University of Passau

Faculty of Computer Science and Mathematics

Chair of Data and Knowledge Engineering

Master Thesis in Informatics

Investigating patent segmentation using artificial intelligence to aid patent semantic search

submitted by

Bochra Smida

1. Examiner:	Prof. Dr. habil. Markus Endres
2. Examiner:	Prof. Dr. Harald Kosch
Supervisor:	Renukswamy Chikkamath
Date:	December 16, 2022

Contents

Abstract

Inventions that are unique, inventive, and non-obvious are eligible for patents, which provide the owner legal protection against others making or selling them. Any person seeking a grant for an invention must prepare a clear and precise patent application document and submit it to the patent office. This is where the examiner plays a role. Examiners frequently need to read patent applications carefully in order to make the right decision to allow or reject them. As part of the examination procedure, the examiner needs patent semantic search to identify pertinent data which is time-consuming as it requires manual work.

In this master thesis, we present two techniques to help with the preliminary steps to aid semantic search. These techniques are claim segmentation and document segmentation. Previous work on patent segmentation focused on hard-coded methods. In this work, we employ artificial intelligence tools using the USPTO dataset. The idea is to segment the claim into separate elements called claim features and segment the reference document into passages called embodiment. The task of the semantic search will be made easy by these two types of segmentation. It would just imply looking up claim features in claim embodiments as answers to queries.

The patent claim segmentation task is done using rule-based, machine learning, and deep learning techniques. The rule-based technique presents claim segmentation as a supervised learning task. For machine learning, we use the random forest, whereas deep learning makes use of CRF, Bi-LTSM, and two different types of pre-trained BERT models. These models were evaluated based on their precision, recall, F1 score, and accuracy. Different scores were achieved by these models considering various parameters. The BERT for patents model achieved the best results, with an F1 score of 88 % and an accuracy of 88 %. For the document segmentation task, we used topic tiling as an unsupervised technique to segment the document using the Latent Dirichlet Allocation topic model, and for the supervised technique, the BERT model is used. Different evaluation scores are used namely P_k, precision, accuracy, recall and F1 score. Our BERT model achieved the best performance with an accuracy of 92% and an F1 score of 81%

Acknowledgments

I would like to express my deepest appreciation to everyone who supported me in my journey. Thanks to you, my studies came to an end.
To Prof. Dr. Markus Endres and Mr. Renukswamy Chikammath, my academic supervisors, I am truly grateful for your support and oversight, as well as your valuable suggestions. I could not imagine having better advisers and mentors for my graduation project.
I am also thankful to Prof. Dr. Harald Kosch for taking on the second examiner's role.
I would like to thank Mr. Christoph Hewel for providing us with insight and guidance as a patent domain expert.
I would also like to extend my gratitude to all of the professors that were a part of my education as well as the members of the jury for having honored me by evaluating my work, in which I hope they can find the clarification and inspiration that they seek.

Dedications

To my father Mahmoud, whose support made me the person I am today. Thank you for always believing in me and for always pushing me to achieve my goals and dreams. Your strength and dedication inspire me to excel in everything I do.
To my mother Faouzia, whose love and encouragement keep me going. Thank you for your constant faith in me and for your confidence that I can do it all. Your kindness and love encourage me to be a better person.
To my siblings, Asma, and Yosra, who constantly encourage me to be the best version of myself. I cannot express my gratitude for your never-ending support. I wish you a bright and prosperous future. To my grandparents, family, and my teachers, who always believed in me. Thank you for your support, trust, and love.
To my friends, Amoul, Saroun, Maryouma, and especially Dali, who are dear to my heart. Thank you for sticking by my side through it all and for being there for me when I needed you most.

List of Figures

List of Tables

1 Introduction

1.1 Introduction and Motivation

Clear language is crucial to guarantee effective communication and reduce the possibility of misunderstanding. The readability of written text serves as a criterion for clarity. There has been an increased effort in recent years to make writing easier to read. These initiatives generally concentrate on making general text simpler to understand for those who require support. This applies to a special category of people who are patent attorneys.
Through a patent, a patent attorney gives an inventor property rights. The innovator receives exclusive rights to the patented process, design, or invention for a specific period of time in exchange for full disclosure of the creation. A written request must be sent to the relevant patent office in order to receive a patent. The patent serves a variety of purposes; in addition to containing instructions on how to create and use the invention—which must be detailed enough for an expert in the field to do so—it also contains claims, which are legal elements that describe the subject matter for which protection is sought. For this reason, their understanding could be a little bit hard and need simplification. After filing the application, the patent examiner reviews the patent application to see whether it complies with the nation's patentability standards. As the patent takes in various details such as the background, summary, and a brief description of the invention, the decision process is not simple and takes time. The way examiners review the patent is different from one examination office to another, the following steps describe an example of the process in the United States Patent and Trademark Office (USPTO) [USP19]:

1) Determine what invention is sought to be patented:
    - Examiners should read the application to find any asserted utility. They should also understand why the invention is thought to be beneficial.
    - Examiners determine the purpose of the invention and the features required to carry out at least one asserted practical application by reviewing the detailed disclosure and specific embodiment of the invention to understand what the applicant has invented.
    - Claim analysis is made in order to determine the limits of the protection that the applicant is requesting while also determining how the claims connect to and describe the innovation that the applicant has claimed. This step is done by identifying each claim restriction and Connecting every claim limitation to the disclosure's descriptions of that limitation. Every limitation contained in a claim must be taken into account when determining its scope. Examiners are not permitted to break down a claimed invention into individual components and then judge the components separately. Instead, one must take into account the entire claim.
2) Conduct a thorough search of the prior art: The outcome of such a search will frequently aid in the examiners' comprehension of the innovation. If there is a reasonable expectation

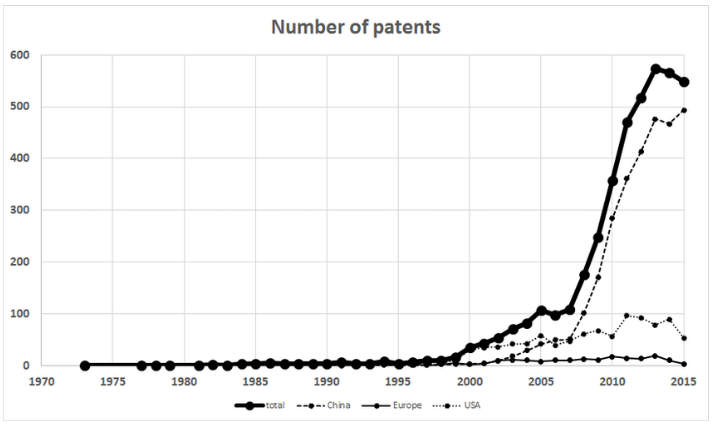


Figure 1.1: Number of patents issued each year

that the unclaimed features of the invention stated in the specification will eventually be claimed, both claimed and unclaimed aspects of the invention should be searched. Any structure or substance mentioned in the specification must be taken into consideration during a search.

3) To make sure the claimed invention conforms with certain laws, additional steps are taken.

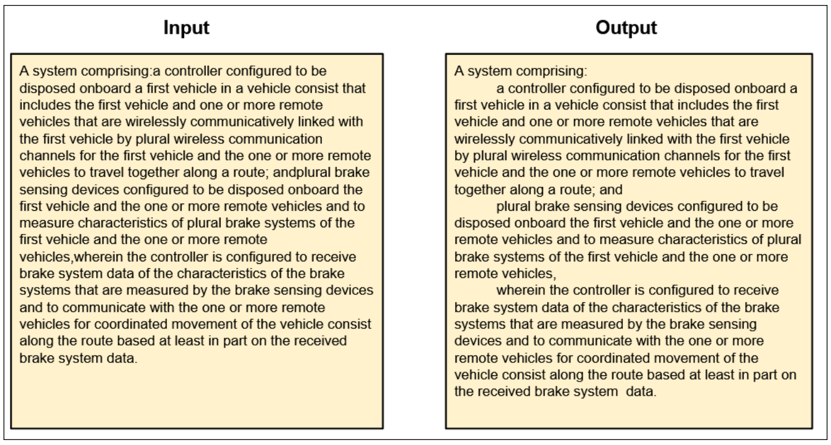
Patent research is becoming more and more important because of different reasons. On the one hand, statistics have shown that the number of issued patents is rising every year. This can be seen in figure 1.1 [PJ17]. On the other hand, Product sales, corporate success, and stock prices all benefit from patent research. Patented businesses have also resulted in improved remuneration, with research showing that they can pay 26 % more than other businesses. The patent provides a wealth of scientific and technological information. Thus, the importance of patent analysis tools such as patent segmentation and patent semantic search, which we will discuss in the following sections.

1.1.1 Patent Segmentation

Patent segmentation is an application of text segmentation from the field of Natural Language Processing (NLP). This process involves splitting independent parts of the patent text. It includes two types of segmentation:

1.1.1.1 Claim segmentation

Analyzing the claim's sub-parts and going through its specifications without missing any information is hard, especially since the claim is always one sentence and the inventor has to add every detail that helps him prove that his invention is novel making patent claims usually long, and breaking them into independent sentences manually is tricky. Claim segmentation is helpful to make claims more readable. As a result, independent chunks that contain useful information, called claim features, are obtained. The claim features are after that used as queries in the patent search. Figure 1.2 shows an example of how a claim could be segmented.



Input

A system comprising:a controller configured to be disposed onboard a first vehicle in a vehicle consist that includes the first vehicle and one or more remote vehicles that are wirelessly communicatively linked with the first vehicle by plural wireless communication channels for the first vehicle and the one or more remote vehicles to travel together along a route; andplural brake sensing devices configured to be disposed onboard the first vehicle and the one or more remote vehicles and to measure characteristics of plural brake systems of the first vehicle and the one or more remote vehicles,wherein the controller is configured to receive brake system data of the characteristics of the brake systems that are measured by the brake sensing devices and to communicate with the one or more remote vehicles for coordinated movement of the vehicle consist along the route based at least in part on the received brake system data.

Output

A system comprising:

a controller configured to be disposed onboard a first vehicle in a vehicle consist that includes the first vehicle and one or more remote vehicles that are wirelessly communicatively linked with the first vehicle by plural wireless communication channels for the first vehicle and the one or more remote vehicles to travel together along a route; and

plural brake sensing devices configured to be disposed onboard the first vehicle and the one or more remote vehicles and to measure characteristics of plural brake systems of the first vehicle and the one or more remote vehicles,

wherein the controller is configured to receive brake system data of the characteristics of the brake systems that are measured by the brake sensing devices and to communicate with the one or more remote vehicles for coordinated movement of the vehicle consist along the route based at least in part on the received brake system data.

Figure 1.2: Input and output for patent claim segmentation

1.1.1.2 Patent document segmentation

This task consists of finding the independent sections that form the patent document. These sections are called embodiments and will serve as answers in the semantic search. Figure 1.3 illustrates an example of embodiments of document segmentation. This process makes finding sections in the reference document simpler. The embodiments could be segmented to find sub-parts that describe different elements of the invention. This task has the advantage of making the examiner's search for a specific detail easier.

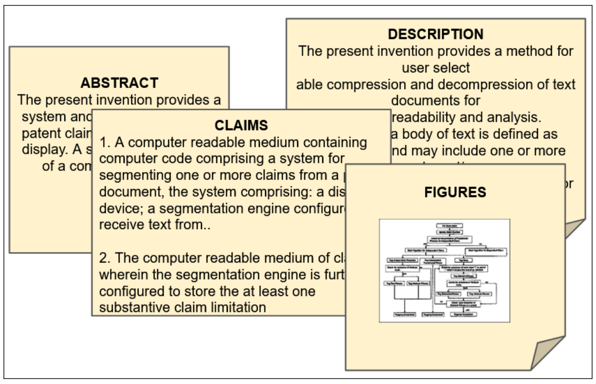

DESCRIPTION
The present invention provides a method for user select
able compression and decompression of text documents for
readability and analysis.
a body of text is defined as
nd may include one or more
ABSTRACT
The present invention provides a
system and
patent clair
display. A s
of a con
CLAIMS
1. A computer readable medium containing computer code comprising a system for segmenting one or more claims from a
document, the system comprising: a dis
device; a segmentation engine configur
receive text from..
2. The computer readable medium of cl
wherein the segmentation engine is furt
configured to store the at least one
substantive claim limitation
FIGURES

Figure 1.3: Embodiments of patent document segmentation

1.1.2 Patent Semantic Search

Semantic Search is a sort of patent search that returns results based on keyword concepts rather than exact keyword matches. Semantic Search is based on the similarity of meaning or semantic content rather than keyword similarity.

Given that the patent examiner has to look for specific details in the reference document when reviewing the claim to check if there is any specification about parts of the claim needed, semantic search offers assistance to the examiner by selecting the sub-parts in the document reference that have a similar topic as the claim. Thus, document and claim segmentation will be used for patent search.

1.2 Research Questions

The contents in a document must be quickly skimmed in order to evaluate the patent. A patent attorney is asked to read the patent claims carefully to analyze every aspect of the claimed invention. They need to go through the patent application, understand its specifications and try to discern the feature that makes it novel. Identifying the sections that discuss technology, innovation, or downsides requires knowledge. For this procedure, they have to make the patent in review more readable by understanding the invention's components and special limitations and they have to carry a search through all the prior-art search to compare the patent with previous work in order to provide a critical examination of a decision in a report. These steps are done manually.
With the rising number of patents in the few years, This process loses effectiveness. According to the European Patent Office (EPO), [EPO22], the European patent grant procedure takes about three to five years from the date the application is filed. Clearly, it is time-consuming and makes the applicants wait a long time to get their grant.
Lately, researchers are becoming aware of these struggles and trying to help in this domain. There are some efforts in aligning patent analysis with new technologies such as [HS86] trying to work on structure recognition of the patent claim. Most of the text simplification processes are hard-coded using rules extracted from patent analysis. Even more recent approaches that use artificial intelligence are using methods that can be used for any text document and their work lacks a deep understanding of the patent domain language. Language understanding with technology is still a difficult topic to solve, especially when it comes to the semantic understanding of the language and the relationships between assertions, despite language understanding being widely used in many fields. To this end, it is necessary to rely on tools that could improve text reading and comprehension by focusing on relationships between elements or models with a focus on the patent domain, which means that a model that understands the patent language is necessary for this type of process. This model's distinctive advantage is that it was developed using data from more than 100 million US patent publications [SY18] which makes it a perfect fit for this study.
So far, there is no study that showed the significance of the BERT model when it is specialized in the patent domain in the field of text segmentation. Most of the previous work worked on general text segmentation techniques. For this reason, adopting this model and evaluating its performance in the field of patent segmentation would be important. In light of this, research questions might be formulated.

a) **How effectively can the model segment the patent claims?**
Examiners must spend hours evaluating the claim; segmenting makes the process of comparison easier. By presenting the patent claims in a more logical manner, it assists

the to decide the overall impact of the invention.

b) **How detailed can claim features be retrieved?**
It could be simple for the examiner to detect the general components of the invention in review. But identifying every single subpart of the invention could be tough but it would be helpful to understand deeper the characteristics that make the invention novel.

c) **How to make the separation of a patent document into embodiments automatic?**
Splitting the document into independent parts now needs a lot of time and concentration from the expert to discern the meaning of each paragraph and how much information about the patent it adds compared to others. The objective is to help the expert get these paragraphs separate and have the essential information that came up from each of them as an overview to the user.

d) **How can bidirectional representation affect patent segmentation?**
Hard-coded segmentation of the reference document can be a solution to get independent sections but a good understanding of the language and relationship between segments turn out to be more reliable and helps to detect the segments effectively.

e) **How to present patent segmentation as a supervised learning task ?**
There is no available dataset that could help especially in claim segmentation. It would be valuable to create a dataset that could help for supervised patent segmentation.

The purpose of claim and document segmentation is to improve the efficiency of semantic search. The idea is to help the expert segment the legal and technical parts to make it easier to detect the relations between them.

f) **How can patent segmentation affect semantic search?**
A claim feature is described in detail in different embodiments of the patent document. Detection of these details can be inaccurate due to embodiments that can be misleading. Therefore, it would be essential to segment these embodiments for a specific semantic search.

1.3 Structure of the Thesis

This thesis contains a total of 8 chapters and it is organized as follows: Chapter 1 is an introduction to the patent domain, more specifically patent segmentation and patent semantic search. It also describes the research questions in relation to these fields. As it is beneficial to have an understanding of the researchers' prior implementation, related work is found in chapter 2. To understand previous artificial intelligence models. Chapter 3 follows, providing background data. This chapter provides definitions for technical terms used in our methodology. Additionally, it includes mathematical and architectural details about AI models. Chapter 4 provides an exploratory analysis of data in our dataset. It includes many visualization graphs that effectively describe the patent dataset. Learning more about the dataset aids in understanding the task. Chapter 5 outlines the implementation details of our approach. It describes how each model works and the tools that are needed for this implementation to work. After making some experiments with our models, the results are provided in chapter 6. It highlights the performance of each model and how its parameters can affect our results. It also gives a comparison between the models' evaluations. Chapter 7 addresses the discussion of the results and finally, chapter 8 provides a conclusion and describes future work.

2 Related Work

In this chapter, we are going to summarize previous work on claim segmentation, document segmentation, and semantic search.

2.1 Claim Segmentation

Claim segmentation comes in order to improve claim readability for better claim analysis. The issue of making patent claims easier to read has not yet been the subject of numerous studies.
Since the claim is a single sentence and special punctuation conventions have been developed and are being used by patent writers. Modern claims follow a specific format. [CF10] is a patent for an engine that uses these assumptions to build a system of rules that could help in segmenting the claim for display. The segmentation engine looks for dependencies to be used for the display too. Each claim is additionally segmented into sub-segments, each of which contains a different component of the claim. Sub-segments are represented using a tree structure. Since this method just considers rules, it is possible for numerous errors to result from this. In fact, the assumption that patent claims follow a particular structure is not entirely accurate. For instance, claims that have a lack of punctuation or have grammatical errors may have an impact on this engine. Thus, an artificially intelligent model is required. The first automatic segmentation was introduced in [FSN14]. They present a two-task method to segment the claim. The first task is to segment it into three parts (preamble, transition, and body) using a rule-based method. Most common transitions like "Comprising", "containing", "including", "consisting of", "wherein", and "characterize in that" are used to detect the transition part. Furthermore, punctuation is used to detect other parts. For instance, the preamble is separated from the transition by a comma, and Each invention in the body text is separated by a semicolon. The second task is based on supervised ML. To train this ML classifier, a patent claim corpus annotated with clause boundaries provided by the TALN Research Group from Universitat Pompeu Fabra is used. Some of their sets involved segmentation issues brought on by consecutive segmentation keywords that produced undesired segments. This occurred because, despite the keywords being consecutive, the classifier segmented each occurrence of a segmentation keyword. The main issue is that the ML classifier uses a statistical approach where it tries to detect frequencies of words and classify them, so there is language understanding which made researchers more interested in claim segmentation using models that detect the relationship between words.

