DEPARTMENT OF INFORMATION AND ELECTRONIC ENGINEERING

MASTER OF SCIENCE

ON

WEB INTELLIGENCE

Development of automatic patent classification techniques

MASTER'S THESIS

by

ELENI KAMATERI

Supervisor: Michail Salampasis
Professor

Thessaloniki, July 2021

This page is intentionally left blank.

INTERNATIONAL HELLENIC UNIVERSITY

DEPARTMENT OF INFORMATION AND ELECTRONIC ENGINEERING

MASTER'S PROGRAM ON WEB INTELLIGENCE

Development of automatic patent classification techniques

MASTER'S THESIS

by

ELENI KAMATERI

Supervisor: Michail Salampasis
Professor

Approved by the evaluation committee on Choose a date.

(Signature)	*(Signature)*	*(Signature)*
.....	.....	.....
Name Surname Choose an item. I.H.U.	Name Surname Choose an item. I.H.U.	Name Surname Choose an item. I.H.U.

Thessaloniki, July 2021

(Signature)

.....

Click here to enter text.

Click here to enter text.

Acknowledgement

First and foremost, I would like to thank my thesis supervisor Prof. Michail Salampasis for giving me the opportunity to work on the interesting area of Information Retrieval and, specifically, investigating the research field of automatic patent classification. Also, I would like to thank him for his great support, guidance and advices throughout this thesis work. Next, I would like to express special appreciation and thanks to PhD student at International Hellenic University Vasileios Stamatis for guiding and helping me throughout this thesis work; our constructive discussions were absolutely invaluable and helped me going deeper in my research. Also, I would like to thank Prof. Konstantinos Diamantaras for his guidance and help with ML related problems that I encountered through my thesis work. Finally, I would like to acknowledge the unconditional love, patience and support provided by my husband Vasili, my family and my husband's family during this demanding period.

This page is left intentionally blank.

Intellectual property is the oil of the 21st century.

(All today's richest men have made their money out of intellectual property)

-Mark Getty

This page is left intentionally blank.

Abstract

Many thousands of patent applications arrive at patent offices around the world every day. One important task when a patent application is submitted is to assign one or more classification codes from the complex and hierarchical patent classification schemes that will enable routing of the patent application to a patent expert who is knowledgeable about the specific technical field. This task is typically undertaken by patent professionals, however due to the large number of applications and the potential complexity of an invention, they are usually overwhelmed. Therefore, there is a need for this code assignment manual task to be supported or even fully automated by classification systems that will classify patent applications, hopefully with an accuracy close to patent professionals. Like in many other text analysis problems, in the last years, this intellectually demanding task has been studied using word embeddings and deep learning techniques. In this thesis we present the results of different word embeddings and deep learning techniques, considering different parts of the patent document on automatic patent classification in the level of sub-classes. Compared with previous works we focus on single-label classification exploiting the <main classification> labels found on patents in the CLEF-IP 2011 collection.

This report is the result of a master thesis in information retrieval field at International Hellenic University during spring term of 2021.

Keywords: Patent, Classification, Single-label, Sub-classes, Deep learning, Word embeddings

This page is left intentionally blank.

Table of Contents

1

Introduction

1.1 Innovation and patents

Patents are considered as significant intellectual resources. They contain valuable technical knowledge that can help engineers to understand detailed concepts and underlying component technology for the design and development of the described innovation [TLL07]. Moreover, patents can prevent commercial infringements within the specified time and the prescribed area, such as selling, importing, distributing, etc. Hence, companies and individuals usually use patents as an effective way to protect their intellectual property (IP) and new products from competitors.

Patents are a measure of technological innovation and development for a country, while they can also promote socio-economic and technological progress [TTW+12]. The effective analysis of patents is beneficial to the industrial, commercial and legal community as they can reveal important technological and business trends, propose novel industrial solutions and make crucial investment decisions [ADK15] [PYP17].

Patents have also a great research value. Since the state-of-the-art technical knowledge can often be found in related patent documents, taking full advantage of patent information allows one to track the progress and frontiers of related technologies, which may help to avoid reinventing the wheel and can thus save research cost and shorten research time [LHC+18].

1.2 The Global Patent System: how it works

Patents have increasingly become an important economic asset and patent filing rates have enormously increased in recent years. In 2019, innovators filled about 3.2 million patent applications worldwide, while the long-term trend shows a constant growing in the patent applications worldwide every year since 1995 with few exceptions (2002, 2009 and 2019) [Wip21]. This long-term trend is depicted in Figure 1.

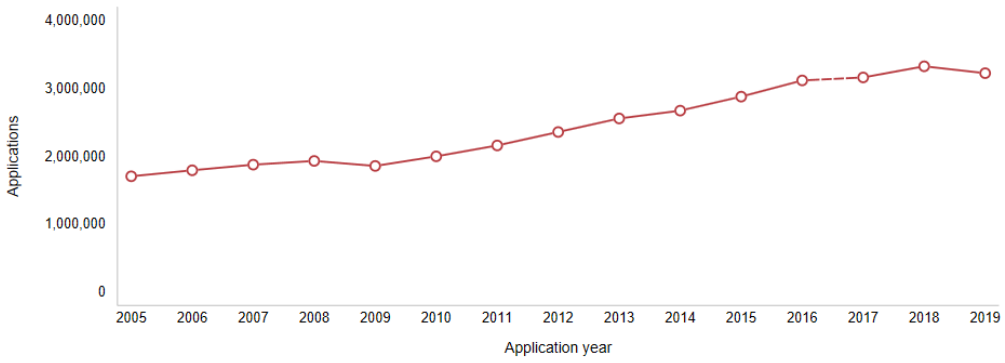


Figure 1: Patent applications worldwide for the period of 2005-2019 (Source: [Wip21]).

Hence, many thousands of patent applications arrive at patent offices around the world every day. Consequently, patent offices internationally have to deal with these large number of applications. Therefore, automating any subtask of the large patent examination process is an important problem which has significant impact because it can speed up the patent examination process.

One important subtask of the patent examination process that can be automated is the pre-classification task which consists the assignment of one or more classification codes from the complex and hierarchical patent classification schemes (e.g., IPC, CPC) when a patent application is submitted. This task is typically done by front-line patent examiners who manually classify an application using a classification scheme.

This initial assignment of code(s) is quite important considering that it will enable routing of the patent application to a patent expert or sub-department in a patent office who is knowledgeable about the specific technical field for detailed examination of the patent application's novelty. Moreover, during the examination process, properly assigning other relevant classification codes ensures that patents with similar technical features will be grouped under the same intellectual scheme, something which is crucially important for many subsequent patent management, search, retrieval or any further patent analysis tasks. For example, in prior art search, the assigned codes could substantially improve the search for relevant patents in different ways such as filtering the search or expanding with extra search terms.

This task is typically undertaken by patent professionals, however due to the large number of applications and the potential complexity of an invention, they are usually overwhelmed. Therefore, there is a need for this code assignment manual task to be supported or even fully automated by classification systems that will classify patent applications, hopefully with an accuracy close to patent professionals.

1.3 The problem of patent classification

At present, the classification of newly applied patent documents is mainly based on manual work. This manual approach leads to a lot of time-consuming work required by the patent examiners earlier in the pre-classification phase and the patent experts in related fields during the detailed examination to complete the classification and may contribute to human errors.

Furthermore, manual patent classification task faces several inherent challenges. Firstly, the classification scheme has a complicated hierarchical structure where each patent must finally be assigned one or more sub-group level labels among thousands of sub-group level classification codes available. Secondly, the distribution of patents among categories is highly unbalanced with about 80% of all the documents classified in about 20% of the categories [LHC+18]. Furthermore, highly structured patent documents are often lengthy and full of technical and legal terminologies, which increases the challenge to efficiently analyze them even for domain experts.

With the recent advancements of information and computer technology (ICT), automatic classification of texts can be used as an aid for the classification of patent texts as well. Automatic or semi-automatic assistance with classification can reduce the workload of the patent examiner and experts and improve the efficiency of classification. At the same time, automatic patent classification can also aid with inherent challenges of the manual patent classification task reducing the uncertainty and errors related to manual classification.

However, based on the current literature review, relevant research is still in the experimental stage. Although several works tried to automate the patent classification task, these do not attain acceptable performance (ideally close to human performance). Therefore, the automation only worked at a higher level of the hierarchy and its large-scale usage has not been applied so far.

1.4 Thesis subject

Two important features for automated patent classification are derived: a) it can be done at different levels of the classification hierarchy and b) it can be done either as a single-label task in case of the pre-classification stage (described above) or as a multi-label task because each patent can have multiple codes assigned, sometimes at a different level of the hierarchy. This

thesis focuses on single-label classification in the main sub-class category of the patent. To evaluate our methods for single-label classification, we use the <main classification> tag that exists in patents in the CLEF-IP 2011 dataset to extract the primary classification. This label denotes the main and most important classification code assigned to a patent.

More specifically, the subject of this thesis is to use different data mining methods to automatically classify patent documents in the level of sub-classes. Data mining methods mainly includes three parts, the first is the data processing, the second is the feature engineering (feature selection and feature representation), and the third is the application of machine learning techniques. In order to cover all these parts, this thesis introduces an end-to-end patent classification pipeline that gets the patent text data, cleans it from irrelevant information, finds the feature words of the text in a patent document, expresses the text as a feature vector, and then classifies it. This patent classification pipeline employs different language models and different machine learning algorithms, considering different parts of the patent document for the patent representation.

Recently, automated patent classification has been examined using deep learning (DL) methods such as convolutional neural networks (CNN) [LHC+18, ZHF+20, AKG+19], Word2Vec and Long-short term memory (LSTM) [GMB17, XWZ18, Sof19] and other DL methods [RK19] to predict the most representative classification code(s) for a patent. In the current study, CNN and RNN models are evaluated since they are experiencing good results in many fields including automated patent classification. Additionally, we enforce an ensemble method to combine the performance of different classifiers and achieve better classification results.

Furthermore, some researchers mentioned that the usage of the title and the abstract of a patent can improve the performance of patent classification [LHC+18, ZHF+20], while in other research works, claims part was used as input data for patent classification [LH0]. Besides, some researchers mentioned that the description part usually shows detailed information about one patent which might be useful in patent classification [LK16, STS17]. In this thesis, we feed our models with input data from different patent parts (title-abstract, description, and claims) used separately or be combined.

1.4.1 Contribution

The thesis contribution is summarized as follows:

1. We performed single-label automatic patent classification down to the sub-class level.
2. We used different language models and different deep learning algorithms to build the patent classifiers and we also compared these performances.
3. We used different parts of the patent as input data in the patent classifiers and we also compared these performances.

4. We set up a patent classification pipeline deploying the different techniques of step 3 and 4.

1.5 Chapter organization

The remainder of this thesis structured as follows: Chapter 2 provides the theoretical background of this study. Initially, this chapter gives a brief explanation of patent documents and the classification schemes followed. Afterwards, it outlines the elements of data mining process. Chapter 3 describes related work in the area of automatic patent classification. Chapter 4 briefly discusses about multiple methods used in this study. Chapter 5 presents the evaluation methodology, including the evaluation criteria, the evaluation system, the data collection and the experimental process. Chapter 6 describes the technical details for the implementations. Chapter 7 presents and discusses the results of all the experiments. Finally, Chapter 8 sums up the work and its achievements and provides suggestions about potential future work with unrevealed topics and aspects that require further investigation.

2

Theoretical background

This chapter introduces some theoretical background necessary to understand the principal ideas of automated patent classification. More specifically, the chapter describes the patent documents and the document/text data mining methods.

2.1 Patents

2.1.1 CLEF-IP campaign

The CLEF-IP campaign[1] was initiated in 2009 with a twofold purpose: i) to encourage and facilitate research in the area of patent retrieval; and ii) to create a large clean test collection of patents in the three main European languages for experimentation. In 2010, the CLEF-IP benchmarking activity included a patent classification task, while in 2011 two patent image related tasks were also offered [PLH+11].

The CLEF-IP dataset includes patent documents published by the European Patent Office (EPO) which contain a mixture of English, German and French content.

2.1.2 The patent documents

A patent document describes a patent application. It contains scientific, technical and administrative information about the patent. The patent application is structured by the following fields:

- Basic information
- Bibliographic data
- Copyright

[1] http://www.clef- campaign.org

- Description
- Claims
- Legal status
- Abstract

Bibliographic data is further broken down into the following parts:

- Technical data, which includes the classification codes and the invention-title, citations and figures
- Parties, which includes the applicants, the inventors, the agents and the assignees
- International convention data
- Publication reference
- Application reference
- Priority claims
- Dates of public availability
- Office specific data

When a document is approved and published to the public, it has already been assigned a patent classification code or more codes. This code or these codes will make the patent easily searchable in a specific category.

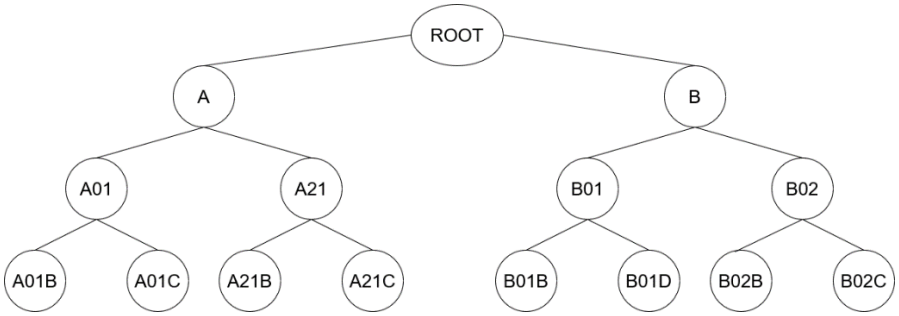
2.1.3 Classification standards

There are different classification standards adopted by different patent authorities [Wol12]. Currently, the most widely used classification scheme to organize patents is the International Patent Classification (IPC)[2]. Other classification structures, such as the European CLAssification (ECLA) [Dic94], the Japanese File Index and F-terms [Sch02] and the new Cooperative Patent Classification (CPC)[3], were designed taking the IPC as a basis.

Classification standards follow a hierarchical structure meaning that each inner node in the hierarchy has exactly one parent and the path to each code is unique. The IPC follows this hierarchical structure containing thousands of codes each representing a more general (at higher levels) or very specific technological concepts. In the version of IPC 2006 for example, the classification scheme contains in total 8 sections, 131 classes, 642 sub-classes, 7,537 groups and 69,487 subgroups. Furthermore, the classification schemes are periodically updated meaning that there is a substantial need to support the classification task of patents. Figure 2 shows this hierarchical structure of IPC scheme.

[2] https://www.wipo.int/classifications/ipc/en/

[3] https://www.cooperativepatentclassification.org/index



Level	IPC code	Description
Section	A	Human necessities
Class	A01	Agriculture; forestry; animal husbandry; hunting; trapping; fishing
Sub-class	A01B	Soil working in agriculture or forestry; parts, details, or accessories of agricultural machines or implements, in general

Figure 2: Hierarchical structure of the International Patent Classification (IPC) scheme. The choice of IPC codes illustrated in the figure was made randomly.

In 2013, the Cooperative Patent Classification (CPC) was introduced by the European Patent Office (EPO)[4] and the United States Patent and Trademark Office (USPTO)[5] as a common classification scheme. CPC consists an extension of the IPC and contains approximately 250,000 individual codes organized in a similar hierarchical structure. Since then a growing number of national patent offices have decided to follow the CPC. Therefore, it is foreseeable that the CPC system will eventually replace the IPC system as the new standard.

However, most of the papers in the field were based on the IPC classification scheme, because this is the classification scheme followed by the patent documents offered in the CLEF-IP 2011 test collection [PLH+11].

2.2 Data mining

Data mining is the process of revealing unknown insights and valuable knowledge from an enormous amount of data. When the object of data mining is a text or a document, as happens in our case, this process is called text/document data mining [AZ12]. Data mining usually consists of three steps:

- data processing
- feature engineering
- classification

[4] https://www.epo.org/index.html

[5] https://www.uspto.gov/

Data processing includes the cleaning from irrelevant information and the preparation of the training data. The feature engineering expresses data to vector and selects feature words. The classification uses machine learning techniques to build a training model. Then this training model is used to obtain the corresponding category of the testing data, which have passed the same processing and feature engineering with the training data. In order to make the steps of data mining clearer, Figure 3 presents the procedure of the text data mining.

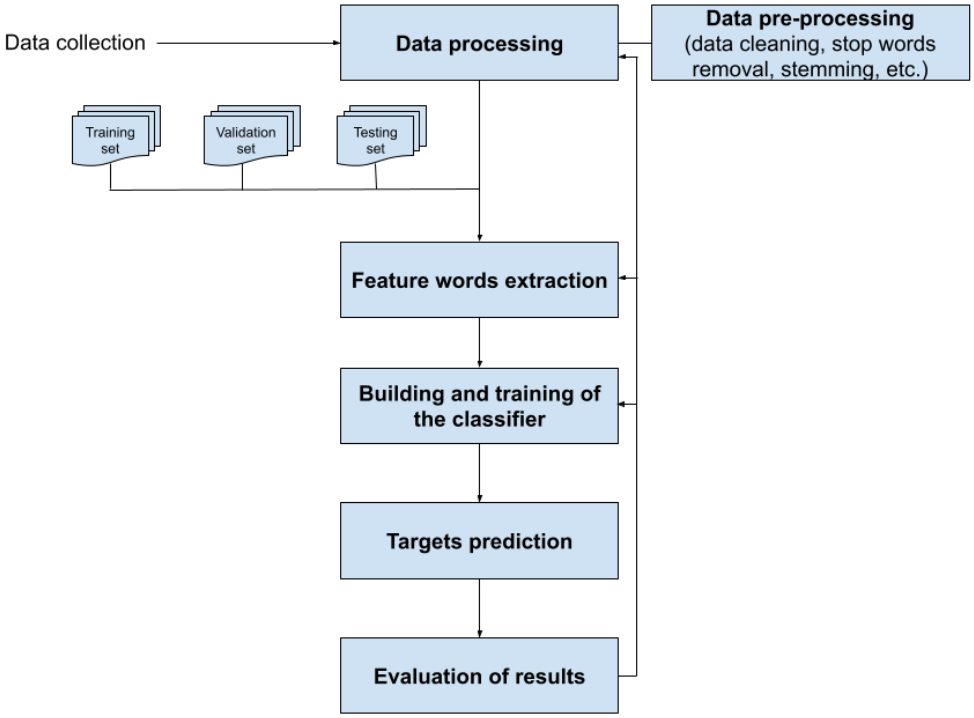


Figure 3: Text/Document data mining pipeline.

