. Robertson and H. Zaragoza. "The Probabilistic Relevance Framework: BM25 and Beyond". In: *Foundations and Trends in Information Retrieval* 3 (2009), pp. 333–389. DOI: 10.1561/1500000019.

[Sar02] W. S. Sarle. *Neural Network FAQ, part 1 of 7: Introduction.* 2002. URL: ftp://ftp.sas.com/pub/neural/FAQ.html.

[SB88] G. Salton and C. Buckley. "Term-weighting approaches in automatic text retrieval". In: *Information Processing & Management* 24.5 (1988), pp. 513 –523. ISSN: 0306-4573. DOI: DOI:10.1016/0306-4573(88)90021-0. URL: http://www.sciencedirect.com/science/article/B6VC8-469WV05-1/2/d251fa7251ec6c4247f833f88efd3068.

[SC98] R. A. P. R. S. Chakrabati B. Dom. "Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies". In: *The VLDB Journal* 7 (1998), pp. 163–178.

[Seb02] F. Sebastiani. "Machine Learning in Automated Text Categorization". In: *ACM Computing Survey* 34 (2002), pp. 1–47.

[SER76] K. S. J. S. E. Robertson. "Relevance weighting of search terms". In: *J. Amer. Soc. Inform. Sci.* 27 (1976), pp. 143–160.

[SKF] SKF. "SKF Internal Sources". SKF internally available information, not available publicly. URL: www.skf.com.

[STD90] G. W. F. T. K. L.-R. H. Susan T. Dumais Scott Deerwester. "Indexing by Latent Semantic Analysis". In: *Journal of the American Society for Information Science* 41 (1990), pp. 391–407.

[The11a] The Institution of Engineering and Technology. *Inspec subject coverage.* The Institution of Engineering and Technology. 2011. URL: http://www.theiet.org/publishing/inspec/about/coverage/.

[The11b] The MathWorks Inc. *MATLAB - The Language Of Technical Computing.* English. The MathWorks Inc. 2011. URL: http://www.mathworks.com/products/matlab/.

[Tho11a] Thomson Reuters. *Intellectual Property Research and Analysis.* English. Thomson Reuters. 2011. URL: http://www.thomsoninnovation.com/.

[Tho11b] Thomson Reuters. *THOMSON DATA ANALYZER - Thomson Innovation.* English. THOMSON REUTERS. 2011. URL: http://thomsonreuters.com/content/legal/products/Thomson_Data_Analyzer.

[TLG04] M. S. Thomas L. Griffiths. "Finding Scientific Topics". In: *PNAS* 101 (2004), pp. 5228–5235.

[ULS07] U. o. P. University Library System. *Compendex Database Description.* English. University of Pittsburg. 2007. URL: http://www.library.pitt.edu/articles/database_info/ei_comp.html.

[Vap95] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Ed. by Springer-Verlag. Springer-Verlag, 1995.

[Weia] E. W. Weisstein. *Levenberg-Marquardt Method.* English. MathWorld–A Wolfram Web Resource. URL: http://mathworld.wolfram.com/Levenberg-MarquardtMethod.html.

[Weib] E. W. Weisstein. *Newton's Method.* English. MathWorld–A Wolfram Web Resource. URL: http://mathworld.wolfram.com/NewtonsMethod.html.

[WIP09] WIPO. *INTERNATIONAL PATENT CLASSIFICATION, (Version 2009), GUIDE.* World Intellectual Property Organization. 2009. URL: http://www.wipo.int/export/sites/www/classifications/ipc/en/guide/guide_ipc_2009.pdf.

[WWC99] Y. S. W. W. Cohen. "Context-sensitive learning methods for text categorization". In: *ACM Trans. Inform. Syst.* 17 (1999), pp. 141–173.

[Yip] Yippy Inc. *Yippy.* Yippy Inc. URL: http://search.yippy.com/.

[YY94] C. G. C. Y. Yang. "An example-based mapping method for text categorization and retrieval". In: *ACM Trans. Inform. Syst.* 12 (1994), pp. 252–277.

[YY99] X. L. Y. Yang. "A re-examination of text categorization methods". In: *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval.* 1999, pp. 42–49.

[Zip35] G. K. Zipf. *The Psychobiology of Language.* Ed. by Houghton-Mifflin. Houghton-Mifflin, 1935.