[Gen21] Focuses on the process of Bi-LSTM (Bidirectional Long Short Memory). In contrast to CRF, this model automatically encrypts context information via word vector without designing feature templates and can capture and learn long-term dependence information from the context from both directions. This allows any neural network to have the sequence information in both directions backward. This strategy aids in successfully segmenting the patent. However, it suggests that adopting a novel pre-trained model like a bidirectional encoder representation from transformers (BERT), which recognizes representation at the word and sentence levels, aids in achieving better performance. The benefit of using BERT is that the model learns a specific task when training it, it learns how to deal specifically with patents. [SY18] introduces the first BERT algorithm trained exclusively on patent text. This could improve the results when used in a specific task like patent segmentation. That way it could understand the special aspect that comes with writing a patent.

2.2 Document Segmentation

Recent research on patent document segmentation for Information Extraction can be found in the works of Brugmann et. al. [Brü+15], who described a comprehensive system for patent document analysis that included a variety of techniques, from document segmentation (section identification, and claim spotting) to claim description analysis, entity recognition, and document summarizing. The infrastructure for document representation and algorithm integration in the workbench is provided by the GATE environment.

As for document segmentation in general, different approaches to accomplish that. There are methods that segment the claim based on visual gaps like the work of Daniel Egnor [Egn07] presented as a patent, which uses three visual aspects to segment the claim: the first one is done by identifying a hierarchical structure of the document based on the visual model, the second is by indexing the document and identifying geographic signals in the document, and the third one is by using a device that has a processor and a memory; The memory stores program instructions that, when carried out by the processor, lead it to retrieve a document that comprises geographic signals and divide it into portions in accordance with the document's visual arrangement.
Another way to approach document segmentation is to use natural language processing tools to segment text documents. These tools fall into two categories:

2.2.1 Unsupervised Techniques

In [SON15] Makoto Sakahara, Shogo Okada, and Katsumi Nitta have presented a domain-independent unsupervised text segmentation method. When the domain knowledge is insufficient, determining the semantic similarity between words that make up sentences is typically challenging and this could affect the accuracy. Thus, using Word2Vec to calculate the similarity between words might be a solution for the previous problem. The basic concept of the proposed method is shown in 2.1. After pre-processing the input text, the word2vec model receives each word in a document as input and calculates the semantic similarity matrix which is defined as the cosine similarity between every two words. A collocation similarity matrix is also calculated through sliding as the frequency of words that appear close to a window's central point. The combined similarity is then applied to affinity propagation clustering.

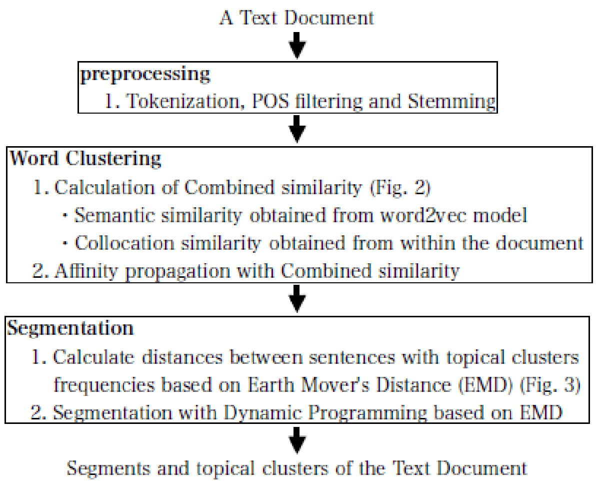

A Text Document
preprocessing
1. Tokenization, POS filtering and Stemming
Word Clustering
1. Calculation of Combined similarity (Fig. 2)
· Semantic similarity obtained from word2vec model
· Collocation similarity obtained from within the document
2. Affinity propagation with Combined similarity
Segmentation
1. Calculate distances between sentences with topical clusters frequencies based on Earth Mover's Distance (EMD) (Fig. 3)
2. Segmentation with Dynamic Programming based on EMD
Segments and topical clusters of the Text Document

Figure 2.1: Basic concept of domain-independent unsupervised text segmentation method

Finally, Segmentation boundaries are automatically optimized using dynamic programming when the similarity between sentences has been determined.

Another technique [Hea97] called TextTiling helps to segment text into multi-paragraph subtopic passages. This method is mostly used for discourse segmentation but it can equally be used for document segmentation. It is based on the idea that if two blocks of text are similar, it is more likely that the current subtopic continues. To employ this strategy, the text is tokenized into token sequences, stop words are eliminated and tokens are stemmed. Next is similarity determination using a sliding window to compare blocks of token sequences. The block-wise similarity is calculated via cosine measure. The final step is boundary identification, the sequence of similarity scores is used to find "changes" over the line to calculate the highest "depth scores" that correspond to the boundary. 2.2 is an example of the result of the block similarity algorithm with a window set equal to 10. both the smoothed and unsmoothed plots are shown. Internal numbers indicate paragraph numbers, the x-axis indicates the token-sequence gap number, and the y-axis indicates the similarity between blocks centered at the corresponding token-sequence gap. Vertical lines indicate boundaries chosen by the algorithm.

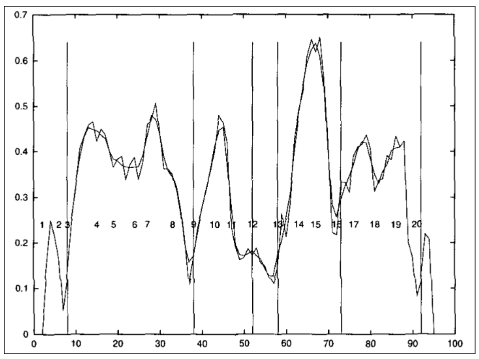


Figure 2.2: Result of the block similarity algorithm

2.2.2 Supervised Techniques

Unsupervised algorithms suffer from two main drawbacks: they are hard to specialize for a given domain and in most cases do not naturally deal with multi-scale issues. To overcome these challenges, a supervised method can be advantageous.
In 2018, a study [Bad+18] was the first one to present a novel supervised neural and attention based approach. Text segmentation is presented as a binary classification task where the input is sentence with K sentence on the left and K on the right as its neighbourhood information. K is the context size. and the prediction tells whether the sentence begin a new segment or not. After padding the sentence to prepare for fixed length embedding, 300D word2Vec embeddings are used and one-hot representations are used for the context sentences. Sentence embeddings are learned using CNNs, and the segments are predicted using context information in the proposed attention-based bidirectional LSTM model. This model is capable of handling context data of various sizes automatically and the evaluation using WinDiff of this model showed that it improves the performance around 7% compared to other solutions.

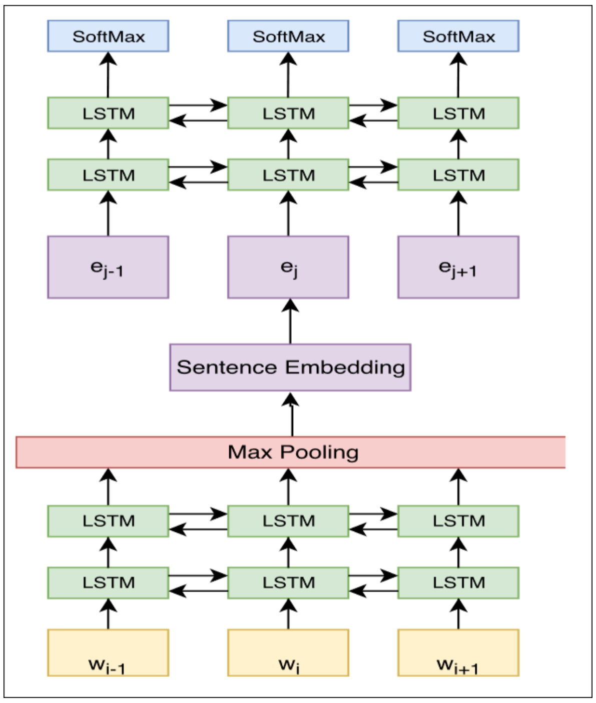


Figure 2.3: Architecture of a supervised text segmentation model

[Kos+18] deals with text segmentation as a supervised learning task. The process is a hierarchical neural model in which word tokens are converted into sentence representations by a lower-level bidirectional LSTM, which is subsequently consumed by a higher-level LSTM, which labels each sentence. the input x is a document represented as a sequence of n sentences (s_1, .., s_n). As shown in figure 2.3For each sentence s_i, the lower-level sub-network consumes the words w_1^i,.., w_k^i of s_i one at a time, and the final sentence representation e_i is computed by max-pooling over the LSTM outputs. The higher-level network then takes a sequence of sentence embeddings e_1,.., e_n as input, and feeds them into a two-layer bidirectional LSTM.

Next is to apply a fully-connected layer on each of the LSTM outputs to obtain a sequence of n vectors. Finally, a softmax function is applied to obtain n - 1 segmentation probabilities. The model predicts for each sentence s_i, the probability pi that it ends a segment.

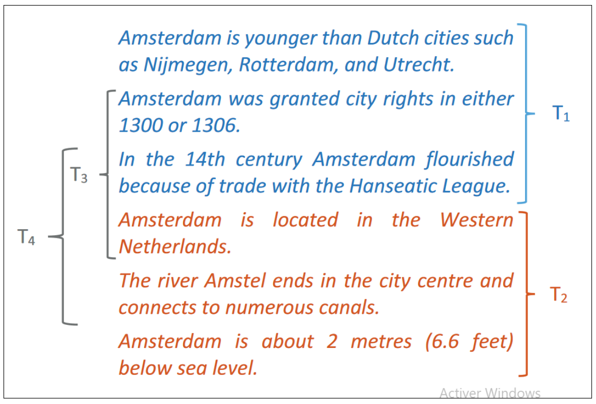
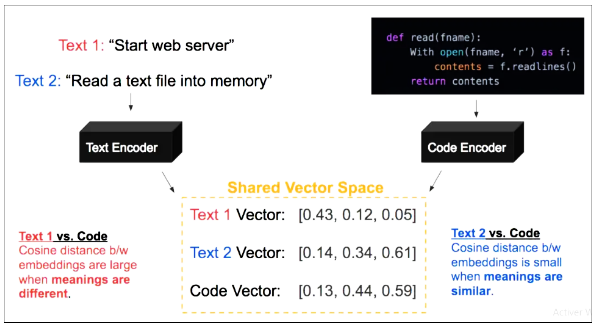

Amsterdam is younger than Dutch cities such as Nijmegen, Rotterdam, and Utrecht.
Amsterdam was granted city rights in either 1300 or 1306.
In the 14th century Amsterdam flourished because of trade with the Hanseatic League.
Amsterdam is located in the Western Netherlands.
The river Amstel ends in the city centre and connects to numerous canals.
Amsterdam is about 2 metres (6.6 feet) below sea level.
T1
T2
T3
T4
Activer Windows

Figure 2.4: Relation between text coherence and segmentation

A multitask learning model was proposed by Glava et al. [GS20] that combines the coherence objective, which distinguishes between correct and corrupt sentence sequences, with the sentence-level segmentation objective. The figure 2.4 shows the difference between coherence and segmentation. where T1 and T2 are more coherent than T3 and T4: all T1 sentences relate to Amsterdam's history, and all T2 sentences to Amsterdam's geography; in contrast, T3 and T4 contain sentences from both topics. T1 and T2 being more coherent than T3 and T4 signals that the fourth sentence starts a new segment. A neural architecture consisting of two hierarchically connected Transformer networks is used to break down the structure of large texts into semantically coherent segments, making the texts more readable. When coupled with cross-lingual word embeddings, the suggested model, known as Coherence-Aware Text Segmentation (CATS), improves its ability to segment texts in languages that were not trained on.

2.3 Semantic Search

Semantic Search is a sort of patent search that returns results based on keyword concepts rather than exact keyword matches. Semantic Search is based on the similarity of meaning or semantic content rather than keyword similarity.
Patent search relies on patent segmentation which consists of retrieving the most important parts in the patent using natural language processing (NLP).
In general, search engines were lexical, meaning they only returned links that contained the precise query in them and searched for words that were a literal match to the query. When doing a standard keyword search, a document either includes the requested word or it does not. However, because semantic search is supported by an automated NLP tool, it can make the process of creating queries simpler.
A basic approach to handle this type of task is described in [Raj20]. This work constructs



```
def read(fname):
    With open(fname, 'r') as f:
        contents = f.readlines()
    return contents
```

Text 1: "Start web server"
Text 2: "Read a text file into memory"
Text Encoder
Code Encoder
Shared Vector Space
Text 1 Vector: [0.43, 0.12, 0.05]
Text 2 Vector: [0.14, 0.34, 0.61]
Code Vector: [0.13, 0.44, 0.59]
Text 1 vs. Code
Cosine distance b/w embeddings are large when meanings are different.
Text 2 vs. Code
Cosine distance b/w embeddings is small when meanings are similar.

Figure 2.5: High level intuition of how to construct a shared vector-space

the corpus's vocabulary, giving each unique term an ID and storing its frequency coefficients. After that, the doc2bow method is used to obtain a bag of words that describes the occurrence of words inside a document for feature extraction. The next step is to construct the Term frequency-Inverse Document Frequency (Tf-Idf) model, which aids in identifying the most crucial words in each document in the corpus before submitting them to the Latent Semantic Index (LSI) model. Search engines employ the idea of LSI to determine how a keyword and content combine to mean the same thing.LSI is founded on the idea that words used in comparable contexts frequently have related meanings.

In [HW18], researchers came up with a procedure so that developers could search for code in repositories even if they are unfamiliar with the syntax or can't think of the right keywords. The idea is to construct a shared vector-space for the text and the code as illustrated in figure 2.5 . The objective is to map code into the vector space of natural language, in a way that the cosine similarity between (text, code) pairs can be measured. That means, pairs that describe the same concept are close neighbors, whereas dissimilar (text, code) pairs are further apart. In order to achieve this, a pre-trained model that extracts features from code and fine-tuning this model to project latent code features into a vector space of natural language is used.

3 Background

In this section, we will be discussing sub-fields in artificial intelligence that can help with problems in natural language processing (NLP). We will define important terms used in our methods.

3.1 Artificial Intelligence (AI)

Artificial intelligence is founded on the idea that human intelligence can be described in a way that makes it simple for a computer to duplicate and carry out activities of any complexity. It aims to emulate cognitive processes in humans. So the term artificial here is opposed to natural, like the human brain. Although AI is based on trying to imitate the intelligence of humans, it has the potential to outperform them in some activities. Its technologies frequently finish work fast and with minimal mistakes, especially when it comes to repetitive, detail-oriented activities like reviewing tons of legal papers to verify that key fields are filled in correctly.

One must comprehend how AI systems function in order to use them successfully. As figure 3.1 shows, speech, text, images, and other types of data can be entered into AI systems. The system then interprets, predicts, and takes action on the input data by using a variety of rules and algorithms. After processing, the system returns a result, such as success or failure, depending on the data input. The output is then evaluated through analysis, research, and feedback. Finally, the system modifies input data, rules and algorithms, and desired outcomes based on its evaluations. This cycle is repeated until the desired outcome is obtained.

As shown in figure 3.2 [Sof19] among the many fields that make up artificial intelligence are

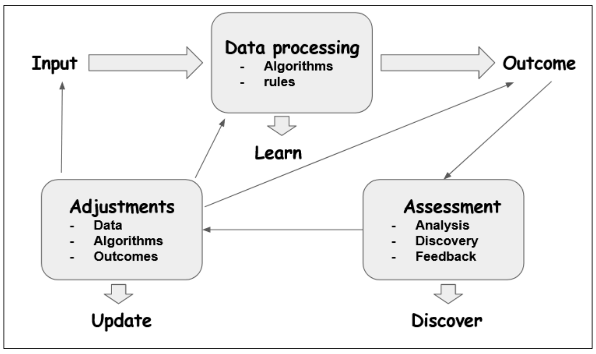


Figure 3.1: How Artificial Intelligence systems work

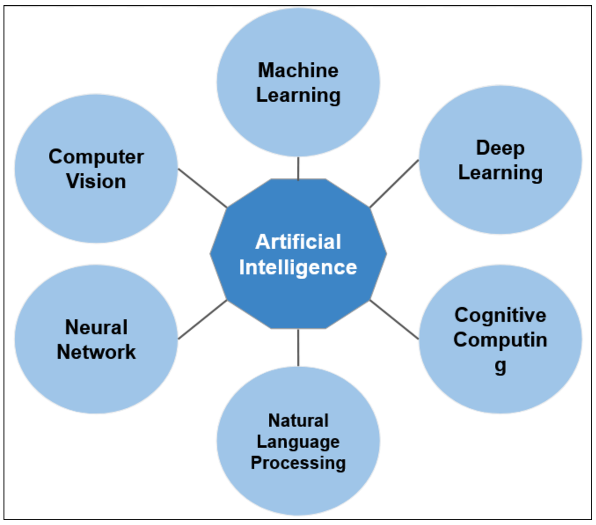


Figure 3.2: Sub-fields of AI

machine learning, deep learning, and natural language processing. In order to achieve better results, all AI components need to work in conjunction with each other.