As we can see in Figure 3, the texts/documents dataset is randomly divided into a training, a validation and a testing set. All sets are processed separately, but following the same processing techniques. Then, each text/document is converted into a vector. Because there are too many feature words in each text, the dimension of the constructed vector will be too large, which will affect the accuracy of a classification [TLL07]. Therefore, it is necessary to select feature words for text classification in order to create an appropriate text representation model. Finally, machine learning techniques are used to construct a classifier and then the classifier is used to classify the testing data and suggest a category for the test document/text. Last, the result of the whole data mining process is evaluated and a score is produced.

2.2.1 Data pre-processing

Data pre-processing is the conversion of text/documents into usable data. The common data processing steps in natural language processing (NLP) include i) the data cleaning, ii) the tokenization which consists a prerequisite for step iii and iv, iii) the removal of stop words, and iv) the stemming.

2.2.1.1 Data cleaning

The texts/documents intended for data mining are often coming from web and pdf scaping and hence, they might contain empty spaces and symbols that are not of any value and therefore they need to be cleaned. The purpose of this step is to remove document's formats (incl. empty spaces), punctuation, symbols and special characters and convert the text to lower case.

2.2.1.2 Tokenization

Sentence tokenization splits the text into individual sentences and word tokenization splits the text into individual words. Usually, sentence tokenization is performed first and word tokenization comes after.

The sentence_tokenize and word_tokenize methods from the Natural Language Toolkit (NLTK) library are two well-known methods to perform tokenization. The word_tokenize method also slits punctuation as separate words. Therefore, someone can then use the .isalpha (or .isalnum) method to strip out non alphanumeric words/characters, such as numbers and punctuation. Another method to perform tokenization is by using the split() method from the String library to split the text into sentences and words.

2.2.1.3 Removal of stop words

Stop words are meaningless words that are so common in the text that have no value in natural language processing. These includes words like "articles, pronouns, prepositions, conjunctions, and auxiliary words". These words can be removed easily, by storing them and later calling them as a function. The NLTK library offers a list of stop words for several languages. Stop words are stored in the nltk_data directory and different stop words lists can be used by just calling the correct file in the directory.

Pre-processing might also include the removal of low or high frequency words. In some cases, these low or high frequency words might have an important influence on text classification, so they should not be removed [Yu08].

2.2.1.4 Stemming

Stemming reduces related words to a common stem. It is an optional process step and it is useful to test accuracy with and without stemming.

2.2.2 Data processing

In this step, the documents are loaded into a pandas dataframe, where the first column is named "label" and is used for preserving the target and the second column is named "text" and used for the text. Next, the label/target column is encoded into one-hot encoding so that it can be used in machine learning models and the text column is tokenized and transformed into a sequence of tokens and padding is used to ensure equal length vectors. Moreover, the dataset is split into training, validation and testing sets so that we can train and validate the classifier and then test the model into an unknown dataset.

2.2.3 Feature engineering

In the feature engineering step, raw text data is transformed into feature vectors and new features are created using the existing dataset. Several methods can be used in order to obtain relevant features, including:

2.2.3.1 Vectorization

Vectorization or embedding is the process of converting the text data into numbers. There are many different approaches to vectorizing words, and one of the simplest approaches is known as a “Bag of Words” approach that is used together with Countvectorizer, Tfidfvectorizer and Word Embeddings for feature extraction.

2.2.3.2 Countvectorizer

Countvectorizer tokenizes a corpus of documents and builds a vocabulary of all terms appeared in the corpus. Moreover, it encodes new documents creating a count vector for each document. The final outcome is a matrix in which every row represents a document from the corpus, every column represents a term of the vocabulary, and every cell represents the frequency count of a particular term appeared in a particular document.

2.2.3.3 Tfidfvectorizer

As Countvectorizer uses the frequency counts of terms for obtaining the output vectors, Tfidfvectorizer uses the relative importance of a term in the document and the entire corpus represented by the Term Frequency – Inverse Document (TF-IDF) frequency. By this way, words like “the” with large counts will be disregarded obtaining a low score.

Tf-idf score is composed by two values: the first is the normalized Term Frequency (TF), the second is the Inverse Document Frequency (IDF), calculated as follows:

$$TF(t) = \frac{\#\ of\ times\ term\ t\ appears\ in\ a\ document}{\#\ of\ all\ terms\ in\ the\ document} (1)$$

$$IDF(t) = log_{10} \frac{\#\ of\ all\ documents}{\#\ of\ documents\ where\ term\ t\ appears} (2)$$

These two values are multiplied to produce the final tf-idf score. The higher the value, the rarer the term. The smaller the value, the more common the term. Tf-idf vectors can be generated at different levels of input tokens, including words, characters, or n-grams.

2.2.3.4 Word embeddings

A word embedding is a form of representing words and documents using a dense vector representation. Different algorithms have been developed to generate embeddings. These can be divided into two broad groups [BDK14]: i) the methods that work with a co-occurrence word matrix, such as the Global Vectors (GloVe) [PSM14]; and ii) the predictive methods, which try to predict neighboring words given one or more context words, such as Word2Vec [MCC+13].

The position of a word within the vector space is learned from text and is based on the words that surround the word when it is used. Word embeddings can be trained using the input corpus itself or can be generated using pre-trained word embeddings such as Glove, FastText, and Word2Vec. Any one of them can be downloaded and used as transfer learning. Below, the basic characteristics of these three language models are detailed.

FastText is a recently developed method [JGB+16] in which embeddings are associated to character n-grams, and words are represented as the summation of these representations. In this method, a word representation is induced by summing character n-gram vectors with vectors of surrounding words. Therefore, this method attempts to capture morphological information to induce word embeddings.

Word2Vec [MCC+13]is a widely used method in NLP for generating word embeddings. It has two different training strategies: i) Continuous Bag-of-Words (CBOW), in which the model receives as input a sequence of words without the middle one, and attempts to predict this omitted word; ii) Skip-Gram, in which the model receives as input a word and attempts to predict its neighboring words. In both cases, the model consists of only a single weight matrix (apart from the word embeddings), which results in a fast log-linear training that is able to capture semantic information.

The Global Vectors (GloVe) method [PSM14] consists a co-occurrence matrix M that is constructed by looking at context words. Each element Mij in the matrix represents the probability of the word i being close to the word j.

2.2.4 Classification techniques

Classification algorithms are an important part of text classification and machine learning. Their mission is a process of mapping test data to specific targets based on the model obtained from the training data. The current classification algorithms are mainly based on: i) statistical learning algorithms, such as Bayesian networks (e.g., Naïve Bayes), Support Vector Machines, instance-based learning algorithms (e.g., k-Nearest Neighbor), ii) neuron connection algorithms (perceptron-based), such as neural networks, and iii) logic-based algorithms, for example decision trees and rule-based classifiers [KZP06].

2.2.4.1 Naïve Bayes (NB)

Naive Bayes (NB) is a probabilistic classification technique based on Bayes' Theorem with an assumption of independence among predictors. This means that the classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

2.2.4.2 Linear (Logistic Regression - LR)

Logistic Regression (LR) measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic/sigmoid function.

2.2.4.3 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised ML algorithm which can be used for both classification and regression problems. The algorithm finds the best possible separated hyperplane (line) so that it can segregate different categories maximizing the distance between the different categories.

2.2.4.4 Random Forest (RF)

Random Forest (RF) is a type of ensemble models, particularly bagging models. It is composed of many decision trees, where there is no correlation between them. When we perform a classification task, new input samples are entered in the random forest and its decision trees. After that, each decision tree will obtain its own classification result and, at the end, the classifier can select the most voted category by summarizing all the decision tree categories.

2.2.4.5 Shallow neural network

A neural network is a mathematical model that is designed to behave similar to biological neurons and nervous system. These models are used to recognize complex patterns and relationships that exists within a labelled dataset. A shallow neural network contains mainly

three types of layers, including the input layer, the hidden layer, and the output layer (Figure 4).

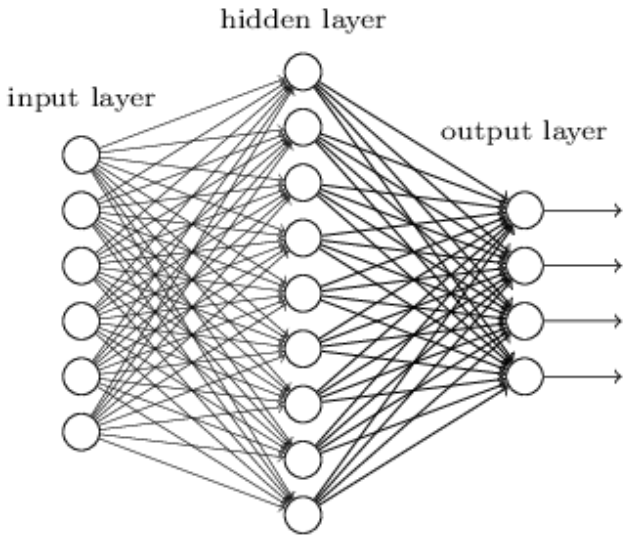


Figure 4: Architecture of a shallow neural network.

2.2.4.6 Deep neural networks

Deep neural networks are more complex that shallow neural networks in which the hidden layers are usually more and perform more complex operations. In Figure 5, we present different types of deep neural networks that are applied in text classification and automated patent classification problems.

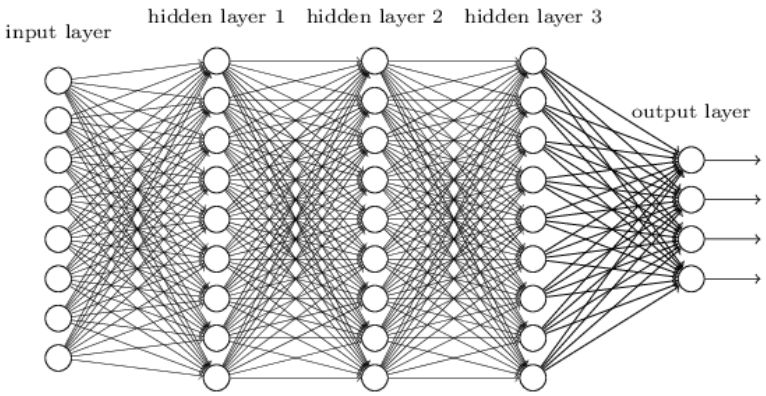


Figure 5: Architecture of a deep neural network.

Convolutional Neural Network (CNN)

In Convolutional neural networks (CNN), convolutions over the input layer are used to compute the output. This results in local connections, where each region of the input is connected to a neuron in the output. Each layer applies different filters and combines their results (Figure 6).

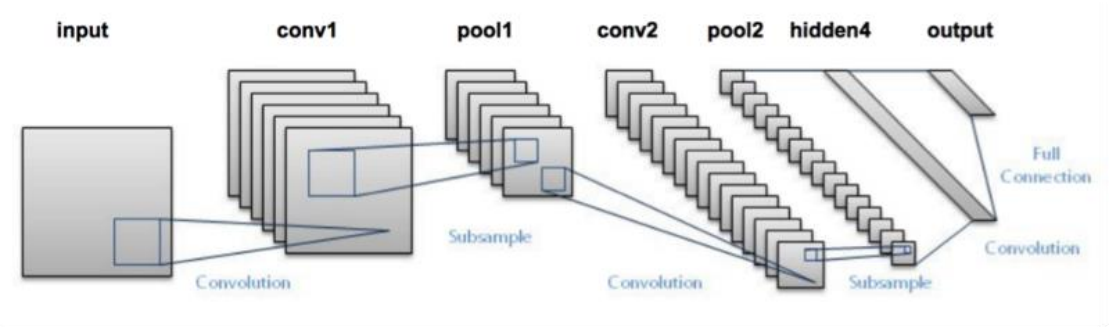


Figure 6: Architecture of a convolutional neural network.

Recurrent Neural Network (RNN)

Another type of neural network suitable for sequential data is a Recurrent Neural Network (RNN). Unlike feed-forward neural networks in which activation outputs are propagated only in one direction, the activation outputs from neurons in RNN propagate in both directions (Figure 7). This creates loops in the neural network architecture which acts as a 'memory state' of the neurons and allows them to remember what have been learned so far. Despite the advantage of the memory state, RNNs present the problem of Vanishing Gradient, where it becomes hard for the network to learn and tune the parameters of the earlier layers while learning with a large number of layers.

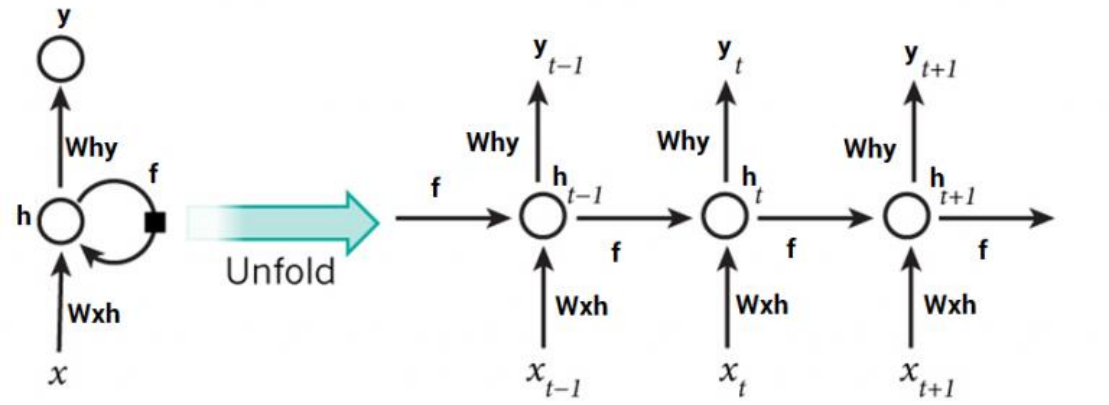


Figure 7: Architecture of a recurrent neural network.

Long Short-Term Memory (LSTM)

The Long Short-Term Memory (LSTM) [HS97] is a special kind of RNN, which has been introduced to improve the vanishing gradient problem of back propagation through time appeared in RNN. The LSTM consists of three parts, the forget gate, the input gate and the output gate. The first gate chooses whether the information coming from the previous state vector is to be remembered or is irrelevant and can be forgotten. The second gate learns new information from the current vector. At last, the third gate passes the updated information from the current to the next vector.

Gated Recurrent Units (GRU)

Gated Recurrent Unit (GRU) [CMB+14] can be considered as a variation of the LSTM because they are both designed to solve the vanishing gradient problem of RNNs. The GRU uses two vectors, the update gate and the reset gate. Basically, these vectors decide what information should be passed to the output, keeping information from long ago, without washing it through time, or remove information which is irrelevant to the prediction.

Bidirectional RNN (BRNN)

Bidirectional recurrent neural networks (BRNN) connect two independent RNNs of opposite directions together forming two hidden layers. With this form of generative deep learning, the output layer can get information about the sequence from past (backwards) from one network and future (forward) states from another simultaneously.

BRNNs do not require their input data to be fixed, while their future input information is reachable from the current state in contrary to standard RNNs where the future input information cannot be reached from the current state. BRNN are especially useful when the context of the input is needed.

2.2.4.7 Ensemble methods

Ensemble method is a machine learning technique that combines several base models (often called "weak learners") that are trained to solve the same problem in order to produce one optimal predictive model and thus better results. The main hypothesis is that when weak models are correctly combined we can obtain more accurate and/or robust models.

An ensemble of classifiers is a set of classifiers whose individual decisions are combined in a way [Die00]. The most typical way to combine them is by using weighted or unweighted voting to classify new entries. Ensembles are often much more accurate than the individual classifiers that make them up.

3

Related work

Here we describe the prior work made by other scientists on automatic patent classification in which we have discovered relevant approaches to the current thesis.

3.1 Automatic patent classification

Several works have been conducted to solve the automated patent classification problem. Actually, patent classification methods advance hand-to-hand with the text and document classification that are typical Natural Language Processing (NLP) applications which deal with the assignment of one or multiple pre-defined labels to a given text or document, respectively [KJH+19] [AKG+19].

Earlier research on patent classification applied basic NLP and feature engineering techniques to pre-process texts before feeding them into well-known classifiers. For example, Fall et al. [FTB+03] performed stop words removal, stemming, term selection using the information gain and then fed the transformed texts into Naïve Bayes (NB), Support Vector Machine (SVM), and K-Nearest Neighbor (KNN) classifiers. Tikk et al. [TBT08] applied stop word removal, stemming, dimensionality reduction and removal of rare terms, and they imported the processed output into a neural network, named HITEC, which copies the tree-like structure of the taxonomy. Similarly, Lim and Kwon [LK16] constructed a stop words list dedicated to a patent domain while they selected their feature set using a TF-ICF weighing scheme, which is a variation of the well-known TF-IDF.

Around 2017, research on automated patent classification turned to test out the effectiveness of new DL methods for text processing tasks. Among them, Grawe et al. [GMB17] used stop word removal and then transformed the cleaned text into a meaningful representation produced by the Word2Vec that was used as input for an LSTM network (a type of RNN). Likewise, Xiao et al. [XWZ18] used Word2Vec embeddings and LSTM to classify patents in the security field.

Another trend that is recent in the relevant literature is the training of Word Embeddings on domain-specific patent datasets. Risch and Krestel [RK19] used the FastText embedding that was trained on a patent dataset together with bidirectional GRUs (another type of RNN) to achieve better performance compared with Word Embeddings trained on Wiki documents. Moreover, Sofean [Sof19] developed a self-trained Word Embedding trained on million patents and then used a LSTM network to perform patent classification.

Another improvement in patent classification with respect to the DL techniques started with the adoption of CNNs. Li et al. [LHC+18] proposed a DL algorithm, DeepPatent, for patent classification by combining word vector representation and a well-designed CNN. Likewise, Zhu et al. [ZHF+20] used the Word Embedding technique to segment and vectorize the input data and then a symmetric hierarchical CNN, named PAC-HCNN, to classify patents outperforming traditional RNN. Moreover, Abdelgawad et al. [AKG+19] compared several recent neural network models and showed that CNNs are a suitable choice for patent classification. The authors also proved that state-of-the-art hyperparameter optimization techniques can further improve the CNN performance.