3.1.1 Machine Learning (ML)

The idea of machine learning has existed for quite some time. AI and computer gaming pioneer Arthur Samuel [WM92], a computer scientist at IBM, is credited with coining the term "machine learning." Samuel created a checkers game for the computer. The more it was used, the more the program learned from its mistakes and used algorithms to predict outcomes.
ML is an application of AI that gives computers the ability to learn from experience and get better over time without having to be explicitly programmed. The goal of machine learning is to create computer programs that can access data and use it to acquire knowledge on their own. Figure 3.3 highlights the key difference between machine learning systems and traditional systems which is that in traditional programming, input data and a well-crafted and verified program would be sent into a machine to produce output. In the learning phase of machine learning, input data and output data are provided to the machine, and it creates a program on its own.
The way an algorithm learns to make predictions that are increasingly accurate is one way that machine learning is frequently categorized. There are four fundamental approaches: reinforcement learning, semi-supervised learning, unsupervised learning, and supervised learning.

3.1.1.1 Supervised Learning

In this type, the data scientist must have labeled input data. It helps to collect data or produce a data output from a previous machine learning deployment. This works in the same way the human actually learns. The common supervised learning tasks are Linear regression, Logistic regression, Decision tree, Naive Bayes algorithm, and K-means.

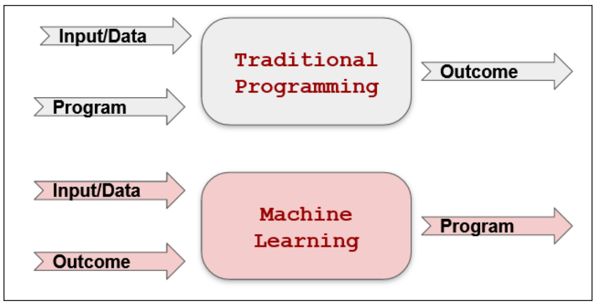


Figure 3.3: Difference between Machine Learning and traditional systems

3.1.1.2 Unsupervised Learning

In this type, the data provided is not labeled which means the data scientist is not obliged to provide the output. The method detects any meaningful relation helping to find all kinds of unknown patterns in data. Two common unsupervised learning tasks are clustering and dimensionality reduction.

3.1.2 Deep Learning (DL)

Deep learning is a subset of machine learning algorithms in which the tasks are divided into smaller components and sent to machine learning algorithms that are arranged in successive layers. Each layer builds up on the results of the preceding layer. Together, the layers form an artificial neural network that resembles how neurons in the human brain carry out distributed problem-solving [War11].
Deep learning analyzes big data sets using neural networks. To make conclusions or make a forecast, data must first be extracted and then examined. Deep learning is typically applied in the real world to recognize photos, comprehend texts, or improve decisions. Neural networks, the foundation of deep learning, have been studied since the early 1940s, so the subject is not new. However, because to big data and rising processing power, it is quickly attracting attention. Deep learning requires a lot of computing. To develop the skills necessary to make accurate forecasts and decisions, training may take months. Millions of model parameters and complicated architectures are to blame for this.

As shown in figure 3.4, the main difference between deep learning and classic machine learning is the ability to use artificial neural networks (ANN) to process unstructured data. While machine learning use hand-crafted features which is tedious and costly to develop, deep learning learns hierarchical representation from the data itself and scales with more data. Artificial neural networks are capable of converting text, images, sounds, and movies that are not structured into numerical values. The subsequent pattern recognition and learning processes employ this retrieved information. This unstructured data cannot be successfully processed using traditional machine learning techniques. With traditional techniques, it is difficult to implement use cases like picture recognition since these models cannot capture complex relationships. Another distinction is that in deep learning, the algorithm handles practically all of the feature engineering, or data pre-processing. Natural language processing

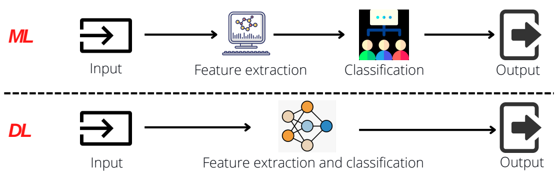


Figure 3.4: Difference between Machine Learning and Deep Learning

(NLP) has seen improvements, speedups, and automation in recent years because of the use of a variety of deep learning models. For instance, this field excelled in multilingual tokenization and text classification. In fact, using word embeddings, which determine the vector value of words, deep convolutional neural networks (CNN) and recurrent neural networks (RNN) can automatically classify the tone and sentiment of the source text. Additionally, CNN-based speech recognition systems can convert unprocessed audio into a text message that provides intriguing details about the speaker. Further, where conventional methods fall short, recurrent neural network models may appropriately respond to questions that are the length of a paragraph[HW15].

3.1.3 Natural Language Processing (NLP)

NLP is a branch of artificial intelligence that focuses on enabling machines to comprehend, produce, or translate human language as it is written and/or spoken. It entails the research and development of several systems as well as the mathematical and computational modeling of various linguistic characteristics [Jos91]. Artificial intelligence and linguistic processing are combined by NLP. The most recent generation of NLP systems is based on artificial neural networks or simple statistical machine learning models. There have been learning models trained on significant amounts of text. In the following sections, some tools and sub-fields related to NLP are described.

3.1.3.1 Named Entity Recognition (NER)

Named Entity Recognition, also known as entity chunking, extraction, or identification, is a form of natural language processing (NLP), a sub-field of artificial intelligence. It is the process of finding and classifying important information (entities) in text. Any word or group of words that constantly refers to the same item is considered to be an entity. Each recognized object is put into a specific category. For instance, a NER ML model might detect the word ”Passau” in a text and classify it as a place.
A NER model is a two-step process:

a) Detects a named entity: detects a word or a string of words where each word is represented by a token. At this step, a BIO (B: Beginning, I: Inside, O: Outside) tagging is commonly used to indicate the start and end of an entity.

b) Categorizes the entity: The creation of entity categories is needed and these categories depend on the task.

NER is appropriate for any circumstance where a high-level summary of a large body of information is useful. With NER, one may swiftly group texts based on their relevance or similarity and comprehend the subject or theme of a body of material at a look. A few noteworthy NER use cases are summarizing resumes to speed up recruiting process in Human resources and extracting crucial information from lab results to raise patient care standards in health care.

3.1.3.2 Latent Dirichlet Allocation (LDA)

LDA [BNJ03] is among the most widely used topic modeling techniques. Each document is composed of a variety of words, and each topic likewise has a variety of terms. The purpose of LDA is to determine the themes a document belongs to, depending on the words in it. For example, let's say we have a document containing a list of words, each word can be assigned a probability that it belongs to a topic. So words like "employer", and "CEO" will have a higher probability in the work-related topic wherever words like "teacher", and "student" will be school related. Before using LDA, some assumptions are fixed:

- Each document is a bag of words. Consequently, the grammatical role and position of the word don't matter.
- Words that don't carry any information like this/the/but are omitted as a preprocessing step.
- The number of topics is already known.
- All words are assigned to the right topic except for the word we are working on, then they will be updated one by one using the model of how documents are generated.

Thus, LDA has mainly two parts: the words that belong to a document which they are already known, and the words that belong to a topic or the probability of belonging to a document that needs to be calculated. It is an application of the Bayesian approach of text processing. For each sentence in a given text, it returns the probability distribution over subjects. To find the probability distribution, LDA goes through the following steps:

1) For each document, it assigns random topics to each word from the list of K topics.
2) For each document d, for each word w, computes:
   - p(topic t | document d): How many words in a given document d are related to the topic t without including the word at hand.
   - p(word w | topic t): Because of the word w, how many documents are in topic t? LDA represents documents as a mixture of topics. Similarly, a topic is a mixture of words. That means, if a word has high probability of being in a topic t, all documents containing this word will be associated to this topic.
3) Update the probability of the word w belonging to a topic t using the equation 3.1 LDA iterates over this step defined number of times. This is one of the parameters for fine–tuning

LDA model.:

$$p(w,t) = p(t|d) * p(w|t) \tag{3.1}$$

Knowing that LDA assumes that topic distribution have Dirichlet prior as in 3.2:

$$p(\theta/\alpha) = \frac{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \prod_{i=1}^{K} \theta_i^{\alpha_i - 1} \tag{3.2}$$

Where $\theta = [\theta_1 ... \theta_K]$ represents probabilities that are equivalent to the proportions of certain topics, the α parameter which controls how widely a topic is covered in documents, the β parameter which determines how closely related words must be in order for them to be assigned to a topic.

3.1.3.3 Topic Tiling

Topic Tiling algorithm is first introduced in (Riedl and Biemann, 2012a) [RB12]. It is based on the last topic IDs assigned by the Bayesian Inference method of LDA rather than on words. Therefore, the word space is reduced to a topic space with lower dimension.
The documents that need segmentation, need to be annotated with topic IDs. For better results, the model needs to be trained on similar-content documents to the test documents. The smallest basic unit for this algorithm is a sentence s_i, a coherence score (c_p) is calculated at each location p between two sentences that are next to one another. a window w is introduced to specify the number of sentences on the left and the right of a position p, that way we get two blocks: $(s_p - w,..,s_p)$,$(s_p + 1,..,s_p + w + 1)$. This last parameter is dependent on the number of sentences a segment should contain. Using LDA wik=th K topics, each block is represented by a K-dimensional vector. each element in the vector represents the topic ID frequency obtained from this block. After that, the coherence score is the cosine similarity to show connectivity between two adjacent blocks. In fact, values that are close to one show significant relation, however, values close to zero show a minimal relation. The next step is to calculate minimum values using depth score d_p to find the possible segmentation boundaries. By examining the highest coherence scores on the left and right, the depth score evaluates how deep a minimum is, and it is determined using the equation 3.3:

$$d_p = \frac{1}{2}\left(hl(p) - c_p + h_r(p) - c_p\right) \tag{3.3}$$

Where hl(p) and hr(p) iterate respectively to the left or right and return the highest coherence score value. The n highest depth scores are used as segment boundaries if n is the number of segments. If not, a threshold is used so that the model predicts a segment when the depth score is greater than $\mu - \sigma/2$ where $\mu and \sigma$ are respectively the mean and standard deviation of the depth score.

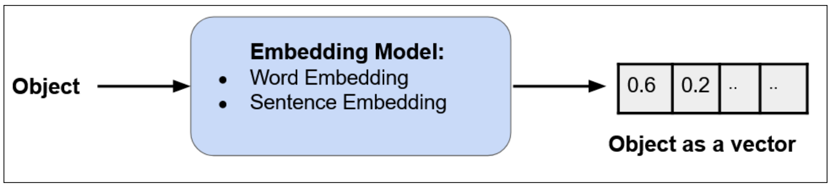


Figure 3.5: Embedding Model

3.1.3.4 Text Representation

As machine learning algorithms need numbers to work with, embeddings are the most useful concept in natural language processing. Numerical or categorical datasets are easy to use in machine learning models but if we encounter something more abstract, like a large text document. For data like this, we build vector embeddings, which are simply lists of numbers, and execute various operations on them. Any item, including an entire paragraph of text, can be converted to a vector. For simpler processes, even numerical data can be transformed into vectors. Figure 3.5 describes the general process of an embedding Model for any type of object.
Vectors, however, are unique in a way that makes them very helpful. It is possible to transform human perceptions of semantic similarity to closeness in a vector space using this representation. For instance, clustering algorithms try to assign similar vectors to elements in the same cluster and keep elements from different clusters as dissimilar as possible. That means, the proximity of these objects and ideas to one another as vector space points can be used to measure their semantic similarity. ——— talk about sentence embedding and hot embedding...(TFID..)——

3.2 ML Models

Decision Trees:
Decision trees are a type of tree used in data exploration and business decision-making technology. They employ a hierarchical representation of the data structure in the form of decision-making sequences with the goal of predicting a result or a class. Each observation that needs to be assigned to a class is described by a group of variables that are put to the test in the nodes of the tree. The internal nodes are where internal tests are conducted, and the external nodes are where final decisions are made. **Random Forest Classifier:**
The random forest classifier is Decision Trees. It consists of a group of decision trees working together to deliver a result. Every single tree in the random forest gives a class forecast, and the classification that receives the most votes becomes the prediction made by our model. Random forests do not overfit and it is fast.

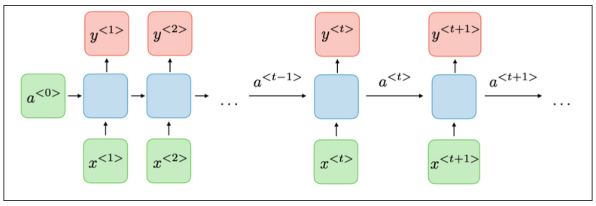


Figure 3.6: Traditional RNN architecture

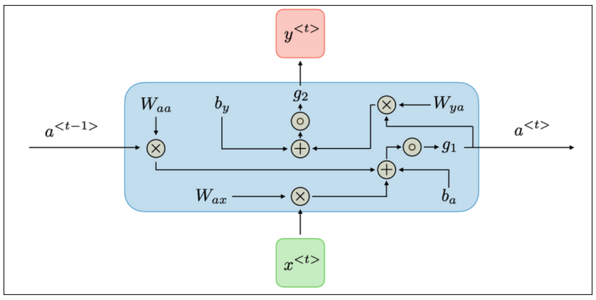


Figure 3.7: RNN at timestamp t

3.3 DL Models

3.3.1 Recurrent Neural Network (RNN)

Recurrent neural networks RNNs [RHW17], are a type of neural network widely used in the field of Deep Learning [LBH15]. RNNs use previous outputs as additional inputs and are well-suited for processing sequential data. Typically, they take the form [AA18] in figure 3.6

Where at timestamp t as shown in the figure 3.7 the relation are expressed as in the equations 3.4:

- a(t) is the activation.
- y(t) is the output.
- W_ax,W_aa, W_ya, b_a,b_y are coefficients that are shared temporally.
- g_1,g_2 are activation functions.

$$\begin{aligned} a^{<t>} &= g_1\left(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a\right) \\ y^{<t>} &= g_2\left(W_{ya}a^{<t>} + b_y\right) \end{aligned} \tag{3.4}$$

In contrast to a traditional ANN (Artificial Neural Network), where the output depends only on the input values, we can clearly see the recurrent aspect of these calculations (the calculation at time t is based on the information provided at time t-1, itself calculated from the information provided at time t-2, etc.). Figure 3.8 roughly describes the difference in the architecture. The loss function in neural networks follows the first equation 3.5. After that,

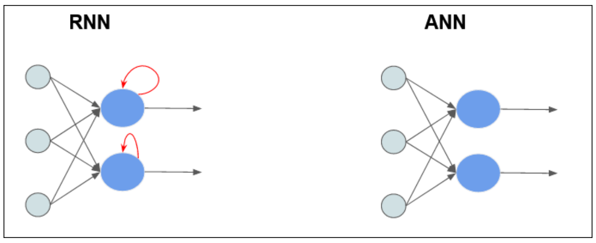


Figure 3.8: Difference between ANN and RNN

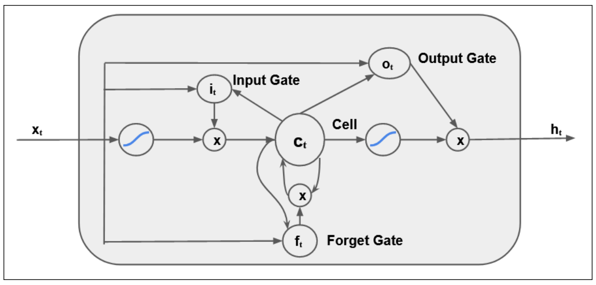


Figure 3.9: Components of an LSTM cell

the backpropagation through time is done as expressed in the second equation 3.5.

$$\begin{aligned}\mathcal{L}(\widehat{y}, y) &= \sum_{t=1} \mathcal{L}\left(\hat{y}^{<t>}, y^{<t>}\right) \\ \frac{\partial \mathcal{L}^{(T)}}{\partial W} &= \sum_{t=1} \frac{\partial \mathcal{L}^{(T)}}{\partial W}|t\end{aligned} \tag{3.5}$$

Although RNNs are very convenient compared to a classical ANN architecture for processing sequential data, it turns out that they are extremely difficult to train to handle long-term dependency [BSF94] due to the gradient vanishing problem. To overcome these shortcomings, new RNN variants have been introduced into the literature such as Long short-term memory (LSTM), Gated Recurrent Unit (GRU).etc

3.3.2 Long Short Term Memory (LSTM)

The LSTM is first proposed by S. Hochreiter [HS97] and subsequently enhanced in the publication by F. Gers and j. Schmidhuber [GSC00] to address the gradient vanishing issue.
In order to model very long-term dependencies, it is necessary to give recurrent neural networks the ability to maintain a state over a long period of time. This is the goal of LSTM cells, which have an internal memory called a cell. The cell allows for maintaining a state for as long as necessary. This cell consists of a numerical value that the network can control according to the situation. The LSTM unit, the fundamental part of an LSTM architecture, is depicted in figure 3.9.
An LSTM forward pass is modeled by the equations 3.6

$$\begin{aligned}
&f_t = \sigma\left(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f\right)\\
&i_t = \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right)\\
&\widetilde{c_t} = \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right)\\
&c_t = f_t * c_t 1 + i_t * \widetilde{c_t}\\
&o_t = \sigma\left(W_o x_t + U_o h_{t-1} + b_o\right)\\
&h_t = o_t * \tanh\left(c_t\right)
\end{aligned} \tag{3.6}$$

where:

- σ is the sigmoid function
- f_t is the forget gate activation vector
- i_t input gate activation vector
- o_t output gate activation vector, cell input activation vector
- c_t cell state
- h_t output vector of the LSTM unit
- all W and U are weights
- b is a bias vector
- the symbol * for the Hadamard product
- Weights W, U, and biases b must be learned during the training process

3.3.3 Bidirectional Bi-LSTM

Bi-LSTM is used primarily on natural language processing. Unlike standard LSTM, the input flows in both directions. and it is able to make use of data from both sides. In both directions of the sequence, it is a potent tool for modeling the sequential relationships between words and phrases. One more LSTM layer is added by Bi-LSTM, which changes the information flow's direction. It simply means that in the additional LSTM layer, the input sequence flows backward. The outputs from the two LSTM layers are then combined in a variety of ways, including average, sum, multiplication, and concatenation. This type of architecture comes in favor of different real-world problems, especially when dealing with natural language processing. In fact, By combining LSTM layers from both directions, Bi-LSTM can create an output that is more meaningful. This can be illustrated by the figure 3.10 of the architecture of a Bi-LSTM [SV19].