Recent trends on DL that includes pre-trained unsupervised language models on large corpuses and fine-tuning them on downstream tasks have produced state-of-the-art performance. Among them, the BERT model, released by Google in 2018, is the first deeply bidirectional, unsupervised language representation, pre-trained using only a plain text corpus in order to infuse later the context in which it will be used. Following this trend, Lee et al. [LH0] leveraged and fine-tuned the BERT-Base model and applied it to patent classification getting better accuracy than other recent DL approaches. Similarly, Roudsari et al. [RAL+20] presented a short work applying and fine-tuning the Distil-BERT model for patent classification.

Innovative architectures have also been proposed working on feature extraction from patent text. Bai et al. [BSP20] proposed a multi-stage feature extraction network, named MEXN, which divides the document to fixed-sized paragraphs and extracts features, then summarizes the paragraph features into a document feature using the attention mechanism, and finally computes the weight matrix with hierarchical category information of the patent. Similarly, Hu et al. [HHY+18] proposed a hierarchical feature extraction model, named HFEM, which is able to capture both local features of phrases as well as global and temporal semantics.

Another important aspect that is worth mentioning is the limited number of works considering the hierarchy of the IPC/CPC codes. In principle, flat methods implement classification using only a single level (e.g. sub-class), while hierarchical methods exploit the knowledge of hierarchy (previous associated codes at different levels) to predict an effective classification. Although flat methods seem to dominate the literature in patent classification techniques, there are still some outstanding examples. Chen and Chang [CC12] presented a novel patent

classification method that combines flat text classification in two different levels of the IPC hierarchy with clustering method to perform patent classification down to the subgroup level. In [GSP15], Giachanou et al. exploited the hierarchical structure of IPC codes to propose a multi-layer collection selection method that can be seen as a patent classification problem. Their method, in addition to utilizing the topical relevance of IPCs at a particular level of interest, exploits the topical relevance of their ancestors in the IPC hierarchy and aggregates those multiple estimations of relevance to a single estimation. Furthermore, Risch et al. [RGK20] used a sequence-to-sequence neural network with label embeddings to generate a sequence of labels for all levels where the embeddings of previous level are considered for generating the next labels.

Last but not least, some researchers tried to identify which parts in a patent document can provide more representative text for classification tasks [LK16, LH0]. Some researchers distinguish the use of technical and background parts extracted from the descriptions section [LK16], the first claim [LH0], the title and abstract [LHC+18, ZHF+20], or at most cases, the title, abstract claims and description [RK19, Sof19, HHY+18, AKG+19].

Table 1 presents the key characteristics of research works discussed above.

Table 1: Comparative table of prior work

Prior work	**Machine learning methods**	**Hierarchy**	**IPC Level**	**Patent text**	**Dataset**	**Accuracy**
[CC12]	Three phase categorization	Yes	1238 sub-groups (Sub-groups appearing in less than 7 patents were excluded)	-	WIPO-alpha	36.07%
[RGK20]	Sequence-to-sequence neural network & FastText trained on patent data	Yes	632 sub-classes	Title and abstract	UPSTO-2M	53.2% & 56.7% for basic and improved method
[GMB17]	LSTM & Word2Vec trained on patent data	No	Sub-class	Title and abstract	USPTO	57% & 63% for 400 and 150 retrieved words
[XWZ18]	LSTM & pre-trained Word2Vec	No	Group	-	Security field	93.48%
[LHC+18]	DeepPatent - CNN & pre-trained Word2Vec	No	Sub-class	Title and abstract	D1: USPTO-2M; D2: CLEF-IP	Recall (top 1) is not given 75.46% Recall (top 4)
[RK19]	Bidirectional GRU & FastText trained on patent data	No	Sub-class	D1 title, abstract, claims, desc; D2: title and abstract.	D1: WIPO-alpha; D2: USPTO-2M.	49% & 53% for D1 and D2
[BSP20]	MEXN - Feature extraction & pre-trained and fine-tuned BERT	No	Section and sub-section	Claims and Description	USPTO	54.95% & 66.69% section; 39.86% & 51,56% sub-section for pre-trained and fine-tuned BERT
[Sof19]	LSTM & Word2Vec trained on patent data	No	43 sub-classes (sub-classes appearing in more than 500 docs)	Title, abstract, desc, claims, inventors, and assignees	EPO and WIPO	67%
[LK16]	Naïve Bayes	No	Sub-class	Title, Abstract, Technical Fields, Backgrounds		87.2 % & 70% Precision (single – all)

Prior work	Machine learning methods	Hierarchy	IPC Level	Patent text	Dataset	Accuracy
[HHY+18]	BiLSTM & Combining features vectors from all sections	No	96 classes	Title, abstract, description and claims	CLEF-IP 2011 (Section F)	54.99% Recall (top 1)
[ZHF+20]	PAC-HCNN - Hierarchical CNN & Word2Vec trained on patent data	Yes	5 sections and 50 classes	Title and abstract	D1: Chinese patents; D2: CNKI	87.8% section, 76,1% class
[AKG+19]	Optimized CNN & GloVe, FastText, Word2Vec trained on patent data	No	451 sub-classes	Title, abstract, claims, and description	WIPO-alpha	55.02%
[LH20]	PatentBERT - CNN & fine-tuning BERT model	No	Sub-class	D1: Title and abstract; D2: IPC and claims; D3: CPC and claims	D1: USPTO-2M D2: USPTO-3M	54.33%, 53.36%, 55.38% Recall (top 1) for D1, D2 and D3

4

Methods adopted in the study

Here follows the introduction of methods and algorithms examined in the thesis. The selected data pre-processing and processing methods come first. Then, the feature engineering methods that were performed follow. Last, the classification algorithms and architecture that were used are presented.

4.1 Data pre-processing

The initial pre-processing steps that we enforced on the raw text of each patent were to break it into sentences using the sent_tokenize() method of the NTLK library, convert it to lower case with the lower() method of the string library, and delete empty spaces, punctuation, symbols and special characters. The produced text was further split into individual words using the word_tokenize() method of the NTLK library. The produced words were checked for validity using the .isalpha() method of the NTLK library, which strips out non alphanumeric words/characters such as numbers and punctuation. Afterwards, stop words removal and stemming have been applied on the cleaned and shopped words of the patent text before re-joining them back into a single string. For the stop words removal, we used a domain-specific stop word list that combines English stop words, corpus-specific stop words, Sklearn stop words, and USPTO stop words, while for the stemming, we used the PorterStemmer.

4.2 Data processing

As described in detail in the data collection section (5.3.1), we initially created a data collection containing text coming from all sections for all patents (418,788 patents) that include the <main classification> tag and later we filtered out patents that do not contain all different sections (i.e., an abstract, a description and a claims section) at the same time, concluding to 302,578 patents. Thereafter, we created three additional data collections containing the "title" and the "abstract", the "description" and the "claims" parts, respectively.

For the creation of different data collections, we worked with the SGML files of patent documents coming from the CLEF-IP 2011 test collection, which have been created and released by IR Lab for several IR-related purposes. Each SGML file abstracts the most significant information of the XML file of the respective patent. An SGML file is structured as displayed in Figure 8.

```
<DOC>
<DOCNO>
EP-0001752
</DOCNO>
<TEXT>
<DATE>
19790516
</DATE>
<IPC-CLASSIFICATIONS>
mainsubclassF16Cmainsubclass furthersubclassF16Cfurthersubclass mainsubgroupF16C11slash04mainsubgroup
furthersubgroupF16C27slash06furthersubgroup mainsubclassF26Cmainsubclass
</IPC-CLASSIFICATIONS>
<TITLE>
resilient pivotal joint.
</TITLE>
<APPLICANT>
trw incus<sep>trw inc.<sep>trw inc.23555 euclid avenuecleveland ohio 44117us<sep>trw inc. <sep>
</APPLICANT>
<INVENTOR>
herbenar edward john<sep>herbenar, edward john<sep>herbenar, edward john5965 whethersfield lane apt. 29bbirmingham
michigan 48010us<sep>herbenar, edward john<sep>herbenar, edward john5965 whethersfield lane apt. 29bbirmingham
michigan 48010us<sep>
</INVENTOR>
<ABSTRACT>
an improved pivot joint (10a) has a rotatable and tiltable  stud (12a). a rotational preload force is applied to the
stud  (12a) by a bearing (16a) which is circumferentially stressed in  tension to apply a clamping force to the stud
(12a). this  clamping force resists rotation of the stud (12a) relative to a  housing (14a) until a predetermined
minimum rotational  force is applied to the stud (12a). a tilting or sidewise preload  force is applied to the stud
(12a). a tilting or sidewise preload  force is applied to the stud (12a) by a resiliently compressible  bushing (18a)
which is disposed between the bearing (16a)  and the housing (14a). since the tilting preload force is  applied to
the stud (12a) by the bushing (18a), the bearing  (16a) can be designed to provide an optimum rotational preload
force without regard to tilting preload force design considerations.  similarly, the bushing (18a) can be designed
to  provide optimum tilting preload forces without regard to  design considerations influenced by the rotational
preload  forces. the pivot joint (10a) is advantageously assembled by  resiliently expanding the bearing (16a),
moving the stud  (12a) into alignment with the bearing (16a) then releasing the  bearing (16a) so that it contracts
part way back to its initia  condition. the diameter of the stud surface (26a) engaged by  the bearing (16a) is large
enough so that the bearing (16a)  cannot fully retract back to its initial condition. this results in  tension
stresses being set up in the bearing (16a) so that it  applies a clamping force against the stud (12a) to hold it
against rotation. the bearing (16a) is provided with stop  means (112) which limit sideward tilting movement of the
stud (12a).
</ABSTRACT>
</TEXT>
</DOC>
```

Figure 8: Example of an SGML file containing the patent text.

For each data collection, the relevant patent tags from the SGML files were identified using the BeautifulSoup library. Thereafter, the content of these tags was copied in a new TXT file. For the initial data collection, the main subclass category was copied first and followed by the content extracted from the title, the abstract, the description and finally the claims tag. For the second data collection which contains the title and the abstract sections, the main subclass category was copied first and followed by the content extracted from the title and the abstract

and similarly for the other data collection. By this way, each SGML file (i.e., a patent document) provides content in a unique row of each data collection, while the total lines of all four data collection are 302,578, similar with the total number of patents chosen for the whole data collection.

Thereafter, the TXT file of each pool is loaded into a pandas dataframe, where the first column was named "label" and used for preserving the patent's target, while the second column was named "text" and used for patent text. An example of a dataframe is presented in Figure 9.

	text	label
0	pharmaceutical composition for administering c...	A61K
1	pyrazino-benzoxazepine and -benzthiazepine der...	C07D
2	method of mounting printing cylinders utilizin...	G03G
3	infra red light emissive devices. in (sb0.1as0...	H01L
4	method and apparatus for rotationally moulding...	B29C

Figure 9: Example of the dataframe where the patent text and patent category (IPC code) has been loaded.

After the selection of the most representative portion of text for each patent, this text was tokenized and transformed into a sequence of tokens, while padding was used to ensure equal length vectors. Tokens were then mapped to embeddings and an embedding matrix representing each patent document has been created. More details regarding the feature engineering will be provided in the next section.

On the other hand, the patent target derived from the column "label" of the dataframe was encoded using the OneHotEncoder() method of keras.preprocessing library.

Moreover, the dataset of 302,578 patents was split into training, validation and testing sets: 80% for training, 10% for validation, while other 10% was kept out for testing the classification model.

4.3 Feature engineering

4.3.1 Feature selection

The text of each patent document is lengthy, written in very technical language and consisted of huge vocabulary and noisy words. Due to the large volume of patent texts, the vector space dimension obtained by the patent text is particularly high. The reduction of the vocabulary size and noisy words by considering the most important sections and feature words of patent texts will be the key in the classification. Maybe there is some highly relevant section or words in a

patent document (appearing either in a single or different parts) from which we can obtain a suitable patent representation. Therefore, it is important to extract this suitable feature vocabulary for classification experiments [TLL07]. Below, we describe different approaches to represent the patent document.

4.3.1.1 First "X" words of the patent text

In this representation, we extracted the first "X" words from the patent text for the representation of a patent document, which is a method commonly found in several literature works [GMB17, XWZ18, LHC+18, RK19].

In the initial data collection, the patent text is the concatenated result of the title-abstract, the description, and the claims sections (placed the one after the other). This means that most of the words used to represent the patent document are taken from the title-abstract section since the average number of words from the title-abstract section is 102 and can reach 7,692 words at the maximum (see Table 3). This leaves out a (often small) number of words which comes from the description section, while words from the claims section will not be used in the final "X" words length representation of the patent document. We repeated the same methodology for the other three "mono-thematic" data collections retrieving each time the first "X" words coming from title and abstract data collection, the description data collection and the claims data collection. For all data collections, the different number of words, called "X", that has been retrieved to represent the patent document was: {20, 40, 60, 80, 100, 200, 300, 400}.

4.3.1.2 First "Y" words for all patent parts

In this representation, we used the first "Y" words of each part for the representation of the patent document. This means that we created a batch of words consisting of the first "Y" words from the title-abstract text, the first "Y" words from the description text, and the first "Y" words from the claims text. To achieve this, we loaded the three different "mono-thematic" data collections into three pandas dataframes. Then, we created a new pandas dataframe retrieving the first "Y" words from the "text" column of each "mono-thematic" dataframe and placing these batches of "Y" words the one after the other into the "text" column of the new dataframe. We keep the total number of words retrieved from all sections same with above: {20, 40, 60, 80, 100, 200, 300, 400}.

4.3.1.3 First "X" significant words of the patent text

In this representation, we used the first "X" most important words sorted by their tf-idf score. This has been done for all four data collections that means that each time the first "X" most important words are coming either from the title and abstract data collection, the description

data collection, the claims data collection or the all sections data collection. Again, we keep the different number of words that needs to be retrieved from all sections same with above: {20, 40, 60, 80, 100, 200, 300, 400}.

4.3.2 Feature representation

4.3.2.1 Tf-idf vectors as features

For detecting and sorting the words of the patent text based on their significance, the scikit-learn's methods of either CountVectorizer – TfidfTransformer or TfidfVectorizer can be used that calculate the tf-idf score of each word of a document and convert the document into a matrix of tf-idf features. However, we noticed that these methods use a different way from the standard notation for tf-idf to calculate the tf-idf scores. More specifically, the standard notation for tf-idf defines the idf as:

$$\mathrm{idf(t)} = \log\frac{\mathrm{n}}{1+\mathrm{df(t)}} \quad (3)$$

The scikit-learn's methods enforces by default smoothing of idf weights by adding one to document frequencies, as if an extra document was seen containing every term in the collection to prevent zero divisions.

$$\mathrm{idf(t)} = \log\frac{1+\mathrm{n}}{1+\mathrm{df(t)}} + 1 \quad (4)$$

Even though the smoothing parameter is disactivated, the "1" count is added to the total idf instead of the idf's denominator.

$$\mathrm{idf(t)} = \log\frac{\mathrm{n}}{1+\mathrm{df(t)}} + 1 \quad (5)$$

Another deviation from standard notation in the scikit-learn's methods is that the resulting tf-idf vectors are normalized by the Euclidean norm.

Because of these deviations from standard notation for tf-idf, we decided to implemented a method that will calculate the tf-idf scores following the default equation for the idf (3) and without enforcing any normalization is the resulting tf-idf vectors.

4.3.2.2 Word embeddings

Pre-trained models

For the transformation of tokens to embeddings, we used the pre-trained language models of FastText, Word2Vec, and Glove, with dimension size set to 300 for the generated word vector.

Word2Vec

We trained the Word2Vec language model in our corpus (the corpus of 302,578 patent documents) and then we used the produced embeddings to represent the patent texts. For the training of Word2Vec, we used the genism toolkit and set the vector size equal to 300, the maximum distance between current and predicted word equal to 8 and the number of iterations equal to 20. The outcome embedding matrix was later used as input to the neural network architectures, which consist of several layers and returns as outcome a probability for each target.

4.4 Classification algorithms

This section describes the neural network architectures and DL methods that were deployed. Although the DL models can be improved by further tailoring their structure with additional layers and fine-tuning the different parameters, we selected to proceed with a baseline version of selected neural networks along with an improved version of them following a quite basic optimization approach for each model.

4.4.1 CNN

CNN is one of the neural networks that have been widely used recently in the domain of automated patent classification. Figure 10 depicts the architecture of the initial CNN model used in the experiments. In our method, the pre-processed text is fed to a CNN through an embedding lookup, which converts words to vectors represented in a high-dimensional vector space. Afterwards, a 1-D convolutional layer is applied on top of the embedding layer. A max-pooling along with a flatten layer are then applied sequentially to the output of the convolutional layer. After a dense of 1024 filters, the output is fed in another dense layer with a softmax activation in order to obtain a probability distribution over all targeted labels (IPC sub-classes).

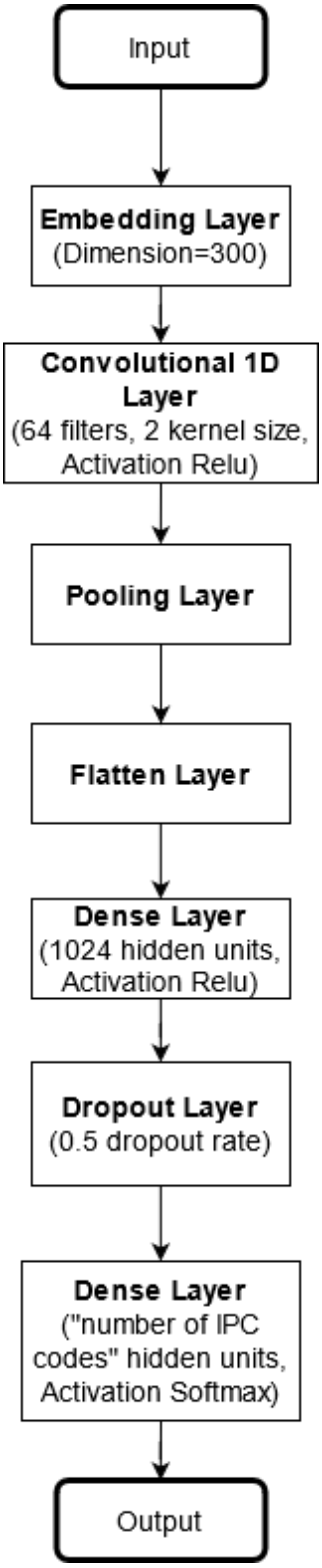


Figure 10: Initial CNN architecture adopted in the experiments.