For instance, let's imagine we want to predict the following word in a sentence. A unidirectional LSTM will, at a high level, see that

"The students opened their.."

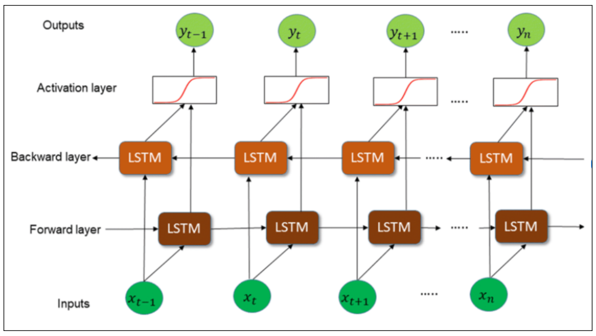


Figure 3.10: Bi-LSTM architecture

With the help of bidirectional LSTM, you will be able to view information further down the road, for instance, and will attempt to anticipate the next word purely based on this context.
Forward LSTM: "The students opened their"
Backward LSTM: "and then they opened their emails."
One can see how it could be simpler for the network to comprehend what the next word is if it uses knowledge from the future.

3.3.4 Conditional Random Field (CRF)

CRF is a discriminative model. It is based on a conditional approach to label and segment data sequences and is in a probabilistic context. The main advantage of CRFs is that their conditional nature allows us to relax the assumptions made about the independence of observations.
Let G = (V, E) be a graph such that Y = (Y_v) $v \in V$. V defines the set of nodes, and E is the set of arcs. Y is therefore indexed with the nodes of G.
CRF is a special case of Markov Random field that satisfies the property:
when the random variable X is fixed, the output Y follows the equation 3.7 where $\sim$ signifies neighbors in the graph:

$$P\left(Y_u \mid X, Y_v, u \neq v\right) = P\left(Y_u \mid X, Y_v, u \sim v\right) \tag{3.7}$$

This property is thus satisfied if the state of the system (the node in which one is) depends only on the neighboring states (nodes), as well as on the probabilities of transitions between the states.
the joint distribution of label sequences Y knowing X is described by the equation 3.8 where:

- fk and gk are (fixed) characteristics
- λk and μk are parameters to be estimated during learning.

$$P_\theta(y \mid x) = \exp\left(\sum_{e \in E, k} \lambda_k f_k\left(e, y|_e, x\right) + \sum_{v \in V, k} \mu_k g_k\left(v, y|_v, x\right)\right) \tag{3.8}$$

The learning phase consists in estimating the values of the parameters ($\lambda 1$,$\lambda 2$,$\lambda 3$..,$\mu 1$,$\mu 2$,..) from a training base $D = \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$ with the empirical distributions $\tilde{p}(x, y)$.
The goal is to maximize the log-likelihood described by the equation 3.9 :

$$\begin{aligned} O(\theta) &= \sum_{i=1}^{N} \log P_\theta\left(y^{(i)} \mid x^{(i)}\right) \\ &\simeq \sum_{x,y} \tilde{p}(x, y) \log p_\theta(y \mid x) \end{aligned} \tag{3.9}$$

3.3.5 Bidirectional Encoder Representations from Transformers (BERT)

Transformers:
A transformer model is a neural network that follows relationships in sequential input, such as the words in this sentence, to learn context and subsequently meaning. Transformer models use an expanding collection of mathematical approaches known as self-attention relationships between data pieces in a series. The self-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence.

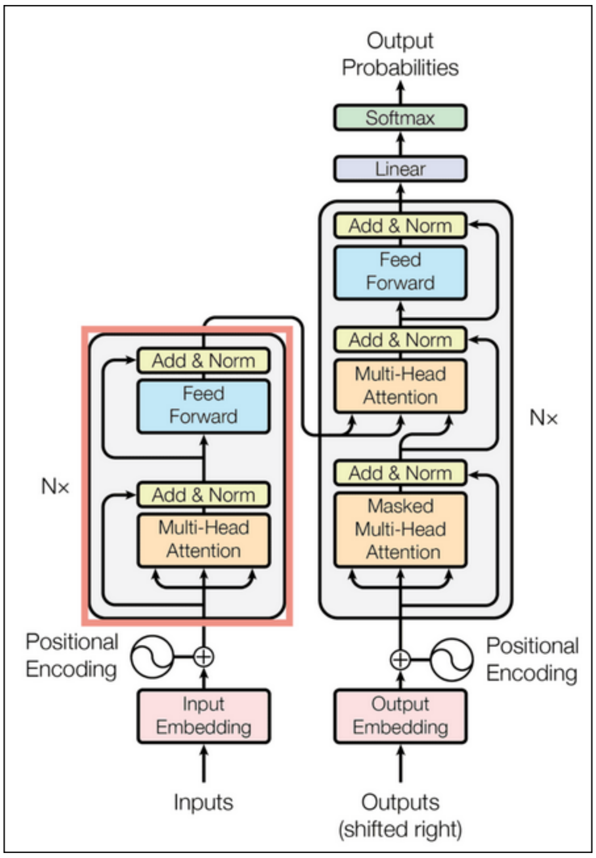


Figure 3.11: Transformer architecture

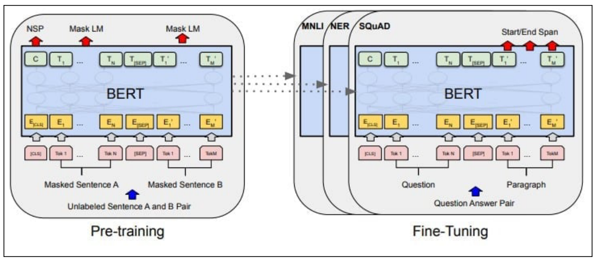


Figure 3.12: BERT Pre-Training and Fine-Tuning

Transformer models are essentially massive encoder/decoder blocks that process data, similar to the majority of neural networks. However, they are particularly powerful due to little but clever changes to these blocks as shown in figure 3.11.
Basically, the encoder's function is to translate an input sequence into a sequence of continuous representations, which is then fed into a decoder, which is located on the left half of the Transformer architecture. In order to create an output sequence, the decoder, located on the right half of the architecture, gets both the encoder's output and the decoder's output from the previous time step. The most important property is that "At each step the model is auto-regressive, consuming the previously generated symbols as additional input when generating the next" [Vas+17].

BERT:
BERT is a pre-trained deep-learning natural language framework developed by GOOGLE. Since its release, the transformer BERT has captured the interest of the data science community with its "State-of-the-art" results, which means that for the vast majority of NLP tasks, Bert provides the best results right now. Even Google advanced to the point where this model outperformed human performance on several tasks, which is a significant step for any machine learning algorithm. It allows the language model to discern word context in a text by learning information from both the left and right side of a token's context during the training using the word embedding.
BERT learns in an unsupervised way. The learning is done in two main phases as shown in figure 3.12:

Pre-training:
the model is trained on unlabelled data over different pre-training tasks. The aim of this step is to understand the language of the input.
To understand how the embeddings of BERT aid to make it more functional and quick, one must understand the method that is used:

- **Masked Language Modeling:** The purpose of BERT is to fill in the blanks, or to output the mask tokens, in phrases with blanks (referred to as "Masks"). This aids BERT in comprehending a language's bi-directional context.
- **Next Sentence Prediction:** Examines if the second sentence comes after the first when given two. This improves BERT in comprehending context in various sentences.

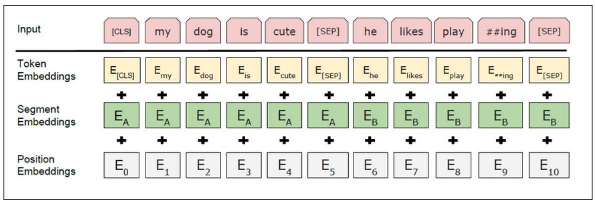

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 3.13: BERT input representation

Fine-tuning: this is the stage where we continue to update the weights trained from the pre-training phase, on the dataset in use for our problem. That means that in this phase the model will learn a specific task to answer the question we want.

This model converts the words to embeddings to make it easy for the model to work with. It uses three different embeddings to create its input embeddings. The process of combining the embeddings to create the final input token is depicted in the diagram 3.13 where:

- **Token embeddings** are the pre-trained embeddings for different words
- **Segment embeddings** are the sentence number that is encoded into a vector.
- **Position embeddings** are the position of the word within that sentence that is encoded into a vector.

3.4 Evaluation Methods

In this section, we will go through the different methods for evaluating the performance of our deep learning or machine learning model as well as the benefits of using one over the other. These methods are commonly used to better understand and assess the effectiveness of classification models.
First, to better understand the methods' calculations, we should clarify the meaning of the following terms:

- **True Positive (TP):** The real and predicted values are identical and positive.
- **False Positive (FP):** The predicted values are positives but different from the real values.
- **True Negative (TN):** The Real and predicted values are identical and negative.
- **False Negative (FN):** The predicted values are negatives but different from the real values.

3.4.1 Precision

Accuracy refers to the proportion of accurate predictions among positive predictions. It assesses the model's capacity to avoid making mistakes when making a positive prediction.

In our model, it's defined as the ratio of the number of patents that the model correctly segments to the number of all events the model recalls as shown in the equation 3.10.

$$\text{Precision } = \frac{TP}{TP+FP} \tag{3.10}$$

3.4.2 Recall

The recall, also known as sensitivity or true positive rate, is the rate of positive values observed by the model. It is defined as the ratio of the number of events the model correctly recalls to the number of all correct events as shown in the equation 3.11.

$$\text{Recall } = \frac{TP}{TP+FN} \tag{3.11}$$

3.4.3 F1 Score

While useful, neither accuracy nor recall can fully evaluate a Machine Learning model. Separately these two metrics are useless:

- If the model predicts "positive" all the time, the recall will be high
- On the contrary, if the model never predicts "positive", the precision will be high

So we will have metrics that tell us that our model is efficient when it is actually more naive than intelligent.

Fortunately for us, a metric that allows us to combine precision and recall exists in the F1 Score. The F1 Score allows a good evaluation of the performance of our model. The higher your F1 Score, the better our model.

It is calculated as shown in the equation 3.12:

$$\text{F1 score } = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3.12}$$

3.4.4 Accuracy

An indicator of the model's performance across all classes is accuracy. When all classes are equally important, it is helpful. It is calculated as the number of accurate forecasts divided by the total number of predictions. In our model, it is the ratio of the correct patent segmentation number to the number of all the patent documents in this experiment. The equation 3.13 shows how we calculate accuracy:

$$\text{Accuracy } = \frac{TP+TN}{Total} \tag{3.13}$$

3.4.5 Penalty Score (Pk)

To overcome the fact that precision and recall are not sensitive to near misses, Beeferemen et al (1997) [DBL97] introduced the probabilistic error metric Pk. It the probability that two sentences drawn randomly from the corpus are correctly identified.
An approach based on sliding windows is used to determine Pk. Typically, the window size is designed to be half the average real segment number. In order to identify whether the two ends of the window are in the same or different segments in the ground truth segmentation, the algorithm slides the window. If there is a mismatch, the program increments a counter. The final score is determined by dividing the number of measurements by the penalty, which is scaled between 0 and 1. The value 0 indicates that the model predicts all the boundaries correctly. Consequently, the lower the score, the better. This score comes with many problems that one should be aware of them. The first problem is that it penalizes the FN more than FP. It is also sensitive to the segment size and doesn't consider if there are multiple boundaries in the same window.

3.4.6 WindowDiff (WD)

This score is introduced to solve the challenges that came with Pk score. This is calculated using a sliding window too. In this context, we only compare the number of boundaries present in the ground truth with the number predicted by the Topic Segmentation model for each position of the window of size k. The equation 3.14 shows how the WD is calculated:

$$WD(\text{ref, hyp}) = \frac{1}{N-k} \sum_{i=1}^{N-k} (|\, b(\text{ ref }_i,\ \text{ref }_{i+k}) - b(\text{ hyp}_i,\ \text{hyp }_{i+k})\,| > 0) \tag{3.14}$$

where b is the function that returns the number of boundaries between two positions in the text and N is the number of sentences in the text. A score of 0 indicates a perfect match between the reference boundaries and predicted ones.

3.4.7 Similarity Scores

There are four techniques to evaluate the similarity between two sentences. The first one is a knowledge-based similarity which helps in detecting the similarity between two concepts. The second one is statistical-based, which used the frequency of words in a text to build a matrix of evaluation. Next, is the string-based method which computes the distance between two strings. The last one is called language model-based similarity, which uses the part of speech tagging to build a tree to create a directed graph and the similarity is the minimum distance between the nodes.

Cosine Similarity
Cosine similarity is one of the string-based similarity measures. These are the most used ones among all the types of similarity scores. It is a similarity measure of two non-zero vectors of an inner product space, which finds the cosine of the angle between them. the equation 3.15

shows how it is calculated:

$$\text{Similarity}(A, B) = \frac{A.B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{3.15}$$

4 Exploratory Analysis

As we mentioned in the introduction section, researchers are gaining more interest in the patent field and they are focusing their efforts on using the most recent artificial intelligence tools to make patent analysis easier. In order to achieve this, administrations in this field are assisting researchers by making data as accessible as possible for them. The United States Patent and Trademark Office (USPTO) provides various datasets to encourage the development of the field of artificial intelligence for patents. being regarded as the most important patent office in the world, owing to the size of the American market, this office offers the most powerful dataset for patents to work with. It makes data available in different formats such as XML, JSON, and other data formats and presents the most recent data. In fact, it contains the full text of every patent award issued weekly (Tuesdays) from January 1, 1976, to the present (excluding images/drawings and reexaminations).
For our study, we will make use of the USPTO dataset that is publicly available as XML files. It is the patent grant full-text data with Embedded Tag Image File Format (TIFF) images, it is a subset of the patent grant full-text data. as shown in figure 4.1, we are using files that contain patents granted in the period from the beginning of January 2022 till the end of October 2022.

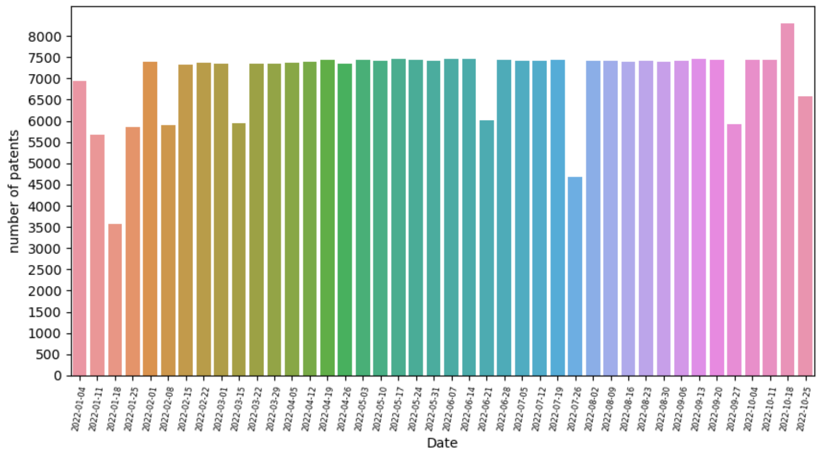
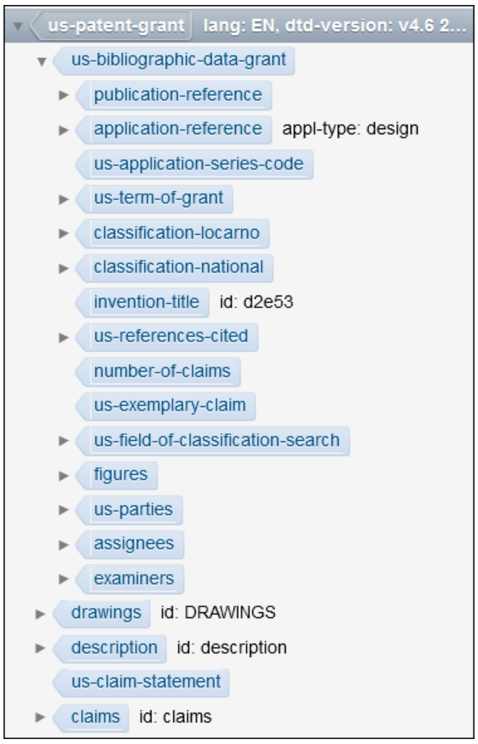


Figure 4.1: Weekly number of patents published in 2022

Each XML file in the dataset is composed of small independent sub-parts, Each sub-part contains detailed information about an invention. Figure 4.2 shows the outline of a sub-part of the dataset. Frequent disclosed details with every patent issuance are:

- **Claims:** they represent the legal part of the application. They provide the examiner the clear points to concentrate on during the review. All other parts come as support



```
▼ us-patent-grant   lang: EN, dtd-version: v4.6 2...
  ▼ us-bibliographic-data-grant
    ► publication-reference
    ► application-reference   appl-type: design
      us-application-series-code
    ► us-term-of-grant
    ► classification-locarno
    ► classification-national
      invention-title   id: d2e53
    ► us-references-cited
      number-of-claims
      us-exemplary-claim
    ► us-field-of-classification-search
    ► figures
    ► us-parties
    ► assignees
    ► examiners
  ► drawings   id: DRAWINGS
  ► description   id: description
    us-claim-statement
  ► claims   id: claims
```

Figure 4.2: Outline of a patent grant in USPTO dataset

for this part.

- **Description:** if there is any information that is not mentioned about the invention in the claim, it needs to be specified here.
- **References:** there are different references that a patent could have. For instance, the publication reference, contains the country of the application, the kind, and the date of the publication of the grant whereas the application reference contains the application type and the date of the application when the patent is in process.
- **Classification:** main and national classification in the country of the application.
- **Invention title:**
- **Grant id:** identification number of the grant

Our data consists of around 290000 samples of patents. As shown in figure 4.3 Each XML file is parsed, and the data is cleaned and collected to be saved as a CSV file. Figure 4.4 shows an example of the data in a CSV file. The same dataset is used for this thesis for the claim and document segmentation but a different approach for parsing will be used for each task. We will discuss relevant data for each type of segmentation in the following sections.

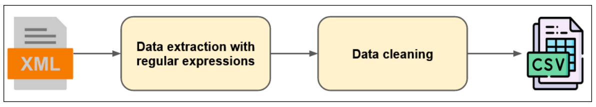
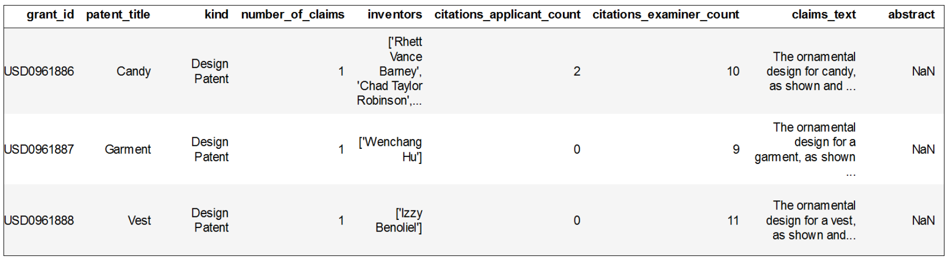


Figure 4.3: Steps to extract data



grant_id	patent_title	kind	number_of_claims	inventors	citations_applicant_count	citations_examiner_count	claims_text	abstract
USD0961886	Candy	Design Patent	1	['Rhett Vance Barney', 'Chad Taylor Robinson',...	2	10	The ornamental design for candy, as shown and ...	NaN
USD0961887	Garment	Design Patent	1	['Wenchang Hu']	0	9	The ornamental design for a garment, as shown ...	NaN
USD0961888	Vest	Design Patent	1	['Izzy Benoliel']	0	11	The ornamental design for a vest, as shown and...	NaN

Figure 4.4: Patent table

4.1 Claim Segmentation

For this task, the parts we need are the claim text and the number of claims. Data extraction is made using regular expressions. Since XML files have a specific outline for all of them so each patent claim can be extracted using the XML tag ¡claim-text¿ and the number of claims is found using the tag ¡number-of-claims¿. The claims are compiled using the following regular expression:

```
<claim-text>[s S<]*</claim-text>
```

After text extraction, the claim text needs to be cleaned. The following parts are cleaned:

- Claim text tags
- Line brakes
- Brackets
- Unnecessary slash symbols

As it is crucial to understand the specific aspect of the data. First, we grouped the samples by the number of claims, which revealed that samples with a single claim are the most frequent in the dataset as shown in figure 4.5.

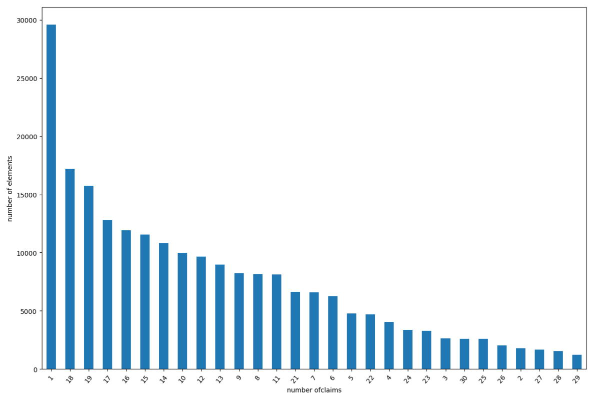


Figure 4.5: Number of samples for each number of claims

As our task is to segment the data, we need some analysis of the claim structure. After some research, we realized that most of the claims in the dataset match a specific structure using specific words. We found out that around 70% share the same punctuation structure and have commonly used words in them. For instance, the words, "comprising", and "wherein" are frequently used to describe inventions. We tried to see the most frequent words present in the claim texts. Figure 4.6 highlights the importance of some words as they appear in the claim texts frequently.

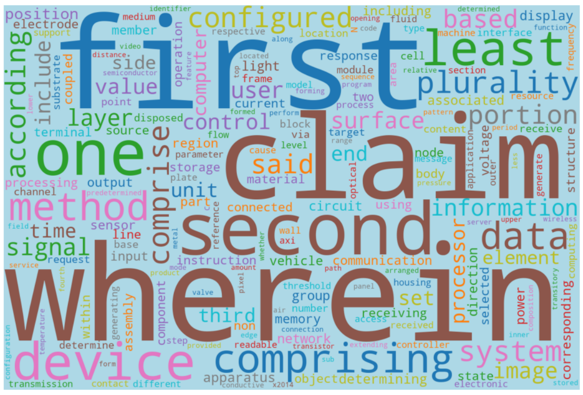


Figure 4.6: Words present in claim texts

Furthermore, based on our analysis and after consideration of how experts review the patent [USP19] and based on our analysis of the dataset. We realized that the patent has to meet the following rules:

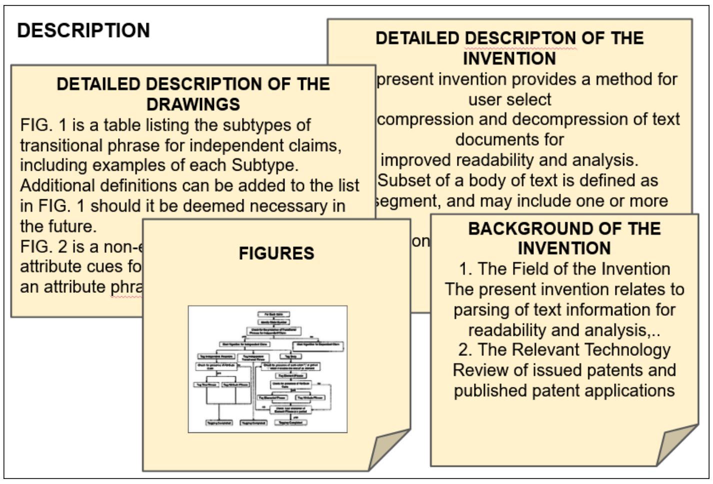
1) Each document can contain one or more claims.
2) Claims can depend on each other.
3) The claim is always one sentence.
4) The claim has a preamble and a body. They are separated by different expressions like "consisting of", "comprising", "characterized by", "having", "being composed of"
    - Comprising refers to an open list
    - Consisting of refers to a closed list
5) If there are parts of the system, they will be separated by a semicolon ";" and the last one will be separated by an "and"
6) If there is an explanation of the relation between parts it will begin with ", wherein"
7) A rule when creating a claim: use exact same term for the same concept
8) The fact that patent claims are always numbered can be used to tokenize claim text into separate claims.
9) Dependent claim usually refers to previous independent claims using certain terminologies ("as in claim #")
10) Use indefinite article "a" or "an" when introducing an element for the first time
11) Use definite article "the" or, "said" when referring back to the introduced element

4.2 Document Segmentation

In order to understand the claim details, the researcher needs to look for details in the reference document. The most important part that contains useful information about the claimed invention is the description part of the document. In the patent document, the description part includes different sections that need to be extracted too. Tags represented in the figure 4.7 are frequently found in the description section. They represent the essential aspects of the invention such as:

- **BRFSUM:** this tag stands for BRIEF SUMMARY. In this section, a description of at least two objectives of the inventions is requested and each objective has to be in one sentence.
- **Heading:** Other headings may be present under this tag such as related art where a brief description of the current state of the art is included.
- **DETDESC:** this tag stands for DETAILED DESCRIPTION. This section needs to provide as many details as possible. The more description provided, the more the invention is able to be claimed. The figures and flowcharts need to be tied to the text in this part and each element must have a number that corresponds to the element in the drawings.
- **Brief description of the drawings:** This is for the description of each drawing or flow chart.

An example of some of these sections is shown in figure 4.8


DESCRIPTION

DETAILED DESCRIPTON OF THE INVENTION
present invention provides a method for user select
compression and decompression of text documents for
improved readability and analysis.
Subset of a body of text is defined as
segment, and may include one or more
on

DETAILED DESCRIPTION OF THE DRAWINGS
FIG. 1 is a table listing the subtypes of transitional phrase for independent claims, including examples of each Subtype. Additional definitions can be added to the list in FIG. 1 should it be deemed necessary in the future.
FIG. 2 is a non-
attribute cues fo
an attribute phra

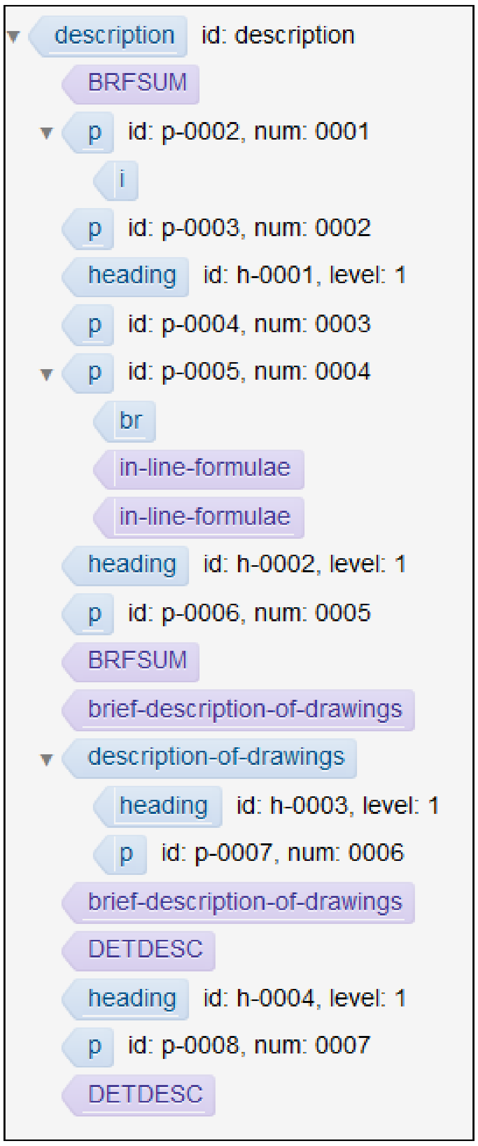
FIGURES

BACKGROUND OF THE INVENTION
1. The Field of the Invention
The present invention relates to parsing of text information for readability and analysis,..
2. The Relevant Technology
Review of issued patents and published patent applications

Figure 4.8: Sections of the description of patent document

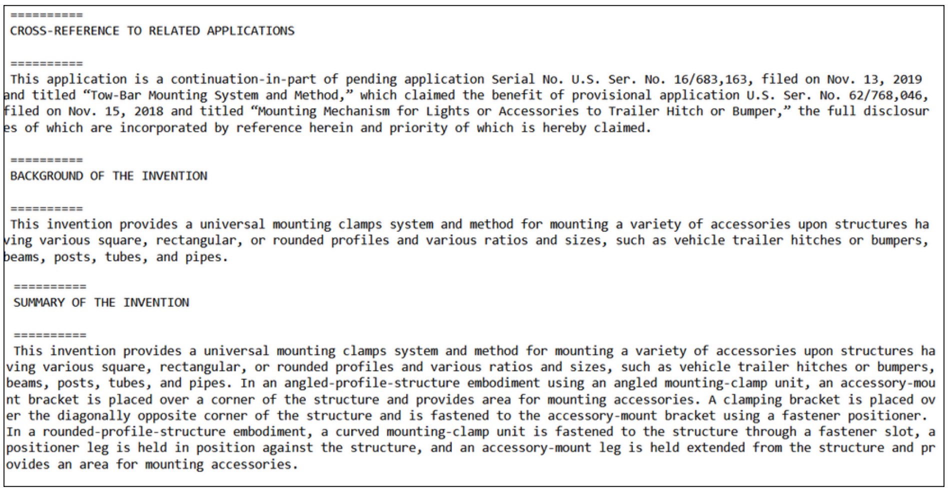
For the extraction of the parts of the document description, the following regular expression is used :

```
<description id="description">(.*?)</description>
```



```
▼ description   id: description
    BRFSUM
  ▼ p   id: p-0002, num: 0001
      i
    p   id: p-0003, num: 0002
    heading   id: h-0001, level: 1
    p   id: p-0004, num: 0003
  ▼ p   id: p-0005, num: 0004
      br
      in-line-formulae
      in-line-formulae
    heading   id: h-0002, level: 1
    p   id: p-0006, num: 0005
    BRFSUM
    brief-description-of-drawings
  ▼ description-of-drawings
      heading   id: h-0003, level: 1
      p   id: p-0007, num: 0006
    brief-description-of-drawings
    DETDESC
    heading   id: h-0004, level: 1
    p   id: p-0008, num: 0007
    DETDESC
```

Figure 4.7: Tags in the description of patent document



```
==========
CROSS-REFERENCE TO RELATED APPLICATIONS

==========
This application is a continuation-in-part of pending application Serial No. U.S. Ser. No. 16/683,163, filed on Nov. 13, 2019
and titled “Tow-Bar Mounting System and Method,” which claimed the benefit of provisional application U.S. Ser. No. 62/768,046,
filed on Nov. 15, 2018 and titled “Mounting Mechanism for Lights or Accessories to Trailer Hitch or Bumper,” the full disclosur
es of which are incorporated by reference herein and priority of which is hereby claimed.

==========
BACKGROUND OF THE INVENTION

==========
This invention provides a universal mounting clamps system and method for mounting a variety of accessories upon structures ha
ving various square, rectangular, or rounded profiles and various ratios and sizes, such as vehicle trailer hitches or bumpers,
beams, posts, tubes, and pipes.

==========
SUMMARY OF THE INVENTION

==========
This invention provides a universal mounting clamps system and method for mounting a variety of accessories upon structures ha
ving various square, rectangular, or rounded profiles and various ratios and sizes, such as vehicle trailer hitches or bumpers,
beams, posts, tubes, and pipes. In an angled-profile-structure embodiment using an angled mounting-clamp unit, an accessory-mou
nt bracket is placed over a corner of the structure and provides area for mounting accessories. A clamping bracket is placed ov
er the diagonally opposite corner of the structure and is fastened to the accessory-mount bracket using a fastener positioner.
In a rounded-profile-structure embodiment, a curved mounting-clamp unit is fastened to the structure through a fastener slot, a
positioner leg is held in position against the structure, and an accessory-mount leg is held extended from the structure and pr
ovides an area for mounting accessories.
```

Figure 4.9: Example of description text in a sample

Next, this part needs to be further analyzed using a simple and efficient library for parsing and creating XML data named The ElementTree XML API in python. This tool helped in extracting sections in the description that are described before. This task will help us prepare the data for the supervised document segmentation. The result of the extracted description is a single sequence in which sections are separated using the symbol ”=======”. Figure 4.9 shows an example of the description text of a sample in our dataset.

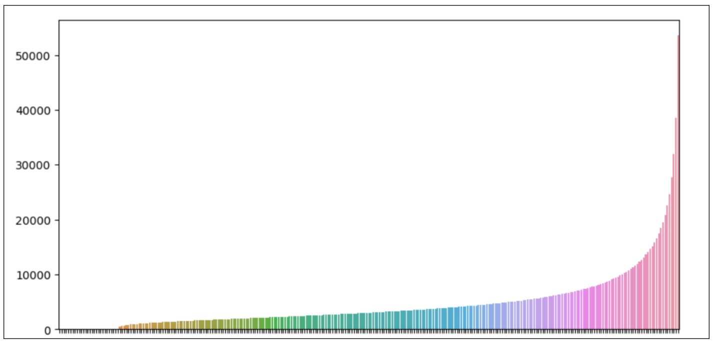


Figure 4.10: Number of words per sample in description text

Statistics about the number of words and the number of sections in each sample are also important to understand how data representation is going to be. We studied the number of words in each sample, figure 4.10 is a curve of the sorted number of words in each sample. Further, we split our samples by the symbol ”=======” and we tried to present the number of sections in each sample as shown in figure 4.11. Based on this, a summary of the mean, median, and standard deviation of word count and sentence count is in the table 4.1.

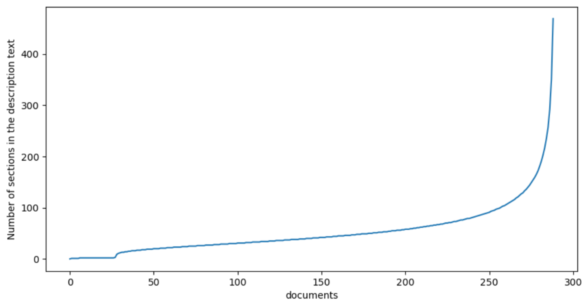


Figure 4.11: Number of sections per sample in description text

Number of	**Mean**	**Std**	**Median**
Words	4585	5714	3024
Sections	53	52	40

Table 4.1: Mean,standard deviation and median values of number of words and number of sections

5 Our Approach

We covered the extraction and cleaning of the data in the section above. We will go over the model's implementation process in this section. Our study we are going to go through three steps illustrated in figure 5.1. To help semantic search, we are concentrating on claim segmentation and document segmentation.

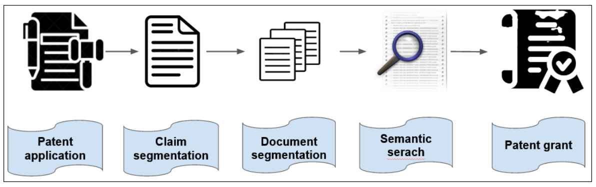


Figure 5.1: Steps of the study process

5.1 Implemetation: Claim Segmentation

5.1.1 Rule Based Method

As the patent has specific rules to follow when writing it, we made an analysis and extracted them to be our first step as not only a hard-coded segmentation method but a preparation for our supervised machine learning and deep learning methods.