4.4.2 RNN

Besides CNN, RNN is another category of neural networks that have been used in text classification. Figure 11 presents the architecture of the initial RNN model, including a bidirectional LSTM, a bidirectional GRU, a LSTM and a GRU, used in the experiments.

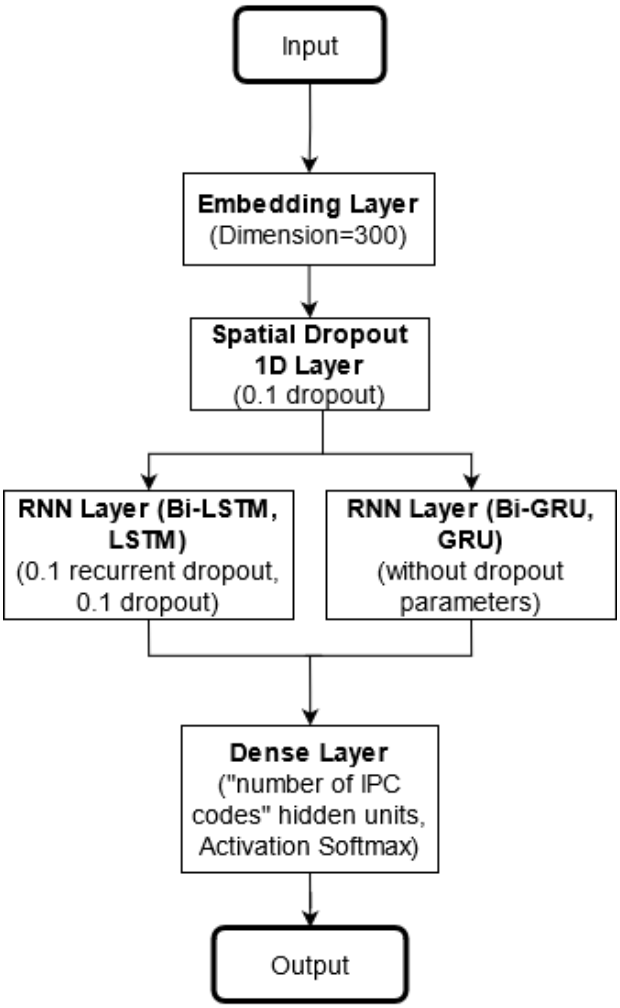


Figure 11: Initial Bi-LSTM/LSTM and Bi-GRU/GRU architecture adopted in the experiments.

More specifically, LSTM and GRU have been proposed as variations of RNN to deal with the exploding gradient and vanishing gradient problems of back propagation through time. Since they are experiencing good results, they have been applied in many fields including automated patent classification. Here, we evaluate both LSTM and GRU methods as they are superior to simple recurrent units. Moreover, we use bidirectional LSTM and GRU so that the input sequence to be processed in two directions. Similar with CNN architecture, we convert each word to its embedding. This embedding layer of words is processed by a spatial dropout, which randomly masks 10% of the input words to make the neural network more robust. The remaining 90% serves as input to the LSTM or GRU layer (also to the bidirectional LSTM and

bidirectional GRU). The output of these steps is inserted in a dense layer with as many units as the targeted labels and a softmax activation.

4.4.3 Ensemble methods

In this thesis, we used an ensemble method to combine classifiers that are trained on different patent parts in order to get better results. We evaluated the combination of same classifiers (either CNN, LSTM, GRU, bi-LSTM or bi-GRU) and for their combination we used the average score of the predicted probability for each target coming from each individual. Figure 12 depicts the architecture of the ensemble method.

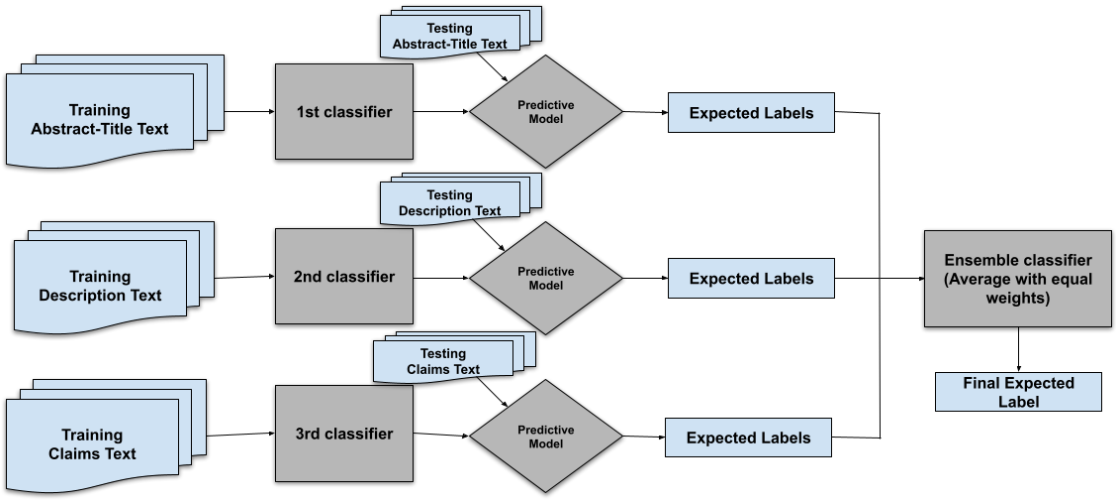


Figure 12: An ensemble architecture of classifiers adopted for our experiments.

5

Evaluation methodology

Here we will present the evaluation methodology for assessing our techniques. More specifically, we describe the metrics for evaluating our experiments, the evaluation system which consists an end-to-end single-label patent classification pipeline, the data collection, and the different sets of experiments that we conducted to address the automatic patent classification problem down to the sub-class level.

5.1 Evaluation criteria for single-label patent classification

A classifier which has been employed for a single-label text classification task might predict the expected label (or not). Hence, this case should be considered in order to evaluate the classifier. For each experiment, we used the following evaluation metrics to evaluate our methods in single-label patent classification task:

5.1.1 Accuracy

Accuracy refers to the probability of relevant results of the classification. Specifically, in case of single-label classification, accuracy evaluates the first prediction of the classifier for each patent document. If it is a correct relevant result, the numerator is increased by one, while the denominator is equal to the number of patent documents in the testing set.

$$Average\ accuracy = \frac{\#\ of\ correct\ predictions}{\#\ of\ all\ the\ patents\ in\ the\ testing\ set} \quad (6)$$

5.1.2 Mean reciprocal rank (MRR)

MRR is defined as the mean of the inverse rank of the first correct relevant result, taken over all n patent documents and its value ranges from 0 to 1. This metric is useful to evaluate whether the classification system returns the best relevant item at the higher position.

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i} \quad (7)$$

where, |Q| denotes the total number of questions/documents and $rank_i$denotes the rank of the first relevant result/category.

For example, if the relevant result is present at rank 1, the MRR score is $\frac{1}{rank_i} = \frac{1}{1} = 1$, while if the relevant result is present at rank 5, the MRR score is $\frac{1}{rank_i} = \frac{1}{5} = 0.2$. For multiple different queries, we can calculate the MRR by taking the mean of the reciprocal rank for each query. In the above case, it would be $MRR = \frac{(1+0.2)}{2} = 0.6$.

5.1.3 Success at n (S@n)

Success at n is the probability of the first correct relevant result to be found in the top-n predictions of the classification. In case of single-label classification, with success at n metric, we evaluate the top-n predictions for each patent document and if one of them is the correct relevant result, the numerator is increased by one. The denominator is equal to the number of patents in the testing set.

$$Success\ at\ n = \frac{\#\ of\ correct\ predictions\ up\ to\ position\ n}{\#\ of\ all\ the\ patents\ in\ the\ testing\ set} \quad (8)$$

5.2 Evaluation system

This section briefly describes the end-to-end single-label patent classification pipeline we implemented in order to evaluate our methods (Figure 13).

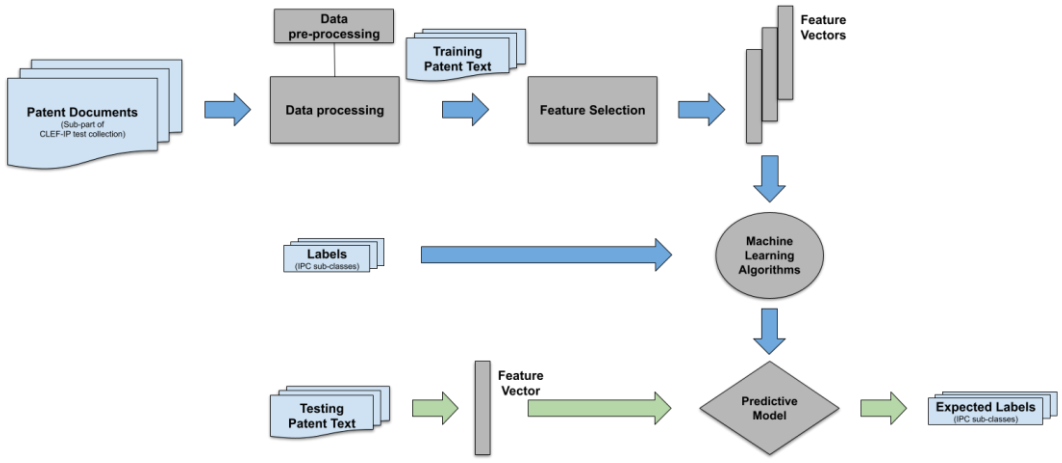


Figure 13: End-to-end single-label patent classification pipeline.

Initially, the patent documents are inserted in the data processing box where the dataset of 302,712 patents is split into training, validation and testing sets (80-10-10). We remind at this point that the selected patents contain the main classification category, i.e. they include the

<main classification> tag, while they also contain a title-abstract, a description and a claims section in the patent document at the same time.

From the patent text of the training and validation sets, we select the sections/words that we will use for feature engineering. These feature words are then tokenized and transformed into a sequence of tokens, while padding was used to ensure equal length feature vectors. Tokens are then mapped to embeddings and an embedding matrix representing each patent document is created. FastText, Word2Vec, and Glove pre-trained language models and Word2Vec trained on the same patent corpus are evaluated in this thesis. Then, the embedding matrix is used as input in a deep learning model, which consists of a number of layers and returns a prediction for each target, i.e., classification IPC code. In parallel, the labels of patents are encoded using one-hot encoding and inserted in the machine learning algorithm box.

The patent text of the testing set is applied a similar processing and feature engineering procedure and the outcome of these steps is used as input in the prediction model produced by the machine learning algorithm box. The predicted label is compared with the expected label and performance metrics are calculated to evaluate the whole patent classification task and its individual methods.

5.3 Data collection

For evaluating our techniques, we used the CLEF-IP 2011 test collection. From all patent documents in the test collection, we kept the documents that have English text, pertaining to 1,149,536 patent documents. From those patents, only 418,788 explicitly have the required information of which is their main classification category, i.e. include the <main classification> tag. This information was mandatory to determine which is the primary classification code, so a single-label classification method can be reliably executed and evaluated.

From those patents, we extracted the “title”, the “abstract”, the “description” and the “claims” sections, as well as the “main classification”. Using this content, we created four different pools of patents: i) the initial pool which contains all 418,788 patents along with all text from the aforementioned sections; ii) a second pool which contains 335,751 patents, those that include the title and the abstract section in the patent document; iii) a third pool which contains 380,799 patents, those that include the description section in the patent document and; iv) a last pool which contains 380,841 patents which contain the claims section in the patent document. These different pools have been created for testing if using different sections will have an effect on the classification task.

These pools remain unequal because even though a patent document may have an abstract section and thus it is listed in the second pool, it may miss a description section and thus it is

excluded from the third pool. In order to make these pools uniform so that they contain exactly the same patents, we filtered out those patents that do not contain a title-abstract, a description and a claims section in the patent document at the same time. By this way, from the initial 418,788 only 302,578 patent documents have been remained. Table 2 shows some basic statistics regarding the number of words per each section in the sub-dataset of 302,578 patent documents that we used:

Table 2: Statistics of different patent parts

	Title-Abstract	Description	Claims
Min num words	2	1	1
Max num words	7,692	12,030	61,972
Mean num words	102	457	1,044

The patent documents chosen by CLEF-IP 2011 dataset based on the aforementioned criteria are spanning the different eight ICP categories with the below distribution (Table 3):

Table 3: Patents distribution across the eight sections

Categories	Number of patents
Section A	3,9643
Section B	48,144
Section C	89,331
Section D	5,236
Section E	6,119
Section F	19,559
Section G	48,437
Section H	46,109
Total	302,578

5.4 Experimental process

Here we describe in detail how we organized the experiments.

5.4.1 Experimental setup 1: Exploration of optimal deep learning model

This experimental layout explores the effect of different deep learning models on patent classification results. After a number of improvements in the structure of the initial CNN and

RNN models and in the chosen parameters by fine-tuning them, the experimental layout was repeated for the improved deep learning models.

In the frame of this experimental setup, we also investigated the ensemble method of combining the results of three similar classifiers which run in parallel in the three different parts of the patent document.

5.4.2 Experimental setup 2: Exploration of optimal feature selection for patent representation

Different batches of words coming from different parts of the patent document may result into different outcomes on patent classification. These experiments investigated how the patent representation may affect the patent classification results, including:

- First "X" words of a patent part
- First "Y" words from all patent parts
- First "X" significant words of a patent part

5.4.3 Experimental setup 3: Exploration of optimal language model for patent representation

In the current experimental setup, we explored how different embedding representations can affect the patent classification outcomes, including:

- The pre-trained language models of FastText, Word2Vec, and Glove
- The Word2Vec language model trained in a patent-domain corpus

5.4.4 Experimental setup 4: Exploration of optimal sub-collection

As mentioned above, the distribution of patents among categories are highly unbalanced with about 80% of all the documents classified in about 20% of the categories. A similar unbalance was noticed examining the distribution of patents among sub-classes. These experiments try to mitigate this problem cconsidering different portions of initial data collection that present lower standard distribution and have contribute to better classification results. More specifically, we evaluated:

- A sub-collection where the rarest IPC codes have been removed
- A sub-collection where the rarest and the most frequent IPC codes have been removed

6

Technical details

Here we mention that technical details of the thesis.

6.1 Platforms and programming tools

This section presents the platforms and tools used for the implementation. For each platform and tool, we give a brief presentation and mention how it was used.

6.1.1 Programming language

Python 3.8

First and foremost, the programming language that was used for the implementations is the Python programming language. More specifically, the version of Python that was used is the 3.8 version. Python is an open-source language that is widely used in the industry and academia. Python offers several useful libraries for easier operations such as NumPy, Pandas, Sklearn and SciPy. It also has several deep learning frameworks that run on top of Python, such as Tensorflow and Keras.

Jupyer

Jupyter Notebook App[6] is a notebook editing and running python documents via a web browser. Jupyter is a tool where one can interactively work with code, while its cell structure enables one to run one cell at a time, enabling quick testing and prototyping

6.1.2 Packages

Anaconda programming package

[6] https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html

For the development of the programming pipeline, the Anaconda Python Distribution was used. Anaconda Distribution is an open-source distribution of Python and R programming languages for data science, machine learning, large-scale data processing, predictive analytics and other relevant purposes. It includes more than 1,500 packages for these purposes and enables the possibility to install, run, and upgrade data science and ML environments and libraries like Scikit-learn, TensorFlow, Keras, and Numpy.

Anaconda provides the Anaconda Navigator[7], a desktop graphical user interface that allows the launch of applications and management of packages, environments and libraries without using a command line interface. More specifically, we used the Jupyter Notebook application, which is available in the Anaconda Navigator.

6.1.3 Libraries

NumPy[8] is a Python library that provides support of multidimensional arrays and matrix data structures.

SciPy[9] is a Python library that uses the multidimensional arrays of NumPy to provide more utility functions for optimization, stats and signal processing.

Matplotlib[10] is a plotting library for creating static, animated, and interactive visualizations in Python.

Pandas[11] is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of Python. It offers matrix visualization matrix treatment.

Sklearn[12] is the python name for Scikit-learn. Scikit-learn is built on NumPy, SciPy and matplotlib. It is built as an efficient and simple tool for handling data, among them data pre-processing and analysis.

BeautifulSoup[13] is a Python library for extracting content out of HTML and XML files.

[7] https://anaconda.org/anaconda/anaconda-navigator

[8] https://numpy.org/

[9] https://www.scipy.org/

[10] https://matplotlib.org/

[11] https://pandas.pydata.org/pandas-docs/stable/index.html

[12] https://scikit-learn.org/stable/index.html

[13] https://www.crummy.com/software/BeautifulSoup/bs4/doc/

6.1.4 Machine learning frameworks

Keras[14] is a high-level neural networks API, written in Python, that supports multiple back-end neural network computation engines, such as TensorFlow, CNTK, Theano, MXNet, and PlaidML. It was developed with a focus on enabling fast experimentation with easy and fast prototyping. Being able to go from idea to result with the least possible delay.

TensorFlow[15] is an end-to-end open source platform for high-performance numerical computation and machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications. Its flexible architecture allows deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

[14] https://keras.io/

[15] https://www.tensorflow.org

7

Evaluation results and discussion

Here we present the results of the experiments we conducted in the form of tables and graphs to illustrate facts, trends and insights. We also give a detailed explanation and commentary on the results, always in relation to the problem that our techniques aspire to solve.

7.1 Deep learning models

7.1.1 Experiments with initial DL models

As mentioned in section 4.4, the machine learning algorithms that were selected to be evaluated are a CNN, a Bidirectional LSTM, a Bidirectional GRU, a LSTM and a GRU. Initially, we performed our experiments using a baseline version of these deep learning models.