First, we conducted a dependency analysis using StanfordCoreNlp. For example, figure 5.2 shows a dependency graph in a sample from the USPTO dataset. We found out that if more than one claim exists, it would be dependent on the first claim: For instance a specification about a part of this claim. So, after discussion with Mr. Christoph Hewel, the patent attorney, and consideration of our results we concluded that the segmentation will be only on the first claim because it's the most important one. This method is described in figure 5.3. We make

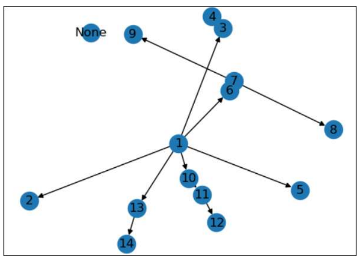


Figure 5.2: An example of the dependency graph for a sample in USPTO dataset

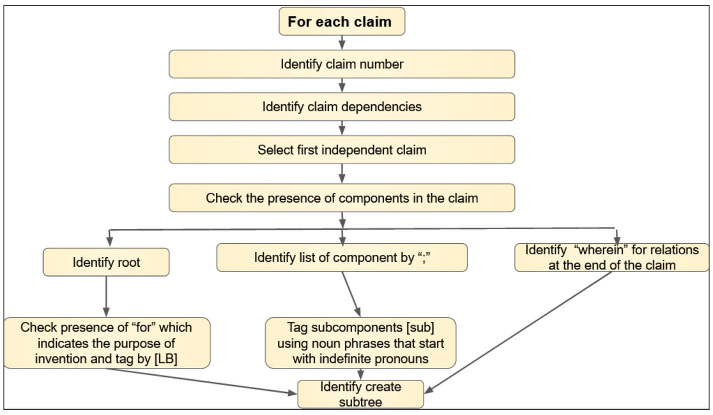

For each claim
Identify claim number
Identify claim dependencies
Select first independent claim
Check the presence of components in the claim
Identify root
Identify list of component by ";"
Identify "wherein" for relations at the end of the claim
Check presence of "for" which indicates the purpose of invention and tag by [LB]
Tag subcomponents [sub] using noun phrases that start with indefinite pronouns
Identify create subtree

Figure 5.3: Implemenation of our rule-based model

use of the USPTO dataset as XML files. We read all data as a pandas data frame. The data sample has one attribute that can be used which is the claim text. After further preprocessing of the sample and cleaning of the claim text, we start the segmentation. The claim will be segmented into 3 main parts:

- Root: This part describes the title and the purpose of the invention.
- Elements: The claimed system have elements described in the claim.
- Relations: These are specifications about the system or relations between the elements.

Our main steps for this method are as follows:

1) Identify the claim number
2) Identify if the claim matches a specific punctuation scheme:
    - A comma "," separates the preamble from the transitional phrase
    - A colon ":" separates the transitional phrase from the body
    - The small paragraphs that define and describe the logical elements are separated with semi-colons ";"

   From our analysis, we found out that 70 % of the data matches the previous scheme
3) In relations we should identify if it's about sub-elements or not
4) Use indefinite article "a" or "an" when introducing an element for the first time
5) Use definite article "the" or, "said" when referring back to the introduced element
6) Use libraries to look for noun phrases and instead of looking for comprising we look for any synonyms of it

5.1.2 Machine Learning Methods

The first idea for out task with machine learning tools is simple. One way to think about the segmentation is to try to make a model that remembers every word in the dataset and its class. We deal with the problem as it is a problem of binary classification that takes all

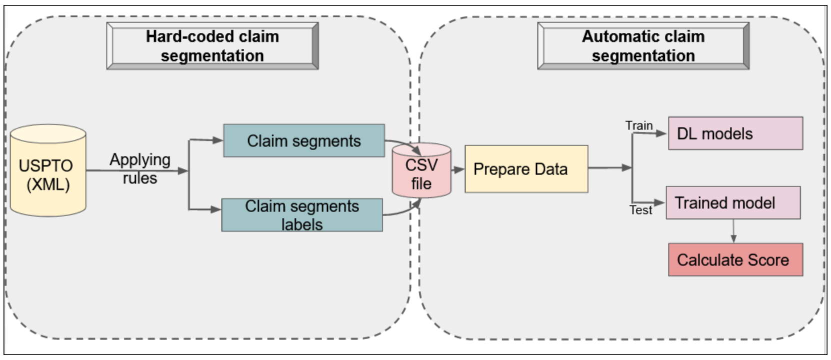


Figure 5.4: General implementation of Deep Learning models

the words that sentences in our dataset contain and try to learn which words end a segment and which ones don't. Based on this idea, we made use of random forest classifier as it is a fast classifier and can work with large datasets. We created a list that contains all the words that are in all sentences in our dataset and their tags. We used RandomForestClassifier from sklearn library in python with different numbers of estimator for our experiments.
For better information about the features that are necessary for the decision . We added memory and contextual data to improve our basic features. We make an effort to provide further details about the feature, such as if the feature is a number or a title. The results of the basic model and its enhancement are discussed in the following chapters.

5.1.3 Deep Learning Based Methods

After doing some research on text segmentation, we found out that NER techniques are useful in our methodology. In fact, claims are always one phrase, it would be better to approach the segmentation as a classification technique for words that make up the sentence. On the basis of this strategy, Our supervised learning models need some special data preparation to work. The general implementation of the models is presented in figure 5.4.
Considering NER techniques, we can use multiple deep-learning algorithms. To reach our goal, we make use of CRF, LSTM, Bi-LSTM, and BERT. The reason why we are using these methods is their bidirectional aspect and their well-known performance in NLP. We are using data presented as a CSV file extracted from USPTO XML files from our previous rule-based method. The sample attributes that are needed for these models are segmentation of the claim and binary values of this segmentation stating if every part in the segmentation ends a segment. these two attributes are lists of the same length. Before feeding the data to the model we preprocess and remove irrelevant segments. After that, the data is mapped into numeric values which means that each word in the input sequence will have a specific numerical value. Once the model is trained on the training dataset, it will predict the series of output states saying if each input value can end a segment or not. The output value will be compared to the true value and will help us calculate scores for our evaluation.

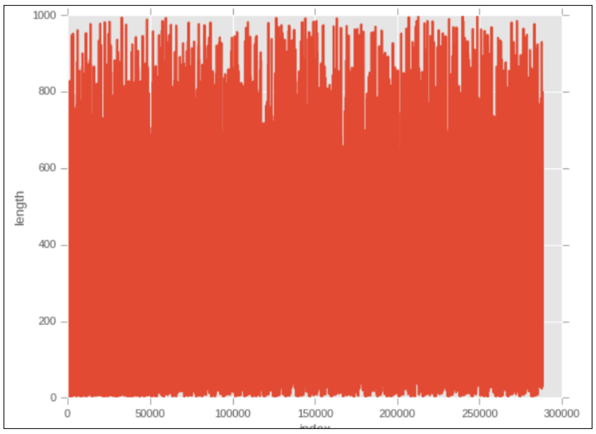
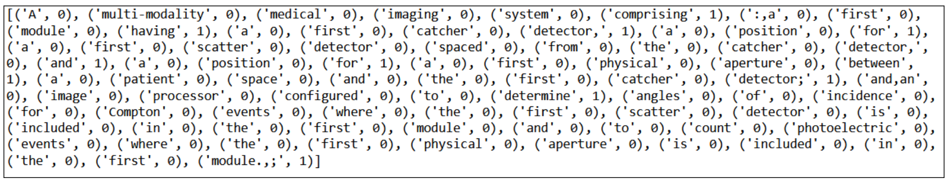


Figure 5.6: Layered architecture for BiLSTM model

5.1.3.1 Bi-LSTM

To prepare the dataset, we start by reading the sequences and their labels and mapping them together. The figure 5.5 shows how a sentence looks like :
We must utilize equal-length input sequences when using neural networks (at least with Keras; this is not a theoretical requirement). Therefore, we checked how long our sentences are using ggplot package from matplotlib library, (a popular plotting package for python, and based on the result in the figure 5.6 we will lengthen our phrases to 200 words which is the mean of the length of the sentence. Then, we map the sentences to a sequence of numbers and then pad the sequence to transform the list of sentences into a 2D array. the list of labels has to be padded too. We used the post padding for both of the previous lists with values n_words -1 and 0 respectively where n_words is the number of unique words in all sentences.



```
[('A', 0), ('multi-modality', 0), ('medical', 0), ('imaging', 0), ('system', 0), ('comprising', 1), (':,a', 0), ('first', 0),
('module', 0), ('having', 1), ('a', 0), ('first', 0), ('catcher', 0), ('detector,', 1), ('a', 0), ('position', 0), ('for', 1),
('a', 0), ('first', 0), ('scatter', 0), ('detector', 0), ('spaced', 0), ('from', 0), ('the', 0), ('catcher', 0), ('detector,',
0), ('and', 1), ('a', 0), ('position', 0), ('for', 1), ('a', 0), ('first', 0), ('physical', 0), ('aperture', 0), ('between',
1), ('a', 0), ('patient', 0), ('space', 0), ('and', 0), ('the', 0), ('first', 0), ('catcher', 0), ('detector;', 1), ('and,an',
0), ('image', 0), ('processor', 0), ('configured', 0), ('to', 0), ('determine', 1), ('angles', 0), ('of', 0), ('incidence', 0),
('for', 0), ('Compton', 0), ('events', 0), ('where', 0), ('the', 0), ('first', 0), ('scatter', 0), ('detector', 0), ('is', 0),
('included', 0), ('in', 0), ('the', 0), ('first', 0), ('module', 0), ('and', 0), ('to', 0), ('count', 0), ('photoelectric', 0),
('events', 0), ('where', 0), ('the', 0), ('first', 0), ('physical', 0), ('aperture', 0), ('is', 0), ('included', 0), ('in', 0),
('the', 0), ('first', 0), ('module.,;', 1)]
```

Figure 5.5: A sentence as input to the model

The model architecture is described in figure 5.7: The first layer takes an input of size max_len where max_len is the mean length of sentences. The data is embedded after that with a 50-dimension embedding. For an effective technique for regularization and preventing the co-adaptation of neurons, we added a Dropout layer with a probability parameter set to 0.1. The fourth layer is our BiLSTM model with a number of units set to 100 as the preliminary value. As the output of LSTM is not softmax, so the dimensionality of this output is equal to the number of units, which is not the dimensionality of your desired target, it's common to add a Dense layer as a softmax output layer.

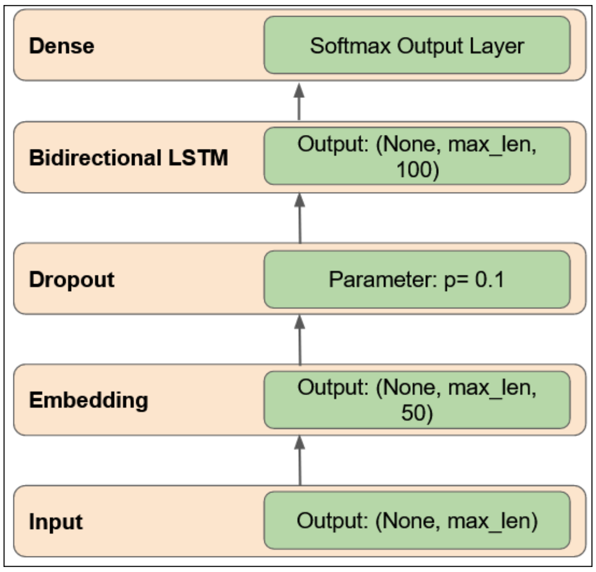


Figure 5.7: Length of the input sentences

5.1.3.2 CRF

We designate x=(x_1,...,x_m)as the input sequence, and s=(s_1,..., s_m) as the series of output states, or the words in a claim phrase. We simulate conditional probability in conditional random fields.

$$p(s_1,\ldots,s_m \mid x_1,\ldots,x_m)$$

To accomplish this, we define a feature map:

$$\phi(x_1,..,x_m,s_1,..s_m) \in R^d \tag{5.1}$$

that converts a full input sequence (x) and full state sequence (s) into a d-dimensional feature vector. The probability can then be represented as a log-linear model using the parameter vector w as in the equation 5.2

$$p(s \mid x,\omega) = \frac{\exp(\omega \cdot \phi(x,s))}{\sum_{s'} \exp\left(\omega \cdot \phi\left(x,s'\right)\right)} \tag{5.2}$$

where s' covers all potential output iterations. We presumptively have a collection of n labeled examples $(x_i,s_i)_{i=1..n}$ for the estimation of w. The regularized log-likelihood function L is now defined as the equation 5.3:

$$L(\omega) = \sum_{i=1}^{n} \log P\left(s^i \mid x^i,\omega\right) - \frac{\lambda_2}{2}|\omega|_2^2 - \frac{\lambda_1}{2}|\omega|_1 \tag{5.3}$$

The parameter vector must be small in the corresponding norm due to the terms $\frac{\lambda_2}{2}|\omega|_2^2 - \frac{\lambda_1}{2}|\omega|_1$. Regularization, which penalizes model complexity, is what is happening here. It is possible to enforce more or less regularization using parameters λ_1 and λ_2. The parameter vector $w*$ is then estimated as shown in equation 5.4

$$w^* = \underset{w \in \mathbb{R}^d}{} \text{argmax}_w \, L(w) \tag{5.4}$$

After estimating $w*$ we estimate s using the equation 5.5:

$$w^* = \text{argmax}_s \, P\left(s \mid x, \omega*\right) \tag{5.5}$$

For this model, first, we load the dataset as a data frame. Then, we count the number of unique words in the sentences to prepare to map the sentence to a sequence of numbers. So, we have 288872 sentences with 456405 unique words. After that, we create some features to prepare the dataset. For each word, we add the words that are neighboring him as features and if this word is in capital letters, This way we help the model get more insights about each word in the sentence. After fitting the model, We can now start the algorithm. We employ the sklearn-crfsuite's CRF implementation. Here we are using the L-BFGS training algorithm (it is the default) with Elastic Net (L1 + L2) regularization. To see if the model works, we perform a 5-fold cross-validation and use the scikit-learn classification report to evaluate the segmenter. The results of the CRF implementation are discussed in the next chapter.
To make results better, we thought it would be a good practice to combine some models and use them as a system for sequence segmentation. Our first combination is the Bi-LSTM-CRF model whose implementation is described in the next section.

5.1.3.3 BiLSTM-CRF

For this section, we are using the CRF model specification. in the section above we defined a feature map ϕ using an input sequence x and their output states s. We modeled the probability as a log-linear model in the equation 5.2. The expression $w\phi(x, s)$ can be interpreted as a score for crf, it can measure how well the state sequence fits the specified input sequence. For Now, the plan is to use a non-linear neural network in place of the linear scoring function. Thus, we establish the score of our model as in 5.6 where W $_{\mathrm{s_{i-1},s_i}}$ is the weight vector corresponding to the transition from $\mathrm{s_{i-1}}$ to $\mathrm{s_i}$ and b is the bias.:

$$score_{lstm-crf}(x, s) = \sum_{i=0}^{n} w_{s_{i=1},s_i} \cdot \text{LSTM}(x)_i + b_{s_{i-1},s_i} \tag{5.6}$$

Using this function we can optimize the conditional probability. For this implementation, we are using the package keras-contrib. Data preparation for the model is the same as the previous one. The sentences are converted to a series of numbers, which is then padded. We added one to the word index in order to use zero as a padding value. This is done because we wish to disregard inputs with a value of 0 using the mask zero option of the embedding layer. The labels are changed to categorical values. After that, the data is split into 80% training data and 20% test data. Figure 5.8 shows a layered architecture of our model. We fit the LSTM-CRF model with a 20-dimension embedding layer. the second layer is a variational Bi-LSTM. We add a dense layer and finally the CRF layer. Using a layered architecture showed some good results that we are going to discuss in the evaluation section. We tried to

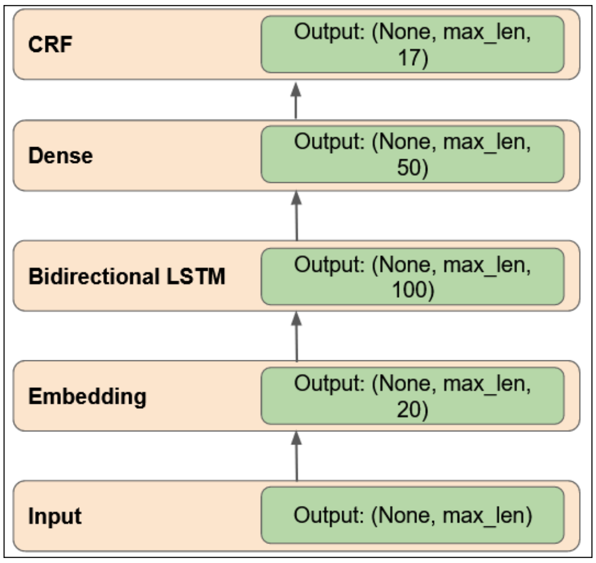

CRF
Output: (None, max_len, 17)
Dense
Output: (None, max_len, 50)
Bidirectional LSTM
Output: (None, max_len, 100)
Embedding
Output: (None, max_len, 20)
Input
Output: (None, max_len)

Figure 5.8: A layered architecture of biLSTM-CRF model

optimize the results by changing the parameters of the model and after that we are going to try a totally different approach. In the next part, we are going to define a special pre-trained BERT model used to get better results than our last layered approaches.

5.1.3.4 BERT for Token Classification

The rise of pretraining and finetuning in natural language processing made us think about using BERT for better performance and better results. For this, we are using the transformer's package in python to be able to access different types of pre-trained BERT models. For performance issues, we need to fix some configurations before training the model. The packages in python need to have compatible versions to work. For this, we installed versions 2.6.0, and 1.2.0 respectively for transformers and PyTorch packages. We will also limit the sequence length to 200 tokens and we will use a batch size of 8.
A defined vocabulary and a pre-trained tokenizer are included with the Bert implementation. We load the pre-trained model "bert-base-cased" and we tokenize all the sentences. The tokenizer is based on a word-piece tokenizer that tokenizes the sentence to words that appear in the vocabulary. Note that, we need to take care of mapping the labels with tokens too before using them for the model. After tokenizing the input we need to pad the sequence to the same length.
Similar to masking in Keras, the Bert model supports a concept called attention mask. In order to ignore the padded components in the sequences, we now design the mask. We divided the dataset and used 10% of it for validation. We must convert the dataset to torch tensors because we're using PyTorch. The data loaders must be defined as the final step. When training, we use the RandomSampler to shuffle the data, and when testing, we use the SequentialSampler to simply pass the data in order.
In our approach, we use a special class from BERT for token levels predictions. The transformers package provides BertForTokenClassification that comes as a BertModel with a token-level

classifier, a linear layer that takes accepts the last hidden state of the sequence as input.
To start the fine-tuning process we fixed some configurations for the optimizer. In fact, we used AdamW, a commonly used optimizer, We also add some weight_decay as regularization to the main weight matrices. In order to linearly lower the learning rate throughout the epochs, we additionally include a scheduler.
After using the "bert-base-cased" implementation. The next step is to use BERT for our specific task. We made use of a pre-trained model on 100 million US patents. This model has the advantage that it understands the context of the legal and technical text in the patent. The model name is "anferico/bert-for-patents" and its implementation is available with the transformers library in python.

5.2 Implementation: Reference Document Segmentation

5.2.1 Unsupervised Methods

Based on the idea of unsupervised text segmentation Based on Latent Dirichlet Allocation (LDA) and topic tiling presented by Mirela Ostrek, Luka Dulci [OD16] the Topic Tiling algorithm predicts segment boundaries based on similarities between adjacent sentences. The feature that is relevant for us in this method is the description of the patent. The preprocessing of the dataset consists of three steps:

1) *Sentence segmentation:* Because each sentence in the description text is separated by a new line character, segmenting sentences was a simple process. Moreover, we had to eliminate unnecessary sentences that were empty or contained nothing but stop words and other meaningless tokens.
2) *Sentence tokenization:* sentences are tokenized using the typical nltk tokenizer. The tokens are after that filtered from the ones that only have English stop words.
3) *Sentence stemming:* this step is done using nltk Porter Stemmer, at this step different variants of a word are traced back to its root form. Then, we vectorize the tokens with scikit-learn CountVectorizer.

We applied the Topic Tiling technique to the given description text in order to split it into semantically coherent pieces. Sentences, paragraphs, and even entire text can be passed to LDA with the aim of fitting our model so that it can successfully process the set of unobserved texts. We use probability distributions across topics for each sentence in the description text generated by the LDA, called topic vectors as inputs for the Text Tiling algorithm. This algorithm uses topic distribution over words in a sentence and Topic Tiling performs the segmentation in linear time and thus is computationally less expensive than other LDA-based segmentation methods. The similarity of adjacent topic vectors is evaluated using cosine similarity in our approach. We used this type of coherence score because it is not too expensive in terms of performance and efficiency for this type of task. Several parameters have to be fixed for this algorithm. For instance, for LDA we need to fine-tune the α, β , and the number of topics K parameters.
Besides Accuracy, precision, recall, and F1 score, this approach will be evaluated based on Pk and WD measures. These last two evaluation methods use a sliding window that needs to be fixed too, this will be more discussed in the Evaluation section.

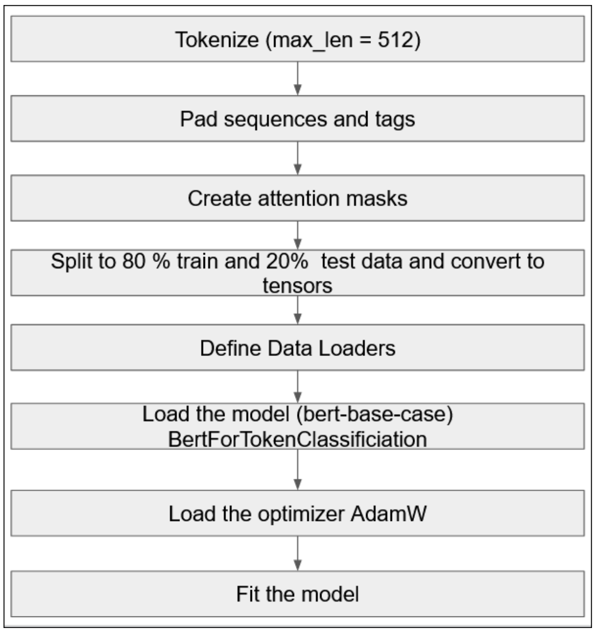


Figure 5.9: Steps of Bert for document segmentation

5.2.2 Supervised Methods

As mentioned in the data exploratory chapter, when extracting the document description, we could segment it using the tags of the XML file. To prepare the data for this model we used the tags used in the XML file in the USPTO dataset. We segmented the description text mainly using the headings, and the references of the figures. For this task, we made use of the BERT model to segment the document in a supervised way using deep learning. The input of this model is a list of sentences and the output is a list with binary values stating if each sentence ends a segment or not. The input should be padded as the model accepts a fixed length. The tokenizer used for this task is a word piece tokenizer that comes with the implementation of the BERT model in the transformers library. The number of tokens used for this model is 512. The batch size is 16. After preparing the attention masks and splitting the data into 20% test data and 80% training data. The data is converted to PyTorch tensors. After that, we fine-tune the model over the epochs. The steps are shown in figure **??**

6 Results

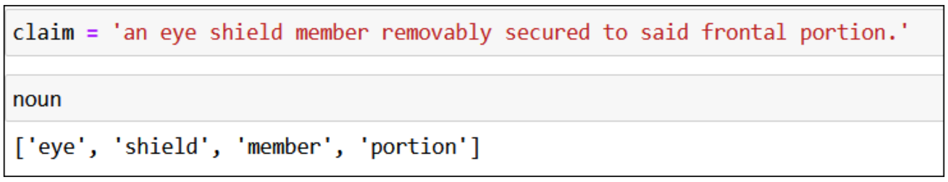
In the previous section, we went through our approach. We discussed the steps of each model for claim segmentation, document segmentation, and semantic search and how it is implemented in our work. In this chapter, we evaluate the performance of each model and try to compare its results with different settings. This will enable us to assess the effectiveness of our methods and how successfully they may be applied to our research.

6.1 Claim Segmentation

For the segmentation of the claim task, we have considered three types of techniques. We employed a rule-based technique for a hard-coded segmentation to be able to train supervised models for the task, a random forest model for the ML technique, and Bi-LSTM, CRF, and BERT models, as well as combinations of these models, for the DL technique. For the evaluation, f1-Score, accuracy, precision, and recall are used.

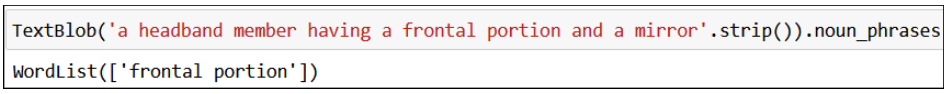
6.1.1 Rule-based Model Results

The output of this model is a list of segments that the claim contains. It helps detect each independent part of the first claim text using some techniques. This technique relies on regular expressions and looking for noun phrases to detect sub-elements. We tried to analyze the results and try to detect how this technique performed. We found some problems with segmentation. For instance, to detect the noun phrases we used different types of nlp libraries such as wordnet, TextBlob, and nlp. We detected some errors with wordnet when the noun is composed of two words. An example of that can be seen in figure 6.1. Using TextBlob for some sentences made us realize some errors too like in figure 6.2 where the model fails to detect some noun phrases. The nlp library seems to be reliable so we made use of it for final segmentation. There are examples where the punctuation doesn't work to segment the claim such as in figure 6.3. Based on these errors we tried to improve our results and add more rules for the segmentation.



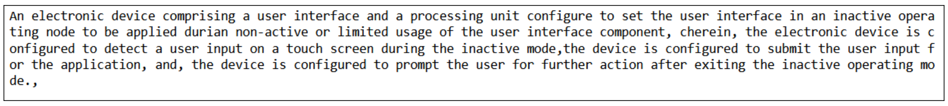
```
claim = 'an eye shield member removably secured to said frontal portion.'

noun

['eye', 'shield', 'member', 'portion']
```

Figure 6.1: Example of noun phrase detection using Wordnet



```
TextBlob('a headband member having a frontal portion and a mirror'.strip()).noun_phrases
WordList(['frontal portion'])
```

Figure 6.2: Example of noun phrase detection using TextBlob



```
An electronic device comprising a user interface and a processing unit configure to set the user interface in an inactive opera
ting node to be applied durian non-active or limited usage of the user interface component, cherein, the electronic device is c
onfigured to detect a user input on a touch screen during the inactive mode,the device is configured to submit the user input f
or the application, and, the device is configured to prompt the user for further action after exiting the inactive operating mo
de.,
```

Figure 6.3: Example of claim

6.1.2 ML Model Results

For this part, all words from claim sentences that exist in our USPTO dataset are concatenated into a single list. The same is true for the tags; we produced a list of all word tags. Our output is the same length as the input, which is 52220836 words. As a start, We built a memory tagger class of estimators that inherits from BaseEstimator and TransformerMixin from sklearn library to build a vocabulary where it should remember every tag for every model that it sees. We set a value of 0 if the model gets an unknown word. Using the cross-validation with a splitter parameter cv equals 5. The first results are shown in table 6.1. The results show a null recall value for class 1 and the precision for this class is low too. The accuracy of the model is 0.88% which is not bad as a baseline but the values of precision and recall were 50% which is not convenient.
To enhance the predictions, We used a random forest classifier with a number of estimators equal to 5. But the result stayed the same. So, our next step was to make the model know a bit about the context of the words and to combine memory and context to try to get better results. The table 6.2 shows the result. Although the results are not promising, we noticed that context made a difference when evaluating the model. A better improvement for the model was to combine the two previous ideas, a model that has both memory and context. We tried using the system with the same parameters and we got the results shown in 6.3. This idea had a noticeable impact especially on the evaluation metric for class 1. Both the precision and the recall for this class started to get better but were still low. For the following experiments, we increase the batch size and evaluate its effect. In the fourth model with batch size 32, precision and recall are improving but accuracy is decreasing. The same effect on accuracy with increasing the batch size but in the fourth model recall is affected too.

	Precision	Recall	F-1 Score	Support
0	0.88	1.00	0.93	45733739
1	0.12	0.00	0.00	6487097
Accuracy			0.88	52220836
Macro avg	0.50	0.50	0.47	52220836
Weighted avg	0.78	0.88	0.82	52220836

Table 6.1: Evaluation metrics of Memory Tagger model

	Precision	**Recall**	**F-1 Score**	**Support**
0	0.88	1.00	0.93	45733739
1	0.14	0.00	0.00	6487097
Accuracy			0.88	52220836
Macro avg	0.51	0.50	0.47	52220836
Weighted avg	0.78	0.88	0.82	52220836

Table 6.2: Evaluation metrics of Random Forest model

	Precision	**Recall**	**F-1 Score**	**Support**
0	0.89	0.98	0.93	45733739
1	0.15	0.12	0.13	6487097
Accuracy			0.86	52220836
Macro avg	0.52	0.55	0.53	52220836
Weighted avg	0.79	0.86	0.83	52220836

Table 6.3: Evaluation metrics of the context-based Random Forest model

Model	**c1**	**c2**	**max iterations**
1	0.1	0.1	100
2	10	0.1	100

Table 6.4: Parameters of CRF model

6.1.3 DL Models Results

DL techniques are the most powerful tools in NLP for their ability to understand the relationship between words in the text. They are also useful in this field due to their ability to work on large datasets. In the following sections, we evaluate the results from different DL models as well as different combinations of them.

6.1.3.1 CRF Results

Coming to the second model. To decide on the parameters to use, we conducted two experiments with the CRF model. First, the coefficients c1 and c2 for the L1 and L2 regularization, are fixed to the value of 0.1. This model looks like a good start. The results are better than the ML model. As can be seen in the table 6.5 the values of the evaluation metrics are interesting, but for class 1 the precision is lower than class 0, which means that predictions for class 0 are more reliable than predictions for class 1. As a trial to improve the model with regularization, we changed the parameter c1 to 10 as a value. Better performance can be detected in the table 6.6 for class 1, while no change for class 0. The summary of the parameters used for the CRF model can be seen in the table 6.4.

	Precision	Recall	F1-Score	Support
0	0.97	0.98	0.97	1425662
1	0.84	0.74	0.78	201074
Accuracy			0.95	1626736
Macro avg	0.90	0.86	0.88	1626736

Table 6.5: Results of model 1 using CRF

	Precision	Recall	F1-Score	Support
0	0.99	0.98	0.99	1425662
1	0.85	0.74	0.79	201074
Accuracy			0.97	1626736
Macro avg	0.93	0.94	0.94	1626736

Table 6.6: Results of model 2 using CRF

6.1.3.2 Bi-LSTM Results

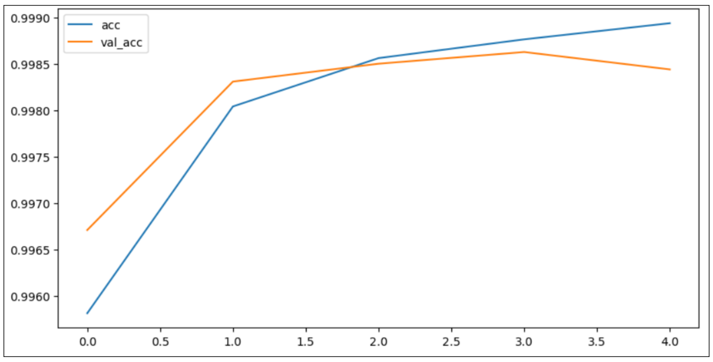


Figure 6.4: Accuracy of Bi-LSTM model

To check the results of the implementation of the Bi-LSTM model, Our experiments were based on the models described in the table 6.7. In the first model, 3000 samples are used to evaluate it. The batch size is fixed to 32, the optimizer used is rmsprop, and the model is trained on 5 epochs. Figure 6.4 describes the accuracy of the training and validation sets among epochs. The figure shows that the value of accuracy is high, it is around 99% but the validation accuracy starts to decrease starting from epoch 3. This indicates that the model is overfitting. For this reason, in the second experiment, the number of samples is set to 50000, the batch size is changed to 8 and the Adam optimizer is used. The evaluation results are described in the table 6.8. The accuracy of this model was 0.17 which is a very low value. The third attempt to optimize the model is to use all the samples of the dataset described in the exploratory analysis chapter. The rmsprop optimizer is used again for this experiment, the batch size is the same as the previous one. This model improved the accuracy to 60% as well as the values of precision and recall.

Model	number of samples	batch size	optimizer	number of epochs
1	3000	32	rmsprop	5
2	50000	8	Adam	1
3	290000	8	rmsprop	1
4	290000	32	rmsprop	1
5	290000	64	rmsprop	1

Table 6.7: Bi-LSTM parameters for the experiments

Model	Precision	Recall	F-1 Score	Accuracy
2	0.84	0.74	0.79	0.17
3	0.92	0.91	0.91	0.60
4	0.95	0.94	0.93	0.50
5	0.96	0.74	0.81	0.19

Table 6.8: Evaluation results of Bi-LSTM model

Precision	Recall	F-1 Score	Accuracy
0.87	0.91	0.88	0.88

Table 6.9: Evaluation results of Bi-LSTM - CRF model

6.1.3.3 CRF - Bi-LSTM Results

For this model, the experimental setup is described in the methodology chapter. A single experiment is conducted and the results are described in table 6.9. With this model, the accuracy achieved is 88% and the F1 score was 88%.

6.1.3.4 BERT Results

BERT is always known for its performance when it comes to language understanding. For this reason, it is the most important model of our study. For checking the performance of the model, we used different combinations of parameters to make the results better. The summary of parameters used for this model is shown in the table 6.10. For each experiment, we changed the learning rate, the number of epochs, batch size, or the maximum length of the input to evaluate the models and improve the efficiency of settings to segment the claims. The results are shown in the table 6.11. For the second model, figure 6.5 describes the learning curve through the epochs. We noticed that the model is underfitting as the validation loss is starting to be higher than the training loss. So we decided to reduce the number of epochs in the following experiments. The highest recall value is achieved in experiment 2,3,4,5 but the problem is that the precision is low. We can notice that increasing the recall value affects negatively the precision value and the opposite is true such as in model 6. So, in the experiments, we need to concentrate on finding the two values that maximize the F1 score as a trade-off between them. The best model result in terms of F1 score can be realized by model 5. An accuracy of 0.83 can be achieved with the BERT model.

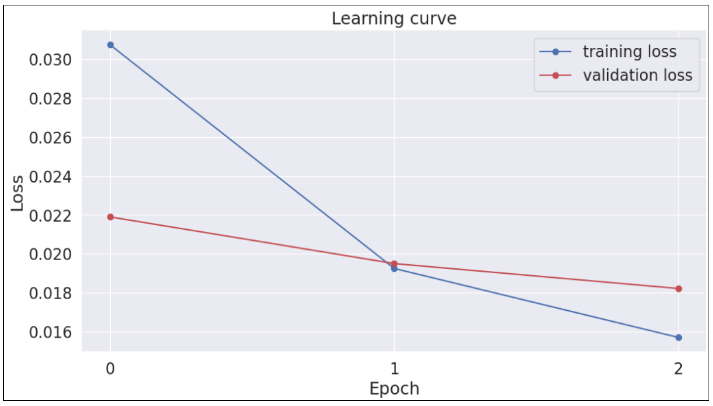


Figure 6.5: Learning curve of BERT model

Model	Number of epochs	Batch size	Learning rate
1	3	32	5e-5
2	3	64	5e-5
3	2	16	5e-5
4	3	32	4e-5
5	3	32	3e-5
6	1	16	3e-5

Table 6.10: Summary of BERT parameters for the experiments for claim segmentation

Model	Precision	Recall	F1 Score	Accuracy
1	0.49	0.93	0.83	0.78
2	0.45	0.98	0.79	0.79
3	0.45	0.98	0.79	0.79
4	0.45	0.98	0.79	0.79
5	0.46	0.98	0.80	0.79
6	0.50	0.60	0.54	0.77

Table 6.11: Evaluation results of BERT model

Coming to the BERT for patents. We made the following settings for this model as they appear to give good results with our baseline BERT model:

- Optimizer: Adam
- Learning rate: 5e-5
- Batch size: 32
- Number of epochs: 2

The evaluation of this model on our dataset can be described in the table 6.12. This model has an accuracy value of 96% and an F1 Score value of 82%. This model gave us good performance in terms of accuracy and F-1 score.