Parameters: For these initial experiments, the first 60 words were retrieved from one data collection each time (i.e., the first 60 words from the Title-Abstract data collection which contain text only from the title and the abstract part of each patent, the first 60 words from the Description data collection which contain text only from the description part of each patent, the first 60 words from the Claims data collection which contain text only from the claims parts of each patent). Afterwards, the FastText pre-trained word embedding was used for the representation of these 60 feature words, while epochs were set equal to 3 and batch size to 128.

In Table 4 and Table 5, the accuracy, MRR, S@3, S@5 and S@10 metrics are displayed for different DL models and data collections.

Table 4: Accuracy, MRR, S@3, S@5 and S@10 scores for different DL models and data collections (Here, for the Title-Abstract data collection and the Description data collection).

Data collection	Title-Abstract					Description				
Metric/ML model	CNN	Bi-LSTM	Bi-GRU	LSTM	GRU	CNN	Bi-LSTM	Bi-GRU	LSTM	GRU
Accuracy (%)	51.12	54.50	55.28	51.42	51.21	51.07	52.61	54.27	50.27	50.99
MRR	0.65	0.68	0.68	0.65	0.64	0.65	0.66	0.67	0.64	0.64
S@3 (%)	74.48	77.05	77.93	73.72	73.74	74.18	75.62	77.26	72.93	73.65
S@5 (%)	82.03	84.22	85.05	81.21	81.10	81.99	83.16	84.47	80.38	81.36
S@10 (%)	89.14	90.89	91.29	88.86	88.58	89.32	90.11	91.20	88.17	88.62

Table 5: Accuracy, MRR, S@3, S@5 and S@10 scores for different DL models and data collections (Here, for the Claims data collection and all sections data collection).

Data collection	Claims					All sections				
Metric/ML model	CNN	Bi-LSTM	Bi-GRU	LSTM	GRU	CNN	Bi-LSTM	Bi-GRU	LSTM	GRU
Accuracy (%)	49.45	52.30	53.44	49.55	49.94	52.69	55.16	54.97	53.10	53.64
MRR	0.63	0.65	0.66	0.62	0.63	0.66	0.68	0.68	0.66	0.67
S@3 (%)	71.83	73.91	75.62	70.97	71.66	75.42	78.23	77.90	75.77	75.89
S@5 (%)	79.46	81.48	82.50	78.83	79.27	82.85	85.25	84.87	83.08	83.19
S@10 (%)	87.45	88.77	89.24	86.31	86.78	90.02	91.70	91.46	90.16	90.23

Figure 14 presents the accuracy scores achieved for different DL models and data collections. It is shown that LSTM and GRU models perform similar, while a significant improvement in accuracy is achieved by the usage of the bidirectional DL models (either GRU and LSTM), which demonstrate the best accuracy scores among all data collections. More specifically, the bidirectional GRU reaches the best accuracy score of 55.28% for the data collection of the title and abstract.

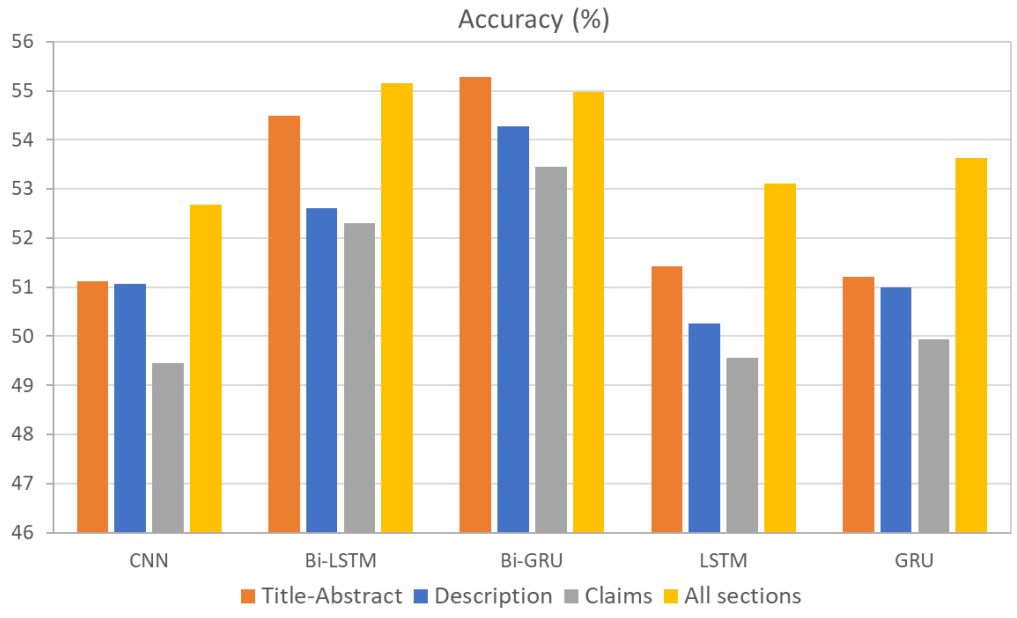


Figure 14: Graphical representation of accuracy scores using different DL models for patent classification. The DL models have been evaluated for the first 60 words coming either from i. the title-abstract section; ii. the description section, iii. the claims section or iv. the concatenation of all the above sections.

7.1.2 Variations in the structure and parameters of the DL models

The choice of machine learning algorithms is a key factor for improving the classification results. In this section, we perform several optimizations in the selected DL models and adopt improvements in the structure and parameters.

7.1.2.1 Variations of the CNN model

We tested different variations of the selected CNN model (Figure 10) by adding and removing optimization layers, including a Dropout layer (after the Dense layer), a Spatial Dropout layer (after the Embedding layer) and a Global Average Pooling layer (after the Convolutional 1D layer). Table 6 presents these variations and the evaluation scores achieved by each of them.

Table 6: Evaluation scores for different variations in the structure of the original CNN model.

Metrics/ CNN variations	Initial model	Add Dropout(0.5)	Add Dropout(0.5), SpatialDropout1D(0.1)	Add Dropout(0.5) & remove MaxPooling	Add Dropout(0.5) & remove MaxPooling, Flatten	Add Dropout(0.5), GlobalAveragePooling1D & remove MaxPooling, Flatten
Accuracy (%)	52.69	53.20	51.45	52.14	failed	49.74
MRR	0.66	0.66	0.65	0.65		0.63
S@3 (%)	75.42	75.94	74.08	74.87		72.22
S@5 (%)	82.85	83.10	81.80	82.47		80.45
S@10 (%)	90.02	89.98	89.23	89.51		88.59

Besides the CNN's structure, we tested different parameters of the Convolutional 1D layer. More specifically, we tested different number of filters {64, 100, 128, 150, 256, 512, 1024} and different kernel sizes {2, 3, 4}. Table 7 and Table 8 present these variations in the parameters and the evaluation scores achieved by each of them.

Table 7: Evaluation scores for different number of filters of the convolutional 1D layer of the original CNN model.

Metrics/ Parameters variations	Initial model - Conv1D(64, 2)	Conv1D(64,3)	Conv1D(100,3)	Conv1D(128, 3)	Conv1D(150,3)	Conv1D(256,3)	Conv1D(512,3)	Conv1D(1024,3)
Accuracy	52.69	52.78	53.86	54.02	54.32	55.21	55.76	55.99
MRR	0.66	0.66	0.67	0.67	0.67	0.68	0.68	0.69
P@3	75.42	75.14	76.29	76.29	76.76	77.48	77.81	78.26
P@5	82.85	82.66	83.68	83.56	83.95	84.46	84.57	85.00
P@10	90.02	89.86	90.48	90.31	90.59	90.91	91.01	91.22

Table 8: Evaluation scores for different kernel sizes of the convolutional 1D layer of the original CNN model.

Metrics/ Parameters variations	Initial model - Conv1D(64, 2)	Conv1D(100,2)	Conv1D(100,3)	Conv1D(100,4)
Accuracy (%)	52.69	53.53	53.86	53.34
MRR	0.66	0.67	0.67	0.67
S@3 (%)	75.42	76.10	76.29	76.14
S@5 (%)	82.85	83.74	83.68	83.40
S@10 (%)	90.02	90.46	90.48	90.36

After this exploration, we selected to insert a Dropout layer with a dropout rate of 0.5 in our initial CNN model and alter the parameters of the convolutional 1D layer to 256 filters and 3

kernels from initial 64 filters and 2 kernels respectively. Even though the accuracy score can be increased more when selecting the number of filters to be more than 64, we preferred to stay in this number due to the increased processing time experienced when we increased the number of filters (about 220 sec for 128 filters, 370 sec for 256 filters and 660 sec for 1,024 filters). Last, the epochs were re-set to 5, where the model seems to start converging.

7.1.2.2 Variations of the RNN models

For initial RNN models, including a Bi-LSTM, a Bi-GRU, a LSTM and a GRU (Figure 11), we made an exploration on optimal selection of the recurrent dropout and dropout parameters, which are internally implemented in these DL models. Table 9, Table 10, Table 11 and Table 12 present the different parameters used for each RNN DL model and the evaluation scores achieved by each variation.

Table 9: Evaluation scores for different recurrent dropout and dropout parameters of the original Bi-LSTM model.

Metrics/ Bi-LSTM variations	Initial model - Epoch=3, recurrent_dropout (0.1), dropout (0.1)	Epoch=15, recurrent_dropout (0.1), dropout (0.1)	Epoch=15, dropout (0.2)	Epoch=15, without recurrent_dropout & dropout
Accuracy (%)	54.50	60.01	59.19	59.03
MRR	0.68	0.72	0.72	0.71
S@3 (%)	77.05	81.95	81.72	81.13
S@5 (%)	84.22	88.39	88.06	87.09
S@10 (%)	90.89	93.41	93.31	92.70

Table 10: Evaluation scores for different recurrent dropout and dropout parameters of the original Bi-GRU model.

Metrics/ Bi-GRU variations	Initial model - Epoch=3, without recurrent_dropout & dropout	Epoch=15, without recurrent_dropout & dropout	Epoch=15, recurrent_dropout (0.1), dropout (0.1)	Epoch=15, dropout (0.2)
Accuracy (%)	55.28	59.03	46.45	59.72
MRR	0.68	0.71	0.60	0.72
S@3 (%)	77.93	80.60	68.65	82.07
S@5 (%)	85.05	87.35	76.61	88.40
S@10 (%)	91.29	92.89	84.67	93.47

Table 11: Evaluation scores for different recurrent dropout and dropout parameters of the original LSTM model.

Metrics/ LSTM variations	Initial model - Epoch=3, without recurrent_dropout & dropout	Epoch=15, without recurrent_dropout & dropout	Epoch=15, recurrent_dropout (0.1), dropout (0.1)	Epoch=15, dropout (0.2)
Accuracy (%)	51.42	58.87	58.93	58.41
MRR	0.65	0.71	0.71	0.71
S@3 (%)	73.72	80.72	80.89	80.43
S@5 (%)	81.21	87.12	87.58	87.01
S@10 (%)	88.86	92.58	93.03	92.82

Table 12: Evaluation scores for different recurrent dropout and dropout parameters of the original GRU model.

Metrics/ GRU variations	Initial model - Epoch=3, without recurrent_dropout & dropout	Epoch=15, without recurrent_dropout & dropout	Epoch=15, recurrent_dropout (0.1), dropout (0.1)	Epoch=15, dropout (0.2)
Accuracy (%)	51.21	58.80	14.15	58.18
MRR	0.64	0.71	0.24	0.71
S@3 (%)	73.74	81.00	26.12	80.68
S@5 (%)	81.10	87.41	33.44	87.12
S@10 (%)	88.58	92.77	44.70	92.86

After this exploration, we selected to keep for bidirectional-LSTM and LSTM models the parameters of recurrent dropout and dropout both equal to 0.1 and ignore these parameters for bidirectional-GRU and GRU models. Last, the epochs were re-set to 15, where the models seem to start converging.

7.1.3 Experiments with optimized DL models

The initial experiments were repeated for improved CNN and RNN models. Table 13 and Table 14 present the accuracy, MRR, S@3, S@5 and S@10 scores of improved DL models for all data collections.

Table 13: Accuracy, MRR, S@3, S@5 and S@10 scores with improved parameters for different DL models (Here, for the Title-Abstract data collection and the Description data collection).

Data collection	Title-Abstract					Description				
Metric/ML model	CNN	Bi-LSTM	Bi-GRU	LSTM	GRU	CNN	Bi-LSTM	Bi-GRU	LSTM	GRU
Accuracy (%)	54.66	59.83	59.43	59.26	58.71	53.50	59.40	59.24	58.28	58.10
MRR	0.67	0.72	0.72	0.71	0.71	0.67	0.72	0.71	0.71	0.71
S@3 (%)	76.89	81.58	81.08	80.74	80.30	76.10	81.39	80.76	80.43	80.04
S@5 (%)	83.79	87.89	87.37	87.11	86.82	83.33	87.58	87.35	86.82	86.77
S@10 (%)	90.24	93.18	92.84	92.70	92.52	89.82	93.14	92.73	92.62	92.44

Table 14: Accuracy, MRR, S@3, S@5 and S@10 scores with improved parameters for different DL models (Here, for the Claims data collection and all sections data collection).

Data collection	Claims					All sections				
Metric/ML model	CNN	Bi-LSTM	Bi-GRU	LSTM	GRU	CNN	Bi-LSTM	Bi-GRU	LSTM	GRU
Accuracy (%)	51.83	58.31	57.93	57.48	56.72	54.17	59.24	59.53	58.51	58.51
MRR	0.65	0.71	0.70	0.70	0.69	0.67	0.72	0.72	0.71	0.71
S@3 (%)	73.74	79.71	79.18	78.43	78.21	76.14	81.42	81.54	80.70	80.70
S@5 (%)	81.12	86.22	85.57	85.22	84.76	83.35	87.75	87.81	87.41	87.02
S@10 (%)	88.30	91.89	91.58	91.20	90.89	90.10	93.00	93.25	93.01	92.82

Moreover, Figure 15 presents the accuracy scores achieved for improved DL models for all data collections. The best accuracy score of 59.83% is achieved for the bidirectional-LSTM DL model for the data collection of title-abstract.

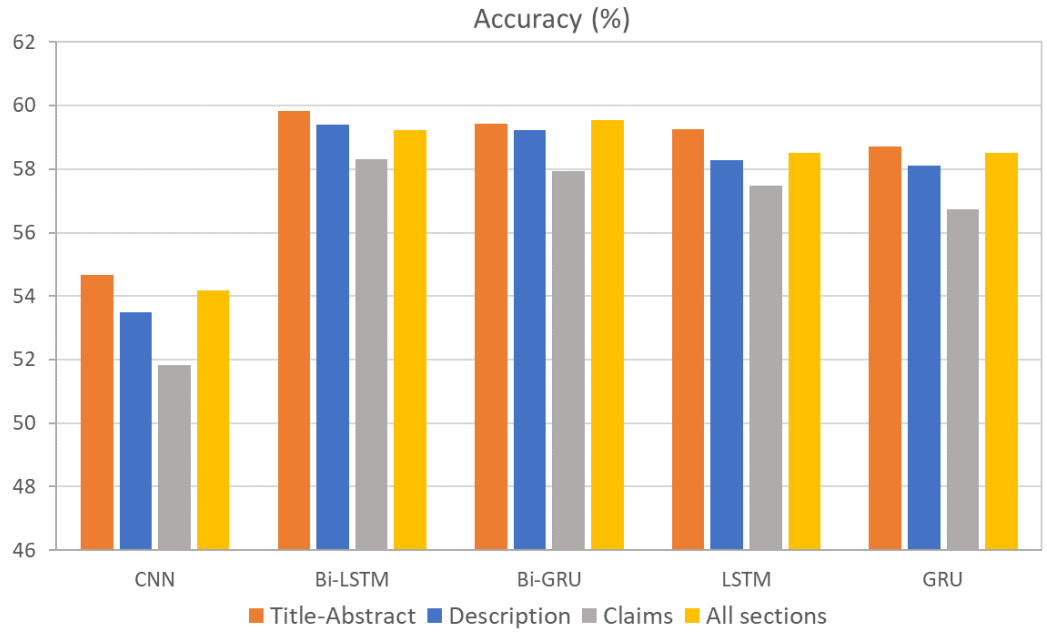


Figure 15: Graphical representation of accuracy scores using optimized DL models for patent classification. The improved DL models have been evaluated for the first 60 words coming either from i. the title-abstract section; ii. the description section, iii. the claims section or iv. the concatenation of all the above sections.

Comparing these results with the results achieved by the initial DL models, we detect that accuracy was improved for CNN (approximately) by 5%, for bidirectional LSTM by 10%, for bidirectional GRU by 8% by LSTM by 14% and for GRU by 13%. This is depicted in Table 15.

Table 15: Improvement in accuracy scores achieved by all DL models after the optimizations.

	CNN	Bi-LSTM	Bi-GRU	LSTM	GRU
Accuracy for initial DL models (averaged across all data collections)	51.08	53.64	54.48972	51.08445	51.44291
Accuracy for improved DL models (averaged across all data collections)	53.54	59.19	59.03	58.38	58.01
Improvement (%)	4.82	10.35	8.34	14.29	12.76

7.1.4 Ensemble method

In the current experimental setup, we investigated whether a simple ensemble method of averaging the outcome of three independent classifiers applied on the three "mono-thematic" data collections can achieve better classification results than each of those classifiers acting on its own. the first 60 words were retrieved for all data collections

Parameters: For these experiments, the first 60 words were retrieved from one data collection each time. Afterwards, the FastText pre-trained word embedding was used for the

representation of these 60 feature words. When CNN models served as individual classifiers, we used the improved structure and parameters of the CNN model (with the exception that the filter parameter of the convolutional 1D layer was set equal to 64 instead of 256 for reducing the processing time). On the other hand, when RNN models were used as individual classifiers, we used the improved parameters of the RNN models. Last, epochs were set equal to 3 and batch size to 128.

Table 16, Table 17 and Table 18 present the evaluation metrics of each individual classifier trained on the 60 words retrieved from the title-abstract, the description and the claims data collection, respectively. More specifically, the first classifier classifies patents based on text coming from the Title-Abstract part, the second classifies patents based on text coming from the Description part and the third classifies patents based on text coming from the Claims part.

Table 16: Accuracy, MRR, S@3, S@5 and S@10 scores achieved by individual classifiers using text only from the Title-Abstract part.

Data collection	Title-Abstract				
Metric/ML model	CNN	Bi-LSTM	Bi-GRU	LSTM	GRU
Accuracy (%)	53.65	59.83	59.43	59.26	58.71
MRR	0.67	0.72	0.72	0.71	0.71
S@3 (%)	75.84	81.58	81.08	80.74	80.30
S@5 (%)	83.28	87.89	87.37	87.11	86.82
S@10 (%)	90.04	93.18	92.84	92.70	92.52

Table 17: Accuracy, MRR, S@3, S@5 and S@10 scores achieved by individual classifiers using text only from the Description part.

Data collection	Description				
Metric/ML model	CNN	Bi-LSTM	Bi-GRU	LSTM	GRU
Accuracy (%)	52.99	59.40	59.24	58.28	58.10
MRR	0.66	0.72	0.71	0.71	0.71
S@3 (%)	75.27	81.39	80.76	80.43	80.04
S@5 (%)	82.48	87.58	87.35	86.82	86.77
S@10 (%)	89.48	93.14	92.73	92.62	92.44

Table 18: Accuracy, MRR, S@3, S@5 and S@10 scores achieved by individual classifiers using text only from the Claims part.

Data collection	Claims				
Metric/ML model	CNN	Bi-LSTM	Bi-GRU	LSTM	GRU
Accuracy (%)	51.54	58.31	57.93	57.48	56.72
MRR	0.64	0.71	0.70	0.70	0.69
S@3 (%)	73.17	79.71	79.18	78.43	78.21
S@5 (%)	80.59	86.22	85.57	85.22	84.76
S@10 (%)	87.92	91.89	91.58	91.20	90.89

Table 19 shows the evaluation scores of the ensemble method combining the results of the above three individual classifiers working on separate parts of the patent text.

Table 19: Accuracy, MRR, S@3, S@5 and S@10 scores achieved by combining the results of three individual classifiers working on separate parts of the patent text.

	Ensemble with 3 classifiers				
Metric/ML model	**CNN**	**Bi-LSTM**	**Bi-GRU**	**LSTM**	**GRU**
Accuracy (%)	59.54	64.57	64.85	63.51	63.44
MRR	0.72	0.76	0.76	0.75	0.75
S@3 (%)	81.20	85.88	85.67	84.82	84.75
S@5 (%)	87.76	91.35	91.16	90.54	90.52
S@10 (%)	93.30	95.65	95.53	95.19	95.14

Figure 16 presents the accuracy scores for each individual classifier and for its combination. We observe that all evaluation scores are much improved when we apply the ensemble method compared with individual classifiers.

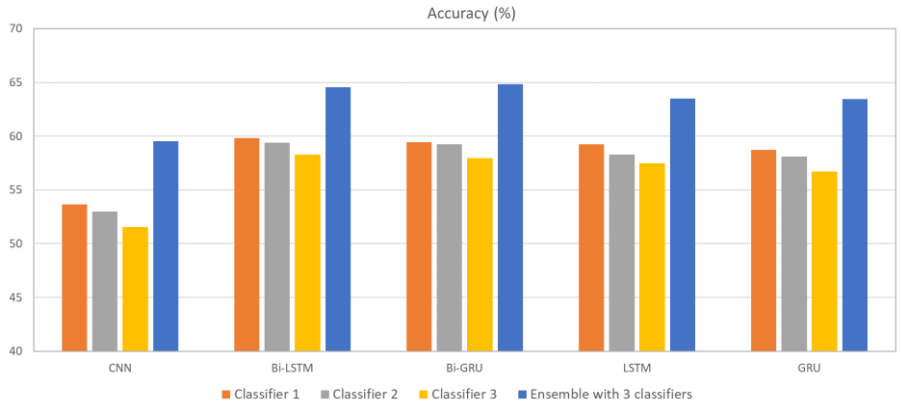


Figure 16: Accuracy scores using ensemble methods of different classifiers.

In order to quantify this improvement, we present in Table 20 the average accuracy achieved by all individual classifiers compared with the accuracy achieved by the ensemble method. The highest increase was observed when we applied the ensemble method with CNN individual classifiers (13%), followed by the ensemble method with bidirectional GRU individual classifiers (10%). Moreover, the combination of bidirectional GRU individual classifiers achieved an accuracy of 65%, which is the best accuracy score among all the experiments conducted within this thesis.

Table 20: Improvement in accuracy scores achieved by ensemble method.

	CNN	**Bi-LSTM**	**Bi-GRU**	**LSTM**	**GRU**
Accuracy for individual classifiers (averaged across all data collections)	52.73	59.18	58.87	58.34	57.84
Accuracy for ensemble method	59.54	64.57	64.85	63.51	63.44
Improvement (%)	12.92	9.11	10.16	8.87	9.69

7.2 Feature selection for patent representation

7.2.1 First "X" words of a patent part

In the current experimental setup, we extracted the first "X" words from each data collections. All sections data collection contains the concatenated result of the title-abstract, the description, and the claims sections, while each of the three "mono-thematic" data collections contain the title and abstract, the description and the claims sections of each patent, respectively. More specifically, we used different number of words to evaluate the patent representation, including: {20, 40, 60, 80, 100, 200, 300, 400}.

Parameters: For the experiments, the FastText pre-trained word embedding was used for the representation of the retrieved words. Epochs were set equal to 3 and batch size to 128. The experiments have been conducted using both the initial and improved version of the CNN model.

Initial CNN model

Table 21 and Table 22 present the accuracy, MRR, S@3, S@5 and S@10 metrics for each data collections when the number of words varies from 20 to 400.

Table 21: Accuracy, MRR, S@3, S@5 and S@10 scores using the initial CNN model for varying number of words retrieved each time from a different data collection (Here, from the Title-Abstract data collection and the Description data collection).

Data collection	Title-Abstract								Description							
Metrics/words	20	40	60	80	100	200	300	400	20	40	60	80	100	200	300	400
Accuracy (%)	52.31	52.43	51.12	51.22	50.61	47.70	47.06	47.71	47.09	50.18	51.07	51.77	51.66	50.46	49.00	47.56
MRR	0.65	0.66	0.65	0.65	0.64	0.61	0.61	0.61	0.60	0.64	0.65	0.65	0.65	0.64	0.63	0.62
S@3 (%)	74.33	75.23	74.48	74.25	73.78	70.31	70.31	70.65	68.65	72.75	74.18	75.19	75.23	73.83	73.18	71.67
S@5 (%)	81.53	82.72	82.03	81.80	81.36	78.86	78.60	78.77	76.46	80.63	81.99	82.89	83.05	82.06	81.36	80.38
S@10 (%)	88.66	89.81	89.14	88.98	88.98	87.11	86.97	87.17	84.35	88.03	89.32	90.25	90.12	89.94	89.22	88.69

Table 22: Accuracy, MRR, S@3, S@5 and S@10 scores using the initial CNN model for varying number of words retrieved each time from a different data collection (Here, from the Claims data collection and all sections data collection).

Data collection	Claims								All sections							
Metrics/words	20	40	60	80	100	200	300	400	20	40	60	80	100	200	300	400
Accuracy (%)	50.22	51.07	49.45	48.50	48.75	45.89	44.17	42.84	52.41	52.74	52.69	52.05	52.21	50.07	49.70	48.90
MRR	0.63	0.64	0.63	0.62	0.62	0.59	0.58	0.57	0.65	0.66	0.66	0.65	0.66	0.64	0.64	0.63
S@3 (%)	71.54	72.88	71.83	70.83	70.87	68.31	66.85	65.05	74.59	75.74	75.42	75.04	75.31	74.03	73.62	72.88
S@5 (%)	78.88	80.50	79.46	79.03	78.84	76.58	75.50	73.96	81.78	83.27	82.85	82.63	82.94	82.13	81.76	81.44
S@10 (%)	86.63	87.93	87.45	86.81	86.98	84.94	84.55	83.10	88.87	90.20	90.02	89.84	90.08	89.51	89.62	89.51

Figure 17 illustrates the accuracy scores for varying numbers of retrieved words from all four data collections, while the classification task has been performed with the initial CNN architecture. We observe that there is an increase in the accuracy score for smaller numbers of

retrieved words, while the accuracy decreases as the number of retrieved words exceeds 100 words for all data collections.

Moreover, the accuracy seems to be better when retrieving words from the data collection of all (concatenated) sections. The scores for all sections and title-abstract section are initially quite similar, which is reasonable considering that the number of words is low and most or even all of the words that are retrieved in both cases are totally coming from the title-abstract section. For bigger number of words, we observe that the accuracy scores for all sections and description section become almost equal, this trend can be detected up to 300 words. This can be interpreted that the first words used in the description section are quite important for the patent representation.

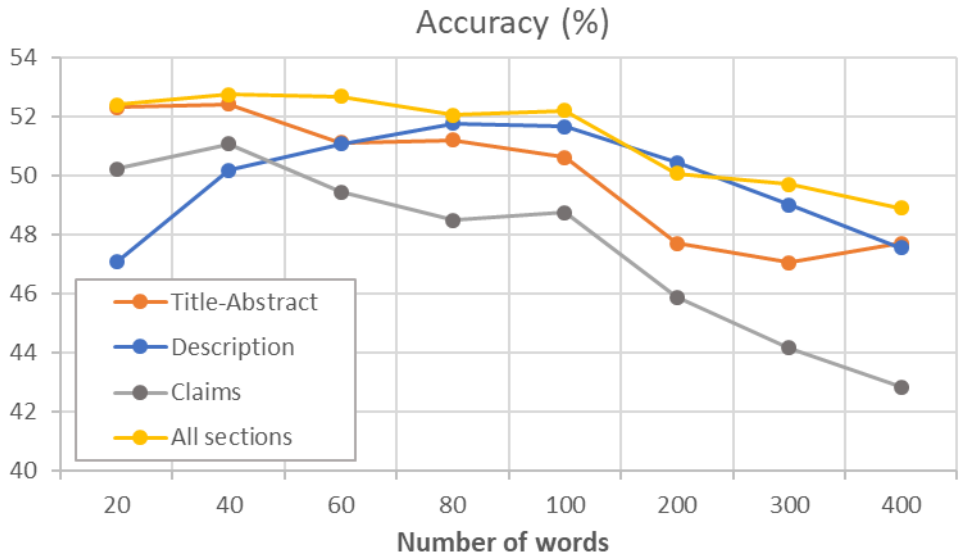


Figure 17: Graphical representation of accuracy scores achieved for varying number of words retrieved each time from a different data collection.

Figure 18 a, b, c, d shows the MRR, S@3, S@5 and S@10 scores for varying numbers of retrieved words from all four data collections. The most interesting observation resulting from the above figures is that the success at n (recall) increases significantly when we are looking at the top three, five and ten returned targets. Specifically, the success at the top three returned targets reaches 75.74% (optimal S@3) when we used the first 40 words for representing the patent document from all sections data collection, while for the same number of retrieved words and the same data collection, the success at the top five returned targets becomes 83.27% (optimal S@5). Last, the classification model will predict the correct label among the top ten returned with an accuracy of 90.25% (optimal S@10) if it's fed with the first 80 words retrieved from the data collection preserving the description sections.

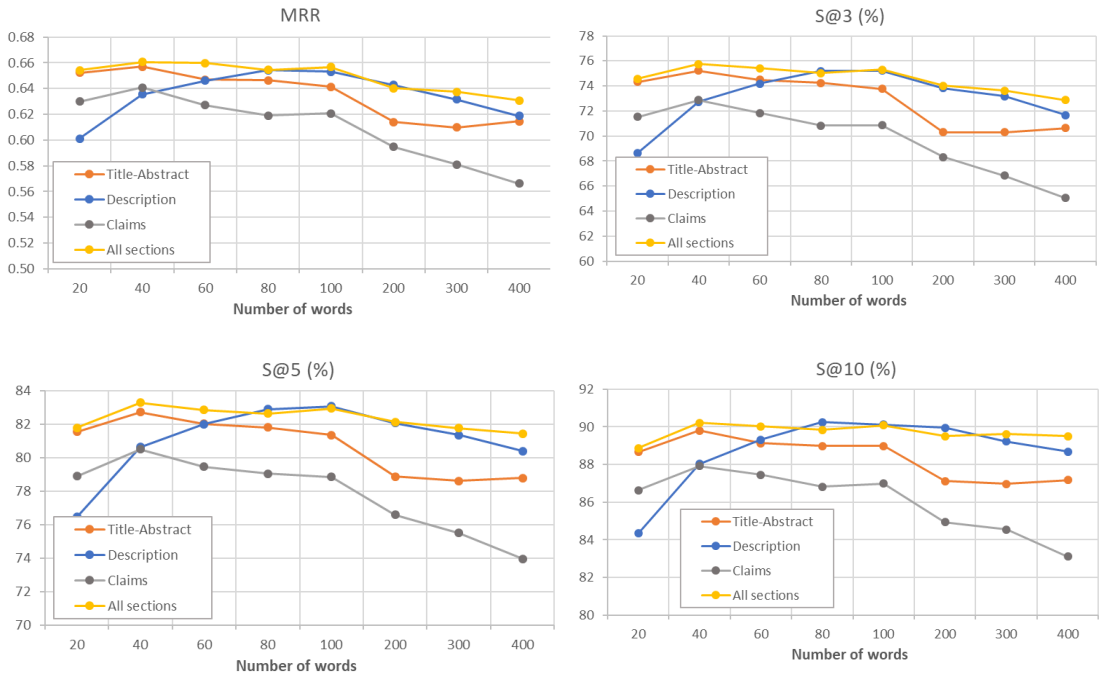


Figure 18a, b, c, d: MRR, S@3, S@5 and S@10 scores for varying number of words retrieved each time from a different data collection.

Improved CNN model

Table 23 and Table 24 presents the accuracy, MRR, S@3, S@5 and S@10 while Figure 19 illustrates the accuracy scores using the improved CNN model for each data collection when the number of words varies from 20 to 400.

Table 23: Accuracy, MRR, S@3, S@5 and S@10 scores using the optimized CNN model for varying number of words retrieved each time from a different data collection (Here, from the Title-Abstract data collection and the Description data collection).

Data collection	Title-Abstract								Description							
Metrics/words	20	40	60	80	100	200	300	400	20	40	60	80	100	200	300	400
Accuracy (%)	53.83	54.77	54.66	53.60	52.40	48.63	50.07	51.63	49.11	52.44	53.50	54.28	53.86	51.65	52.61	52.74
MRR	0.67	0.68	0.67	0.67	0.66	0.62	0.63	0.65	0.62	0.66	0.67	0.67	0.67	0.65	0.66	0.66
S@3 (%)	75.70	77.00	76.89	76.08	74.92	71.11	72.37	74.28	70.63	74.91	76.10	77.16	76.27	74.83	75.73	76.31
S@5 (%)	82.59	84.05	83.79	83.22	82.43	79.30	80.36	81.69	78.03	82.17	83.33	84.25	83.82	82.84	83.69	84.18
S@10 (%)	89.38	90.59	90.24	89.93	89.43	87.19	87.89	89.05	85.46	89.29	89.82	90.78	90.62	90.15	90.99	91.12

Table 24: Accuracy, MRR, S@3, S@5 and S@10 scores using the optimized CNN model for varying number of words retrieved each time from a different data collection (Here, from the Claims data collection and all sections data collection).

Data collection	Claims								All sections							
Metrics/words	20	40	60	80	100	200	300	400	20	40	60	80	100	200	300	400
Accuracy (%)	52.63	53.11	51.83	51.42	50.26	49.53	47.58	48.14	54.07	54.81	54.17	54.43	53.69	52.51	53.17	54.38
MRR	0.65	0.66	0.65	0.64	0.63	0.63	0.61	0.61	0.67	0.68	0.67	0.67	0.67	0.66	0.67	0.68
S@3 (%)	73.89	74.67	73.74	73.25	72.18	71.66	69.46	70.18	75.89	77.07	76.14	76.53	76.43	76.02	76.18	77.57
S@5 (%)	80.92	81.88	81.12	80.94	79.90	79.33	77.33	77.85	82.75	84.19	83.35	83.80	83.86	83.61	83.74	85.09
S@10 (%)	87.74	88.83	88.30	87.92	87.94	86.91	85.59	85.95	89.51	90.73	90.10	90.11	90.47	90.57	90.72	91.73

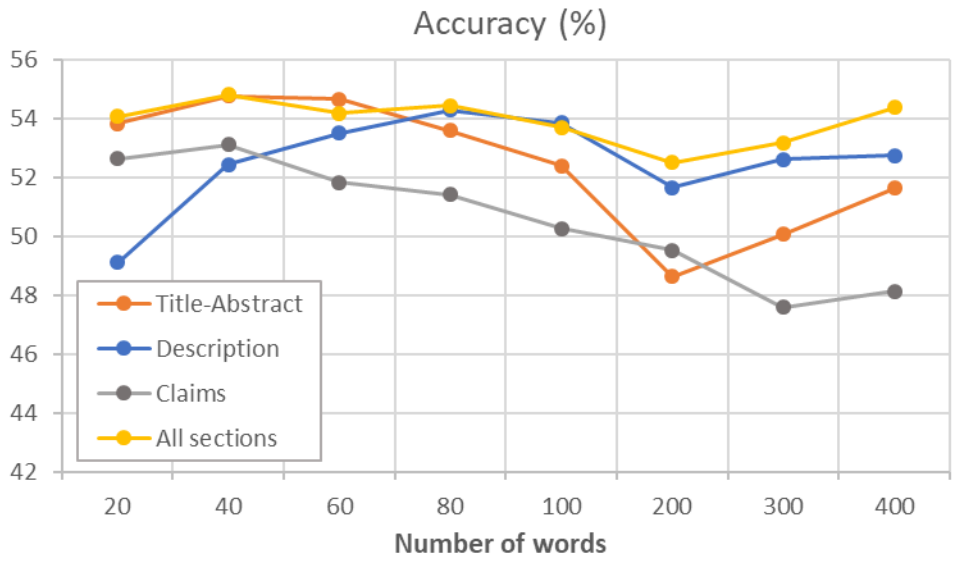


Figure 19: Graphical representation of accuracy scores achieved using the improved CNN model for varying number of words retrieved each time from a different data collection.

Moreover, Figure 20 a, b, c, d shows the MRR, S@3, S@5 and S@10 scores for varying numbers of retrieved words from all four data collections for the improved CNN model.