Precision	Recall	F1 Score	Accuracy
0.78	0.88	0.82	0.96

Table 6.12: Evaluation results of BERT model for patents

6.2 Reference Document Segmentation

Coming to the document segmentation task. Our models can be grouped into two groups: supervised and unsupervised models. For each group, we conducted experiments to evaluate their performance with different parameters. Results are described in the following sections.

6.2.1 Unsupervised Model Results

For this model, we made use of the parameters described in the table 6.13. This model is trained and tested on 10000 description documents. For the first experiment, we considered the number of topics to be 60 topics. The parameters document topic prior α and word document prior β of LDA are fixed to 0.2 and 0.01 respectively, the multiplier parameter of topic tiling m used is 0.1 and the maximum of iterations is 100. The results were not promising the value of PK was equal to 0.55 which is not reliable. Figure 6.6 shows the coherence score of a sample with Pk= 0.69. The blue color represents cosine similarity, the green lines are for correct boundaries, the red lines are for false boundaries, and the dashed green lines are for FN boundaries. From the figure, the number of FN boundaries is significant which results in a low recall. For this sample the value of accuracy is 0.22, recall is equal to 0.18 and precision is equal to 0.82. To optimize the model, the number of topics is increased to 150 topics, α and β of LDA are fixed to 0.1 and 0.01 respectively and m is set to 0.2. The results were better using this combination, the P_k was 0.52 which is better than the last model but still not reliable for our task.

Model	Number of topics	Max iterations	α	β	m
1	60	100	0.2	0.01	0.1
2	150	100	0.1	0.01	0.2

Table 6.13: Summary of TopicTiling parameters for the experiments

6.2.2 Supervised Models Results

The model used for this task is the BERT model. Different settings are used to compare results for this model. the maximum length of the input here is set to the maximum that our model can accept which is 512 and the optimizer used is an Adam optimizer with different learning rates. Different batch sizes are used for the evaluation too. The summary of the settings of the experiments is shown in table 6.14. The first experiment as described in the model was using a learning rate equal to 5e-5 and 3 epochs for training. The batch size was fixed to 16. Figure 6.7 shows the learning curve of this model. From the second epoch, the

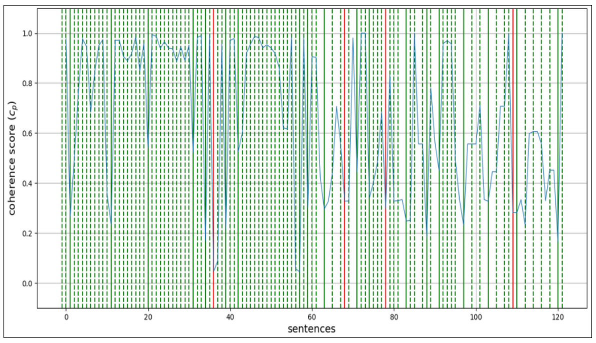


Figure 6.6: Coherence score of a sample

model is starting to overfit as the validation loss is starting to be higher than the training loss and it is starting to increase. That's why for the following experiments we decided to lower the learning rate and the number of epochs. The results of these experiments are described in the table 6.15. The trade-off between optimizing the precision and the recall can be seen in these results too. The best F1 score achieved is 81% with an accuracy of 90%. Further discussion of the results will be in the following chapter.

Model	Number of epochs	Batch size	Learning rate
1	3	16	5e-5
2	1	8	5e-5
3	1	16	3e-5
4	1	1	3e-5

Table 6.14: Summary of BERT parameters for the experiments for document segmentation

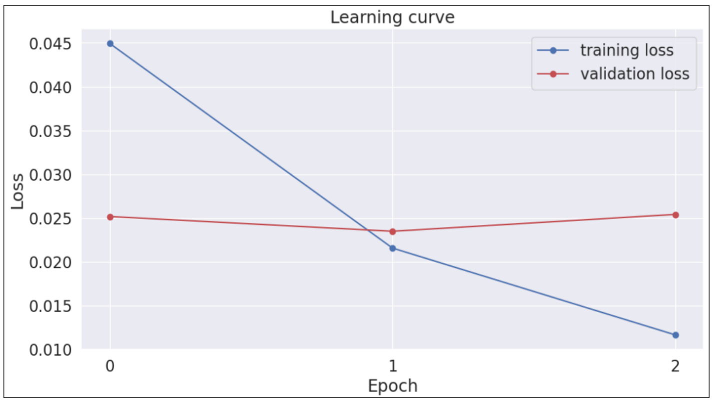


Figure 6.7: Learning curve of BERT model for document segmentation

Model	**Precision**	**Recall**	**F1 Score**	**Accuracy**
1	0.49	0.91	0.81	0.90
2	0.47	0.96	0.81	0.92
3	0.38	0.95	0.64	0.91
4	0.38	0.99	0.64	0.91

Table 6.15: Evaluation results of BERT model for document segmentation

7 Discussion

In the previous chapter, We presented the results from different models that we used for our study. These results demonstrated how the models behaved in various settings based on different parameters. In this chapter, we will examine our findings to comprehend the performance of the models and the reasons behind them. We will also compare the various techniques that are deployed for each task. The results of claim segmentation and document segmentation will be examined in this discussion.

7.1 Claim Segmentation

For the claim segmentation task, various settings for our models are used. We'll talk about each model's scores, and the factors that contributed to them, and compare the models to one another to evaluate them.
Before focusing on intelligent models, we must clarify that our supervised models are all evaluated based on a rule-based dataset that is hard coded. This method made it possible to evaluate our models which were not possible due to the absence of any segmentation dataset for the patent claim. But This came with some challenges of segmentation mainly errors in detecting noun phrases to detect new sub-elements in the invention. For ML models, from the support in the classification report, we can notice that the number of words for the class of words that do not end a segment is way larger than the words that end a segment. We can see that we are dealing with imbalanced data which explains the weak F1 score that we got from this model. This model had no good results because it lacks context even when we added some features to words and it deals with words as separate from each other, without any relationship between them. For the CRF model, the same problem of the difference between performance for the two classes. Although the general results seem to be outperforming other models, the problem with this model is that it is probability based and it is working,g on words, not sentences together. For the Bi-LSTM model, the accuracy was poor which means that the number of correct predictions is low but we got a high value of F1 score. For this model, changing the optimizer and adding more data helped with increasing the accuracy value but a high batch size affects it negatively. Combining the CRF and Bi-LSTM models for this task was clearly an improvement and showed excellent results. With this system, we reached an accuracy of 88% and an F1 score of 88%. This model performed better than each model separately. Coming to the BERT model, this model was evaluated on several combinations of parameters. Two types of this model are used, the first one was based on the general English language and the second is pre-trained on patent documents. For the first one, trying to use different parameters to optimize precision values made the recall value lower and resulted in a lower F1 score. From the results, we can see that decreasing the learning rate is helping for better accuracy but it affects negatively the F1 score. The best results in terms of F1 score and accuracy were detected with a batch size equal to 32, a learning rate

of 5e-5, and a number of epochs equal to 3. We tried to use these last results to build our second model. As expected, the BERT model based on patents outperformed all previous models reaching a high accuracy and a high F1 score.

7.2 Document Segmentation

For the document segmentation task, the results from the unsupervised technique were not promising. This is because the model is based on topic change. It has a fixed vocabulary for each topic that can help him to detect boundaries between the segments. But for our description text, even though it is describing different aspects of the patent but it is focusing on the same invention. This aspect could be the reason why this model performed poorly with this task. Another reason is the number of topics fixed to be segmented. We can notice that if we increase the number of topics the P_k value is lower which makes the model better. So, if we increase the number of topics, the model becomes more precise in detecting the boundaries.
As a supervised technique to segment the document, we used the BERTForTokenClassification model. Due to the small size of the dataset, the model was overfitting with 3 epochs and a lower number of epochs didn't affect the accuracy. For this model the precision is low and the recall is high. We have more sentences that don't end a segment than sentences that do, which is a problem caused by unbalanced classes. The accuracy of the model was high with different settings around 90% which makes this model considered accurate and outperforms other document segmentation techniques. For the F1 score, we reached 81% as a trade-off between precision and recall.

8 Conclusion and Future Work

This study provides strategies for automating some steps in the patent analysis process. Our objective is to employ artificial intelligence to automate patent segmentation, which will enhance semantic search. In practice, a quick and effective segmentation process can result in an efficient patent search. Our work specifically focuses on two tasks: claim segmentation and reference document segmentation. The following sections explain a summary of our contributions and future work for each task.

8.1 Claim Segmentation

For this task, we are working on claim text. Our goal is to precisely identify claim features, which requires segmenting the claim so that each novel element of the invention can have its own segment. To present this task as a supervised learning task, we used a rule-based method on USPTO dataset to create new data with segmented claim text that will be used for our AI models. The outcome of the rule-based method is a list of words and each word is classified to be the end of a segment or not.
Now that the data is ready, for our research, we employed Random Forest a machine learning model as well as CRF, Bi-LSTM, and BERT for deep learning models. To test the performance of models, various parameters were used for each of them. The evaluation is done using precision, recall, F1 score, and accuracy. For the Random Forest model, the highest performance with an accuracy equal to 86% and an F1 score equal to 83%. The CRF model yields an accuracy of 97% and an F1 score of 94% when the regularization metrics were set to c1=10 and c2=0.1. Although it appears to have perfect results, this model is affected by data imbalance. As it is working on the words, It has a much higher score for one class than the other one. Our Bi-LSTM model trained with a batch size 8, a rmsprop optimizer produced 60% accuracy and 91% F1 score. The combination of the two previous models gives us good results with an F1 score of 88% and an accuracy score of 88%. For the pre-trained BERT model, we achieved an F1 score of 83% and an accuracy of 78%. The last model outperformed all the previous ones, which is BERT for patents model. With this model trained with a batch size of 32, an adam optimizer with a learning rate of 5e-5, and a number of epochs equal to 2, we could get 96% of accuracy and 82% of F1 score with both precision and recall with high values.
In all the previous models, we are using the same dataset that contains around 300K samples. We noticed that some models are suffering from overfitting. For this reason, a larger dataset should be considered for future experiments to achieve higher accuracy. Imbalanced data problems should be considered for a better F1 score, this can be done by undersampling one of the classes.

8.2 Reference Document Segmentation

The document segmentation task is mainly focusing on the description part of the document. Our aim is to segment the description text as precise as possible to be able to detect every aspect that it mentions of the invention. This will allow the examiner to get more precise details. First, we make use of tags in the dataset to be able to detect all the subparts that exist in this text. We prepare our dataset to be used for the AI model.
For this task, we used two types of learning techniques: The first technique was an unsupervised technique based on topic tiling and the second technique was a supervised technique based on a pre-trained BERT. The Topic Tiling method was based on the LDA model using the cosine similarity metric, it detects whether the topic changes between sentences. The penalty score is used to evaluate this model. The best performance was $P_K = 0.52$. Our pre-trained BERT model was evaluated using precision, recall, accuracy, and F1 score. Different experiments are done using this model and the best performance was detected with a batch size equal to 8, and a learning rate equal to 5e-5. The accuracy, in this case, was 92% and the f1 score was 81%. To get better performance we can use the BERT for patents model that is trained on patents document. The experiments showed that the model is overfitting too, which makes adding more data important for future experiments.

Our aim is to improve the patent segmentation task for a better semantic search. We have already made it possible to detect independent parts of claim text and reference document text to help prepare precise queries and answers for semantic search. The next possible step and future work should assess the benefits of segmentation and similarity scores for patent search.

Bibliography

[AA18] A. Amidi and S. Amidi. *Recurrent Neural Networks cheatsheet.* 2018. URL: `https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks` (visited on 2018).

[Bad+18] P. Badjatiya et al. “Attention-based Neural Text Segmentation”. In: (2018).

[BNJ03] DM. Blei, AY. Ng, and MI. Jordan. “Latent Dirichlet Allocation”. In: *Journal of Machine Learning Research* 3 (2003), pp. 993–1022.

[Brü+15] S. Brügmann et al. “Towards content-oriented patent document processing: Intelligent patent analysis and summarization”. In: *World Patent Information* 40 (2015), pp. 30–42.

[BSF94] Y. Bengio, P. Simard, and P. Frasconi. “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE Transactions on Neural Networks 5* (1994), pp. 157–166.

[CF10] KH. Chan and TW. Fallin. “SYSTEM AND METHOD FOR TEXT SEGMENTATION AND DISPLAY”. In: *United States Patent* (2010).

[DBL97] D.Beeferman, A. Berger, and J. Lafferty. “Statistical Models for Text Segmentation”. In: 36 (1997), pp. 24–26.

[Egn07] D. Egnor. *Document segmentation based on visual gaps.* 2007. URL: `https://patents.google.com/patent/EP1834260A1/en` (visited on 2018).

[EPO22] European Patent office (EPO). *European patent grant procedure.* 2022. URL: `https://www.epo.org/service-support/faq/own-file.html`.

[FSN14] G. Ferraro, H. Suominen, and J. Nualart. “Segmentation of patent claims for improving their readability”. In: *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)* (2014), pp. 66–73.

[Gen21] B. Geng. “Text segmentation for patent claims simplification via Bidirectional Long-ShortTerm Memory and Conditional Random Field”. In: (2021).

[GS20] G. Glavas and S. Somasundaran. “Two-Level Transformer and Auxiliary Coherence Modeling for Improved Text Segmentation”. In: (2020).

[GSC00] F. Gers, J. Schmidhuber, and FA. Cummins. *Learning to Forget : Continual Prediction with LSTM. Neural Computation.* 2000. URL: `https%20://doi.org/10.1162/089976600300015015.` (visited on 2000).

[Hea97] MA. Hearst. “TextTiling: Segmenting Text into Multi-paragraph Subtopic Passages”. In: (1997).

[HS86] R. Hackl-Sommer and M. Schwantner. “Learning representations by backpropagating errors.” In: *Nature 323, 533–536 (1986)* (1986).

[HS97] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural Computation 9* (1997), pp. 1735–1780.

[HW15] D. Hong and B. Wan. "Attention-based Recurrent Neural Networks for Question Answering". In: *Stanford Junior University* 7.2 (2015).

[HW18] H. Husain and H-H. Wu. *How To Create Natural Language Semantic Search For Arbitrary Objects With Deep Learning.* 2018. URL: `https://towardsdatascience.com/semantic-code-search-3cd6d244a39c`.

[Jos91] AK. Joshi. "Natural Language Processing". In: *Science* 253.5025 (1991), pp. 1242–1249.

[Kos+18] O. Koshorek et al. "Text Segmentation as a Supervised Learning Task". In: (2018).

[LBH15] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning." In: *Nature 521 :436–444* (2015).

[OD16] M. Ostrek and L. Dulcic. "Unsupervised Text Segmentation Based on Latent Dirichlet Allocation and Topic Tiling". In: (2016), p. 5.

[PJ17] S. Park and S. Jun. "Technology Analysis of Global Smart Light Emitting Diode (LED) Development Using Patent Data". In: 15 (2017).

[Raj20] Ajit Rajput. *Semantic Search using NLP.* 2020. URL: `https://medium.com/analytics-vidhya/semantic-search-engine-using-nlp-cec19e8cfa7e`.

[RB12] M. Riedl and C. Biemann. "TopicTiling: A Text Segmentation Algorithm based on LDA". In: (2012), p. 5.

[RHW17] DE. Rumelhartand, GE. Hinton, and RJ. Williams. "Patent Claim Structure Recognition". In: *Archives of data sciences, series A* (2017).

[Sof19] SoftwareTestingHelp. *What Is Artificial Intelligence: Definition Sub-fields Of AI.* 2019. URL: `https://www.softwaretestinghelp.com/what-is-artificial-intelligence/` (visited on 10/25/2022).

[SON15] M. Sakahara, S. Okada, and K. Nitta. "Domain-Independent Unsupervised Text Segmentation for Data Management". In: (2015).

[SV19] T. Singh and DK. Vishwakarma. "A deeply coupled ConvNet for human activity recognition using dynamic and RGB images". In: *researchgate.net* 2 (2019).

[SY18] R. Srebrovic and J. Yonamine. "Leveraging the BERT algorithm for Patents with TensorFlow and BigQuery". In: (2018).

[USP19] Gov USPTO. *Patent Examination Process.* 2019. URL: `https://www.uspto.gov/web/offices/pac/mpep/s2103.html` (visited on 2019).

[Vas+17] A. Vaswani et al. "Attention Is All You Need". In: (2017).

[War11] K. Warwick. "Artificial Intelligence: The Basics". In: 192.2 (2011), pp. 13–31.

[WM92] G. Wiederhold and J. McCarthy. "Arthur Samuel: Pioneer in Machine Learning". In: *IEEE Xplore* 2 (1992).

Declaration of Academic Integrity / Eidesstattliche Erklärung

Ich erkläre, dass ich die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe und dass alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, als solche gekennzeichnet sind. Mit der aktuell geltenden Fassung der Satzung der Universität Passau zur Sicherung guter wissenschaftlicher Praxis und für den Umgang mit wissenschaftlichem Fehlverhalten vom 31. Juli 2008 (vABlUP Seite 283) bin ich vertraut. Ich erkläre mich einverstanden mit einer Überprüfung der Arbeit unter Zuhilfenahme von Dienstleistungen Dritter (z.B. Anti-Plagiatssoftware) zur Gewährleistung der einwandfreien Kennzeichnung übernommener Ausführungen ohne Verletzung geistigen Eigentums an einem von anderen geschaffenen urheberrechtlich geschützten Werk oder von anderen stammenden wesentlichen wissenschaftlichen Erkenntnissen, Hypothesen, Lehren oder Forschungsansätzen.

Passau, 16

Bochra Smida

I hereby confirm that I have composed this scientific work independently without anybody else's assistance and utilising no sources or resources other than those specified. I certify that any content adopted literally or in substance has been properly identified. I have familiarised myself with the University of Passau's most recent Guidelines for Good Scientific Practice and Scientific Misconduct Ramifications from 31 July 2008 (vABlUP Seite 283). I declare my consent to the use of third-party services (e.g., anti-plagiarism software) for the examination of my work to verify the absence of impermissible representation of adopted content without adequate designation violating the intellectual property rights of others by claiming ownership of somebody else's work, scientific findings, hypotheses, teachings or research approaches.

Passau, 19. Dezember 2022

Bochra Smida