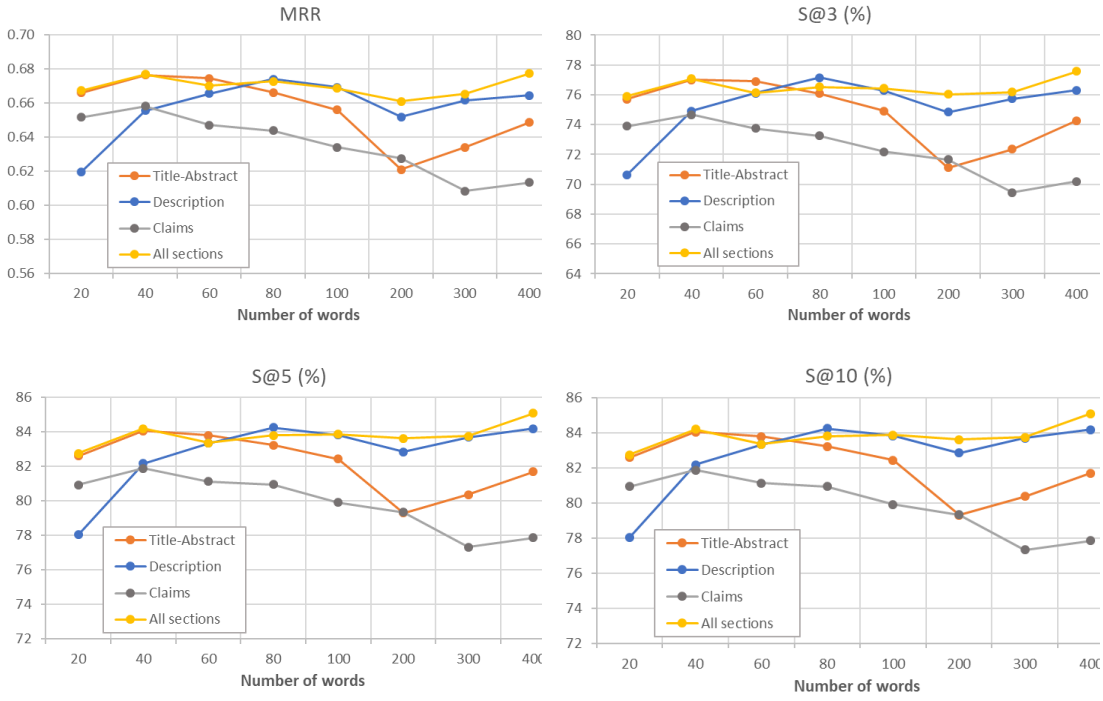


Figure 20 a, b, c, d: MRR, S@3, S@5 and S@10 scores using the improved CNN model for varying number of words retrieved each time from a different data collection.

Although all evaluation scores are improved now with the use of the optimized CNN model, we observe that similar trends with the initial CNN model are detected. The accuracy is better when retrieving words from the data collection of all (concatenated) sections, which is quite similar with i) the accuracy achieved when retrieving words from the title-abstract section for

small numbers of retrieved words and ii) the accuracy achieved when retrieving words from the description section for larger numbers of retrieved words.

In general, we observe that there is an increase in the accuracy score for smaller numbers of retrieved words, while as the number of words increases the performance of patent classification decreases. This mainly happens due to over-fitting conditions experienced in the DL model. In this case, a better optimization of the machine learning algorithm is required to overcome the over-fitting of the model.

Moreover, the success at the top three returned targets reaches 77.57% (optimal S@3) when we used the first 400 words for representing the patent document from all sections data collection, while for the same number of retrieved words and the same data collection, the success at the top five returned targets becomes 85.09% (optimal S@5) and the success at the top five returned targets becomes 91.73% (optimal S@10).

7.2.2 First "Y" words from all patent parts

In these experiments, we used the first "Y" words of each part creating a concatenation of words coming from the title-abstract, the description and the claims part at the same time for the representation of the patent document. The total number of words retrieved from all sections should account (approximately) to the following: {20, 40, 60, 80, 100, 200, 300, 400}.

Parameters: For the experiments, we used the FastText pre-trained word embedding for the text representation and the improved CNN model for patent classification.

Table 25 presents the accuracy, MRR, S@3, S@5 and S@10 metrics when the number of retrieved words for each part varies from 7 to 133. Epochs were set equal to 5 and batch size to 128.

Table 25: Accuracy, MRR, S@3, S@5 and S@10 scores for varying number of words retrieved in parallel from all patent parts.

Total words	21	39	60	81	99	201	300	399
Metrics/words per part	7	13	20	27	33	67	100	133
Accuracy (%)	48.39	54.43	55.66	56.43	56.61	56.80	55.81	55.05
MRR	0.61	0.67	0.68	0.69	0.69	0.70	0.69	0.68
S@3 (%)	68.85	75.78	77.54	78.19	79.15	79.09	78.49	78.15
S@5 (%)	76.32	82.72	84.51	84.74	85.93	86.02	85.55	85.53
S@10 (%)	84.18	89.36	90.81	90.98	91.70	92.24	92.03	91.97

Figure 21 illustrates the accuracy scores achieved by the previous experiment (section 7.2.1) where we retrieve words by each data collection separately, compared to the accuracy achieved in the current experiment where we retrieve words from all patent parts simultaneously. We

notice that the accuracy for the current experiment is slightly better, and therefore this representation might be more suitable.

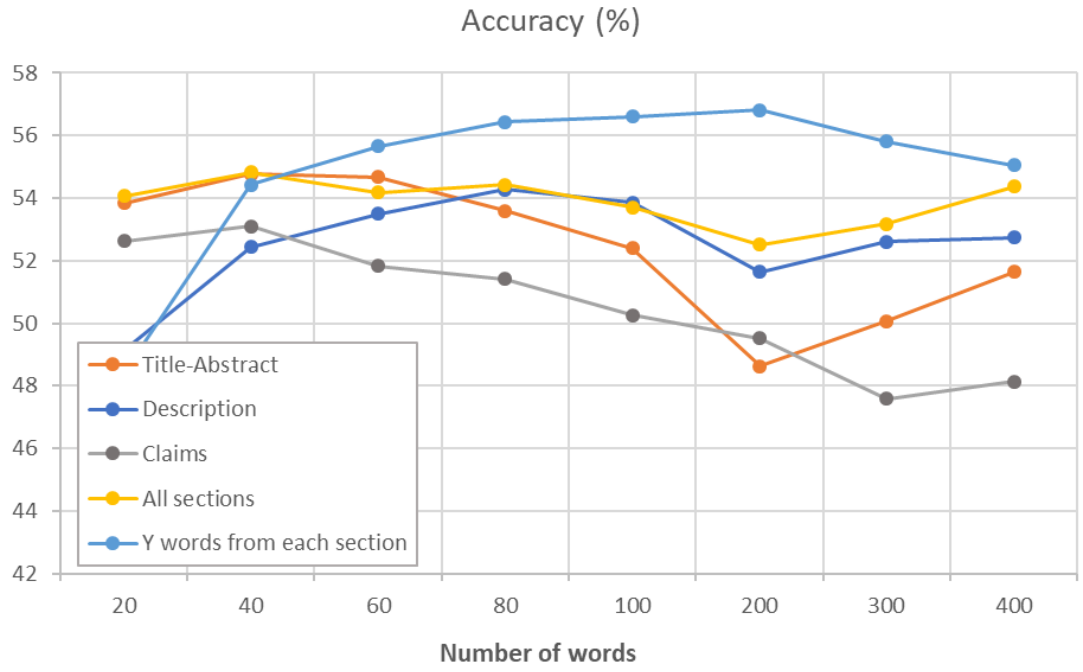


Figure 21: Graphical representation of accuracy scores achieved for a varying number of words retrieved each time from a different data collection (same as Figure 19) in comparison with the accuracy score achieved for a varying number of words retrieved in parallel from all patent parts.

7.2.3 First "X" significant words of a patent part

In these experiments, we explored different ways for selecting the most significant words for each data collection enforcing different processing steps. More specifically, we retrieve the first "X" words from the following data collections which contain: i) the stop words removed patent text, ii) the stemmed and stop words removed patent text, iii) the tf-idf sorted patent text, iv) the tf-idf sorted patent text from which the stop words have been removed, and v) the tf-idf sorted patent text from which the stop words have been removed and stemmed has been enforced.

Parameters: For the experiments, the first 60 words were retrieved from each data collection. Moreover, we used the FastText pre-trained word embedding for the text representation and the improved CNN model for the patent classification (with the exception that the filter parameter of the convolutional 1D layer was set equal to 64 instead of 256 for reducing the processing time). Epochs were set equal to 5 and batch size to 128.

Limitations: In this experimental setup, we encountered memory issues when we tried to create the tf-idf vectors for the claims and all sections data collections. Because of that, we skipped

the experiments on the claims data collection, while for all sections data collection we calculated the tf-idf scores using the scikit-learn's methods.

Table 26 and Table 27 present the accuracy, MRR, S@3, S@5 and S@10 metrics for different ways of selecting the most significant words for the title-abstract, the description and all sections data collections.

Table 26: Accuracy, MRR, S@3, S@5 and S@10 scores for different ways of processing and selecting the feature words from the Abstract-Title data collection.

Data collection	Title-Abstract					
	Initial dataset	Stopwords	Stopwords and stemming	tf-idf	tf-idf stop	tf-idf stop and stem
Accuracy (%)	54.66	48.58	43.30	41.52	48.12	43.35
MRR	0.67	0.62	0.57	0.55	0.62	0.57
S@3 (%)	76.89	71.13	65.66	63.60	71.58	65.81
S@5 (%)	83.79	79.00	74.22	72.36	79.86	74.47
S@10 (%)	90.24	87.09	82.82	81.77	87.82	83.65

Table 27: Accuracy, MRR, S@3, S@5 and S@10 scores for different ways of processing and selecting the feature words from the Description data collection.

Data collection	Description					
	Initial dataset	Stopwords	Stopwords and stemming	tf-idf	tf-idf stop	tf-idf stop and stem
Accuracy (%)	53.50	42.89	39.11	13.49	36.93	32.53
MRR	0.67	0.57	0.53	0.24	0.51	0.47
S@3 (%)	76.10	66.17	61.61	26.16	59.46	54.09
S@5 (%)	83.33	74.67	70.32	33.81	68.93	63.72
S@10 (%)	89.82	83.90	80.11	45.58	79.50	75.08

Table 28: Accuracy, MRR, S@3, S@5 and S@10 scores for different ways of processing and selecting the feature words from all sections data collection.

Data collection	All Sections (SKLearn implementation)					
	Initial dataset	Stopwords	Stopwords and stemming	tf-idf	tf-idf stop	tf-idf stop and stem
Accuracy (%)	54.17	42.61	38.18	15.67	32.43	27.17
MRR	0.67	0.56	0.52	0.27	0.47	0.40
S@3 (%)	76.14	64.70	59.73	30.13	53.82	46.52
S@5 (%)	83.35	72.91	68.50	38.51	63.78	55.83
S@10 (%)	90.10	81.95	78.20	50.54	74.72	67.57

Figure 22 presents the accuracy scores achieved using different ways of selecting the feature words for the representation of the patent document coming from all data collections. We observe that the accuracy score is decreasing for all different processing steps we enforced to retrieve the most significant words to represent the patent document. The most significant decrease has been experienced when selecting the first 60 words after sorting words based on their tf-idf score. An increase in the accuracy is achieved when selecting the first 60 words after having removed the stop words and sorted words based on their tf-idf score (but the accuracy is still worse than the accuracy achieved in initial version of the data collection).

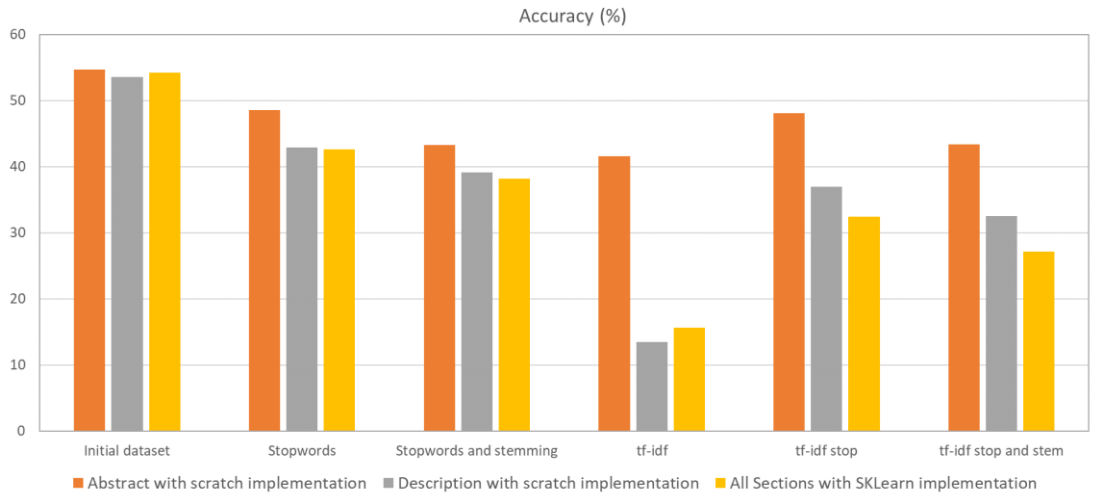


Figure 22: Graphical representation of accuracy scores using different ways of selecting the feature words for the representation of the patent document coming either from i. the title-abstract section; ii. the description section, iii. the claims section or iv. the concatenation of all the above sections.

It is important to mention that additional exploration is required to investigate: i) the representations of significant words coming from the claims and all sections, while using the manual crafted algorithm for the calculation of the tf-idf score, and ii) the retrieval of significant words from all patent parts at the same time.

7.3 Language models for patent representation

In the current experimental setup, we explored how different word embeddings can affect the patent classification outcome. We tested the pre-trained language models of FastText, Word2Vec, and Glove, as well as the Word2Vec trained in our patent-domain corpus.

Parameters: For the experiments, the first 60 words were retrieved from each data collection every time. Moreover, the initial version of the CNN model was used for patent classification, epochs were set equal to 3 and batch size to 128.

In Table 29 and Table 30, the accuracy, MRR, S@3, S@5 and S@10 scores are displayed for different language models and data collections.

Table 29: Accuracy, MRR, S@3, S@5 and S@10 scores for different language models and data collections (Here, for the Title-Abstract data collection and the Description data collection).

Data collection	Abstract-Title				Description			
Metrics/Language model	FastText	Word2Vec	Glove	Domain-trained Word2Vec	FastText	Word2Vec	Glove	Domain-trained Word2Vec
Accuracy (%)	51.12	50.06	48.61	50.87	51.07	49.46	47.31	49.31
MRR	0.65	0.64	0.62	0.64	0.65	0.63	0.61	0.63
S@3 (%)	74.48	73.02	70.93	73.50	74.18	72.49	70.18	72.75
S@5 (%)	82.03	80.91	78.90	81.18	81.99	80.63	78.69	80.79
S@10 (%)	89.14	88.63	86.95	88.73	89.32	88.54	86.94	88.25

Table 30: Accuracy, MRR, S@3, S@5 and S@10 scores for different language models and data collections (Here, for the Claims data collection and all sections data collection).

Data collection	Claims				All sections			
Metrics/Language model	FastText	Word2Vec	Glove	Domain-trained Word2Vec	FastText	Word2Vec	Glove	Domain-trained Word2Vec
Accuracy (%)	49.45	48.45	46.09	48.29	52.69	50.46	48.68	52.01
MRR	0.63	0.61	0.60	0.61	0.66	0.64	0.62	0.65
S@3 (%)	71.83	69.91	68.26	69.89	75.42	73.37	71.49	74.81
S@5 (%)	79.46	77.94	76.34	77.88	82.85	81.15	79.69	82.16
S@10 (%)	87.45	86.01	84.63	86.10	90.02	88.66	87.28	89.57

In Figure 23, the accuracy scores for different language models and data collections are displayed. Among pre-trained language models, the FastText seems to achieve better representation of the patent text and thus better accuracy scores than other two pre-trained language models (i.e., Word2Vec and Glove). For the FastText, the accuracy reaches 52.69% for the dataset of all sections. It is also interesting that even when the Word2Vec language model has been trained on the domain-specific corpus and then used for the representation of the patent text, it reaches worse results (52.01%) than FastText.

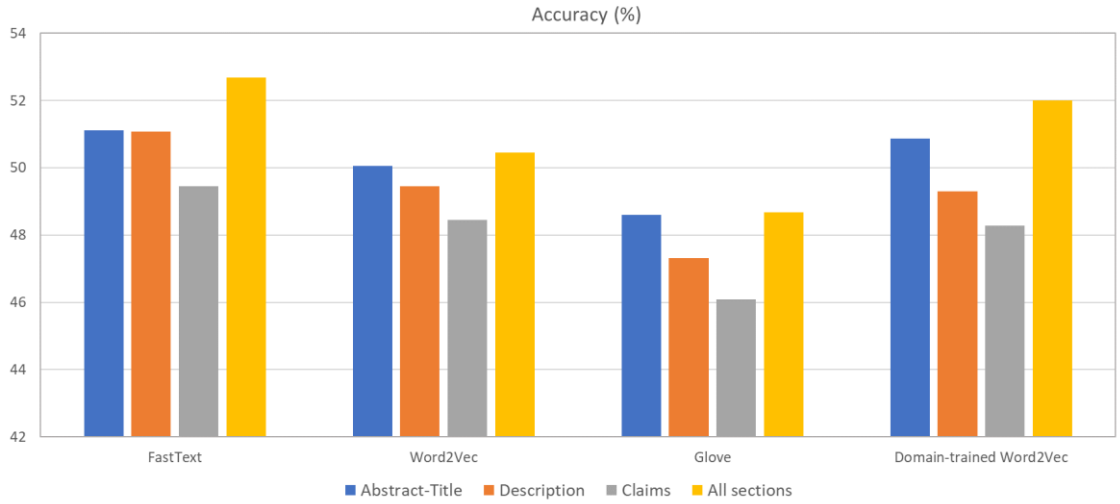


Figure 23: Graphical representation of accuracy scores using different language models for representing the patent document. The embeddings has been created considering the first 60 words coming either from i. the title-abstract section; ii. the description section, iii. the claims sections and iv. the concatenation of all the above sections.

However, this experimental set needs further exploration, using: i) using different dimensions of the pre-trained language models, ii) different parameters for training the Word2Vec on the domain-specific corpus, and iii) different language models trained on the domain-specific corpus.

7.4 Sub-collection removing the most rare/frequent IPC codes

Checking more carefully the data collection and the distribution of IPC codes, we noticed that there are IPC codes with a very high frequency (e.g., the IPC code A61K has been assigned as the main classification code to 15,525 patent documents), while at the same time there are IPC codes with a very low frequency (e.g., the IPC code M56M appears only in one patent document, while there are 39 IPC codes in total that similarly appear in only one patent document). This creates significant unbalances in the machine learning process which is better taught to recognize the most often IPC codes. Figure 24 displays the distribution of the IPC codes' frequency (logarithmic value) which follows the normal distribution.

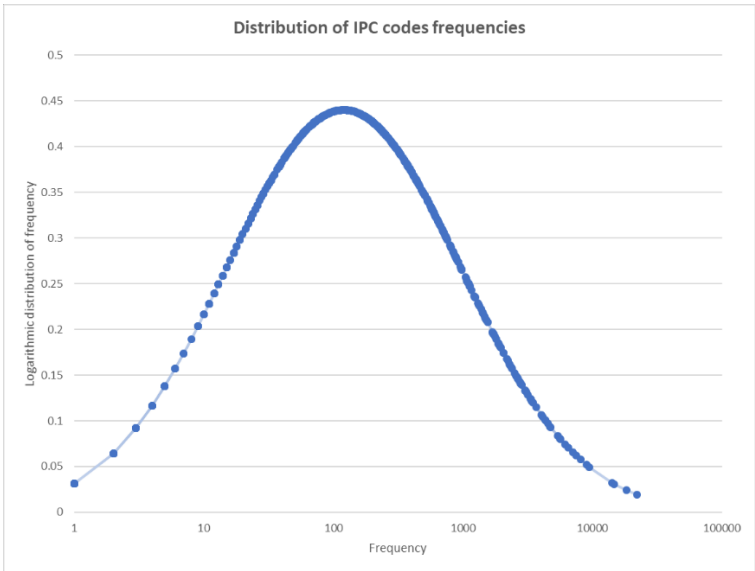


Figure 24: Logarithmic distribution of IPC codes' frequency across patents.

We decided to set some cutoffs and remove either the rarest (0.5σ) or the rarest and the most frequent (1σ) IPC codes. By this way, we also filter out the respective patent documents where these IPC codes appear creating two trimmed versions of the original data collection.

Thereafter, we repeated the experimental setup of using different number of words and data collections to evaluate whether this approach of excluding IPC codes resulted in better patent classification outcomes.

Parameters: For these experiments, the first 60 words from one data collection each time. Moreover, we used the FastText pre-trained word embedding for the text representation and the improved CNN model for the patent classification (with the exception that the filters parameter of the convolutional 1D layer was set equal to 64 instead of 256 for reducing the processing time). Epochs were set equal to 5 and batch size to 128.

Table 31 and Table 32 presents the accuracy, MRR, S@3, S@5 and S@10 metrics for all data collections when removing either the rarest or the rarest and the most frequent IPC codes.

Table 31: Accuracy, MRR, S@3, S@5 and S@10 scores when removing the patent documents either with the rarest or with the rarest and the most frequent IPC codes and using words from the Abstract-Title data collection and the Description data collection.

Data collection	Title-Abstract			Description		
Metrics/Words	Initial data collection	Removing the rarest IPC codes	Removing the rarest and most frequent IPC codes	Initial data collection	Removing the rarest IPC codes	Removing the rarest and most frequent IPC codes
Accuracy (%)	53.16	53.18	49.72	52.40	52.47	50.58
MRR	0.66	0.66	0.62	0.66	0.66	0.63
S@3 (%)	75.59	76.01	70.06	75.15	75.74	71.83
S@5 (%)	83.09	83.41	77.58	82.69	83.28	78.80
S@10 (%)	89.93	90.37	85.68	89.89	90.40	86.49

Table 32: Accuracy, MRR, S@3, S@5 and S@10 scores when removing the patent documents either with the rarest or with the rarest and the most frequent IPC codes and using words from the Claims data collection and all section data collection.

Claims			All sections		
Initial data collection	Removing the rarest IPC codes	Removing the rarest and most frequent IPC codes	Initial data collection	Removing the rarest IPC codes	Removing the rarest and most frequent IPC codes
51.34	52.03	45.94	53.17	53.84	49.95
0.64	0.65	0.58	0.66	0.67	0.63
72.85	74.12	65.95	75.24	76.16	70.97
80.35	81.55	73.12	82.71	83.94	78.41
87.74	88.84	81.80	89.70	90.62	86.39

Figure 25 presents the accuracy scores achieved for all data collections, when they have been trimmed out to keep only the rarest or the rarest and the most frequent IPC codes. We observe that when trimming out patents assigned the rarest IPC codes, the accuracy scores for all data collections are slightly improved. However, in order to draw reliable conclusions additional experiments are required to evaluate an extensive list of cutoffs.

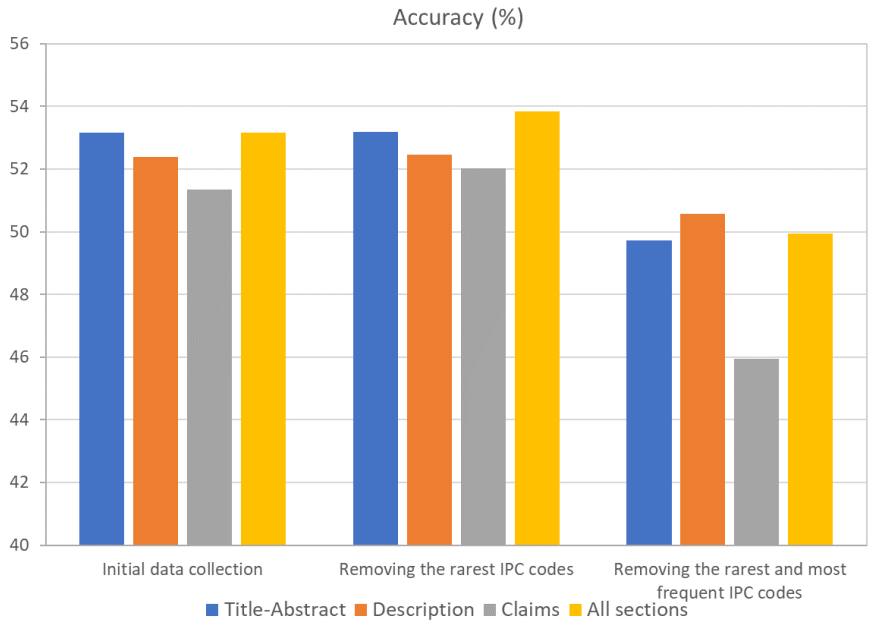


Figure 25: Graphical representation of accuracy scores for different sub-collections of the original data collection, reduced based on the rarest or the rarest and the most frequent IPC codes.

7.5 Evaluation summary

The performance of automated patent classification algorithms depends on many factors such as the choice of machine learning algorithms, the choice of feature words and language models to represent the patent document and the choice of the data collection.

The evaluation of different DL methods showed that a bidirectional-LSTM or GRU implementation can achieve better results compared with other DL methods. More specifically, a bidirectional-LSTM reaches the best accuracy score of 59.83% among (improved versions of) DL models for the data collection of the title and abstract. Moreover, an ensemble method combining the results of individual classifiers achieves improved results for all selected classifiers. More specifically, the combination of bidirectional GRU individual classifiers achieves an accuracy of 65%, which is the highest score achieved among all the experiments conducted within this thesis.

The sections from which we retrieve the representation words of a patent also seem to play an important role for automated patent classification. Although the abstract section offers the most valuable words for automated patent classification, the description section contains also important information that makes a patent distinguishable.

Moreover, the number of retrieved words seems to be an important factor for improving the performance results of automated patent classification. As the number of retrieved words increases the performance of patent classification deteriorates. This happens due to over-fitting that can be resolved with a better optimization of the selected DL model.

Furthermore, the selection of feature words from all parts of a patent at the same time achieves better results than the selection of the first words from each part, and therefore it can be considered as a more suitable representation of a patent.

Last, although the exploration of different processing steps to retrieve the most significant words to represent a patent didn't conclude on insightful results, it can be thought as an initial step for additional experiments towards this direction.

With respect to the language models, the FastText seems to achieve better representation of a patent text and thus better accuracy scores than the other two pre-trained language models (i.e., Word2Vec and Glove) and the Word2Vec language model trained on a domain-specific corpus.

Last but not least, a better distribution of IPC codes among patents can provide improved results, even though it is not practically feasible due to highly unbalanced distribution of patents among categories [LHC+18].

8

Epilogue

Here we present a summary of the diplomatic process and future work.

8.1 Summary and conclusions

Our work on automated single-label patent classification testing several DL methods was mainly driven by our need to experiment and become familiar on how DL methods could be used to develop tools that can effectively support the pre-classification task in patent offices. Additionally, we wished to better understand how various parameters of the DL methods and domain specific parameters, such as the selection of patent document sections/feature words and language modelling representations, may influence automated patent classification.

Initially, we provided the theoretical background of this study, including a brief explanation of patent documents, the classification schemes followed and the elements of data mining process. Then, we reviewed related prior art work using similar DL methods to predict the most representative classification code(s) for a patent. Afterwards, we presented the specific methods used and the evaluation methodology, including the evaluation criteria, the evaluation system, the data collection and the experimental organization. Last, we discussed and evaluated our end-to-end single-label patent classification pipeline employing different DL methods and language modelling representations, considering also different sections/words of the patent document.

The evaluation of different DL methods showed that a bidirectional-LSTM or a bidirectional-GRU implementation can achieve better results than other DL methods, especially when it is combined with FastText word embeddings. Moreover, an ensemble method can provide better results. For example, it produces a promising result when the top 5 or 10 recommended classification codes are predicted.

Patent sections from which the feature words are retrieved and words selected to represent the patent document also seem to play an important role for automated patent classification. Although the abstract section offers the most valuable words for automated patent classification, the description section contains also important information that makes a patent distinguishable. Moreover, the selection of feature words from all parts of a patent at the same time achieves better results than the selection of the first words from each part.

Although significant results derived from all the experiments conducted within this thesis, the highest accuracy score was 65%, achieved when an ensemble of three bidirectional GRU classifiers was used getting as input the title-abstract, the descriptions and the claims section, respectively.

We believe that the initial work we presented in this thesis clearly illustrates that our methods are useful to support patent professionals in a pre-classification task.

8.2 Future extensions

Our plan is to continue this work by focusing on better optimizations of DL models, better representations of patent documents, better exploration of language models and better selection of the data collection. Also, we would like to utilize the hierarchical structure of IPC codes and the description provided for each of them. Last, we plan to extend these approaches on multi-label patent classification and use the total CLEF-IP test collection that might also resolve the over-fitting problems of DL models.

In conclusion, we feel that in this thesis we have already produced some initial knowledge and useful results which will help engineers to produce effective patent classification tools for patent pre-classification or to support other patent management, search and retrieval tasks.

9

Bibliography

[ADK15] S. Altuntas, T. Dereli, A. Kusiak. Forecasting technology success based on patent data. Technological Forecasting and Social Change, 96, pp. 202–214, 2015.

[AKG+19] L. Abdelgawad, P. Kluegl, E., Genc, S., Falkner, F. Hutter. Optimizing neural networks for patent classification. In Joint European Conference on Machine Learning and Knowledge Discovery in Database. pp. 688-703, Springer, Cham, 2019.

[AZ12] C. C. Aggarwal, C. X. Zhai. Mining Text Data. New York: Springer, Science+Business Media, 2012.

[BDK14] M. Baroni, G. Dinu, G. Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers, pp. 238-247, 2014.

[BSP20] J. Bai, I. Shim, S. Park. MEXN: Multi-Stage Extraction Network for Patent Document Classification. Applied Sciences, 10(18), pp. 6229, 2020.

[CC12] Y. L. Chen, Y. C. A. Chang. Three-phase method for patent classification. Information Processing & Management, 48(6), pp. 1017-1030, 2012.

[CMB+14] K. Cho, B. Van Merriënboer, D. Bahdanau, Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259, 2014.

[Dic94] D. T. Dickens. The ECLA classification system. World Patent Information, 16(1), pp. 28-32, 1994.

[Die00] T. G. Dietterich. Ensemble methods in machine learning. In International workshop on multiple classifier systems, pp. 1-15, Springer, Berlin, Heidelberg, 2000.

[FTB+03] C. J. Fall, A. Törcsvári, K. Benzineb, G. Karetka. Automated categorization in the international patent classification. In ACM SIGIR Forum, 37(1), pp. 10-25, New York, NY, USA: ACM, 2003.

[GMB17] M. F. Grawe, C. A. Martins, A. G. Bonfante. Automated patent classification using word embedding. In 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 408-411, IEEE, 2017.

[GSP15] A. Giachanou, M. Salampasis, G. Paltoglou. Multilayer source selection as a tool for supporting patent search and classification. Information Retrieval Journal, 18(6), pp. 559-585, 2015.

[HHY+18] J. Hu, S. Li, J. Hu, G. Yang. A hierarchical feature extraction model for multi-label mechanical patent classification. Sustainability, 10(1), pp. 219, 2018.

[HS97] S. Hochreiter, J. Schmidhuber. Long short-term memory. Neural computation, 9(8), pp. 1735-1780, 1997.

[JGB+16] A. Joulin, E. Grave, P. Bojanowski, T. Mikolov. Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759, 2016.

[KJH+19] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, D. Brown. Text classification algorithms: A survey. Information, 10(4), pp. 150, 2019.

[KZP06] S. B. Kotsiantis, I. D. Zaharakis, P. E. Pintelas. Machine learning: a review of classification and combining techniques. Artificial Intelligence Review, 26(3), pp. 159-190, 2006

[LH20] J. S. Lee, J. Hsiang. Patent classification by fine-tuning BERT language model. World Patent Information, 61, 101965, 2020.

[LHC+18] S. Li, J. Hu, Y. Cui, J. Hu. DeepPatent: patent classification with convolutional neural networks and word embedding. Scientometrics, 117(2), pp. 721-744, 2018.

[LK16] S. Lim, Y. Kwon. IPC Multi-label Classification Based on the Field Functionality of Patent Documents. In International Conference on Advanced Data Mining and Applications, pp. 677-691, Springer, Cham, 2016.

[MCC+13] T. Mikolov, K. Chen, G. Corrado, J. Dean. Efficient estimation of word representations in vector space. In Proceedings of International Conference on Learning Representations Workshop (ICLR-2013), 2013.

[PLH+11] F. Piroi, M. Lupu, A. Hanbury, V. Zenz. CLEF-IP 2011: Retrieval in the Intellectual Property Domain. In CLEF (notebook papers/labs/workshop), 2011.

[PSM14] J. Pennington, R. Socher, C. D. Manning. Glove: Global vectors forward representation. Proceedings of the 2014 Conference on Empiricial Methods in Natural Language Processing (EMNLP-2014), 12, pp. 1532–1543, 2014.

[PYP17] Y. Park, J. Yoon, F. Phillips. Application technology opportunity discovery from technology portfolios: Use of patent classification and collaborative filtering. Technological Forecasting & Social Change, 118, 170–18, 2017.

[RAL+20] A. H. Roudsari, J. Afshar, C. C. Lee, W. Lee. Multi-label patent classification using attention-aware deep learning model. In 2020 IEEE International Conference on Big Data and Smart Computing (BigComp), pp. 558-559, IEEE, 2020.

[RGK20] J. Risch, S. Garda, R. Krestel. Hierarchical Document Classification as a Sequence Generation Task. In Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020, pp. 147-155, 2020.

[RK19] J. Risch, R. Krestel. Domain-specific word embeddings for patent classification. Data Technologies and Applications, 2019.

[Sch02] I. Schellner. Japanese File Index classification and F-terms. World Patent Information, 24(3), pp. 197-201, 2002.

[Sof19] M. Sofean. Deep Learning based Pipeline with Multichannel Inputs for Patent Classification. 1st Workshop on Patent Text Mining and Semantic Technologies. PatentSemTech, 2019.

[STS17] A. Suominen, H. Toivanen, M. Seppänen. Firms' knowledge profiles: Mapping patent data with unsupervised learning. Technological Forecasting and Social Change, 115, pp. 131-142 2017.

[TBT08] D. Tikk, G. Biró, A. Törcsvári. A hierarchical online classifier for patent categorization. In Emerging technologies of text mining: Techniques and applications, pp. 244-267, IGI Global, 2008.

[TLL07] Y. H. Tseng, C. J. Lin, Y. I. Lin. Text mining techniques for patent analysis. Information processing & management, 43(5), pp. 1216-1247, 2007.

[TTW+12] A. J. Trappey, C. V. Trappey, C. Y. Wu, C. W. Lin. A patent quality analysis for innovative technology and product development. Advanced Engineering Informatics, 26(1), pp. 26-34, 2012.

[Wip21] World Intellectual Property Organization (WIPO), World intellectual property indicators 2020, 2021. Available at: https://www.wipo.int/edocs/pubdocs/en/wipo_pub_941_2020.pdf.

[Wol12] B. Wolter. It takes all kinds to make a world–some thoughts on the use of classification in patent searching. World Patent Information, 34(1), pp. 8-18, 2012

[XWZ18] L., Xiao, G. Wang, Y. Zuo. Research on patent text classification based on word2vec and LSTM. In 2018 11th International Symposium on Computational Intelligence and Design (ISCID), 1, pp. 71-74, IEEE, 2018.

[Yu08] B. Yu. An evaluation of text classification methods for literary study. Literary and Linguistic Computing, 23(3), pp. 327-343, 2008.

[ZHF+20] H. Zhu, C. He, Y. Fang, B. Ge, M. Xing, W. Xiao. Patent Automatic Classification Based on Symmetric Hierarchical Convolution Neural Network. Symmetry, 12(2), pp. 186, 2020.

[TLL07] Y. H. Tseng, C. J. Lin, Y. I. Lin. Text mining techniques for patent analysis. Information processing & management, 43(5), pp. 1216-1247, 2007.