Evaluation of methods for full-text search in patents

OTTO LAGERQUIST & EBBA TOREHEIM
MASTER´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Evaluation of methods for full-text search in patents

Otto Lagerquist
eng14ola@student.lu.se
&
Ebba Toreheim
tfy15eto@student.lu.se

Department of Electrical and Information Technology
Lund University
AWA Sweden AB & Mindified AB

Supervisor: Fredrik Edman, Anders Fredriksson & Henrik Benckert

Examiner: Erik Larsson

June 21, 2021

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

In this thesis we have evaluated methods for doing full-text searches in patent documents. The aim of patent searches is to find evidence and relevant documents when an invalidity search is done on a patent.

With three different language models, BOW, SPECTER and SBERT, we have evaluated the results of two different text segmentation methods, greedy sentence split and paragraph split, and two different clustering methods, euclidean and spherical. We have found that the spherical clustering outperforms the euclidean one and that both segmentation methods works well for finding relevant parts of documents, both methods with its own advantages and drawbacks.

The configurations were evaluated in four stages, where the first three were automatic and the last one was a manual evaluation by employees at AWA and Lund University. We conclude that our methods have great potential but more testing on a better engineered test set as well as more data from the manual evaluation is needed to draw further conclusions.

Keywords: *natural language processing, full-text patent search, legal tech, document similarity, sentence embeddings, clustering*

Popular Science Summary

Every day the amount of publicly available information increases and with this, a need to quickly and efficiently navigate this jungle of information arises. Wouldn't it be nice if there was a method that understood what you would like to find and then picks out the most relevant texts for you? Well, that's exactly what we've been trying to find and we can tell you that the results are promising.

How many times have you had to go back and change your search query due to not finding what you searched for? How many times have you found tons of document but none that seemed to contain precisely what you were looking for? We believe there is a solution to these problems. The language models of today are actually quite capable of capturing the meaning of shorter texts. Combine them with a method for cutting up long documents into smaller parts and we are almost good to go.

This scenario described is especially troublesome for patent attorneys in their search for prior arts when for instance doing an invalidity search. The most common method today is searching in a patent database using keywords, which makes it easy to miss relevant documents due to synonyms and language variability from different authors. The patent documents are also very long and sometimes all it takes is a short relevant sentence in a long, otherwise irrelevant, document to prove that something is already known.

We believe that the work in our thesis is a good step towards a tool that could help the patent attorneys in their work. By doing searches based on claims we've managed to find shorter passages of patent documents that were deemed relevant by professionals. Even though this might not do all the job for the patent attorneys, we see a big potential for an application that gives a list of potentially relevant documents that otherwise might have been missed.

In our thesis we have tried different methods to segment the long patents into smaller texts which then can be represented by embeddings created by our language models. Of course this leads to a very large number of texts, but by using clustering we can efficiently limit the number of texts we have to search to find the right hits.

Acknowledgements

We would like to thank our supervisors Fredrik Edman at LTH, Anders Fredriksson at AWA and Henrik Benckert at Mindified for all the support and encouragement we have received during our work. We would also like to thank the employees at AWA and some contacts of Fredrik Edman for evaluating our results as well as the employees at Mindified for helping and guiding us through the thesis.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Background

One of the tasks of a patent engineer is to perform an invalidity search for a patent. It means that you search for reasons for the patent to be invalid and that it should not have been approved in the first place when going through the application process. The most common reasons are that the invention is not novel, i.e. previously known, or not inventive, i.e. obvious for a skilled person in view of what was known at the time of filing the patent. For something to be known it has to have been published or in any other way made publicly available before the filing date of the patent application. To do this search, the patent engineer usually searches in a database of patent documents to find similar patent documents. If the invention is found in a patent document published before the application it is not novel and then there is a reason for the patent to be invalidated. Due to the vast amount of information published and in other ways made available, and that an invention can be described with different words, there is a big risk of missing important information that could affect the patentability of an invention.

1.2 Problem

A patent document comprises of a few standardized sections such as abstract, description and claims. The search in the patent database is usually done using keywords which is good for finding the documents that matches the exact search words, but this might lead to missing documents that are related but uses other words. The description part can be very lengthy and describe many different things related to the invention even if it is something that is not protected by the claims. Those small bits of information can be difficult to find when they are written using other words than used in the search query. They are also easily overlooked during analysis due to the length of the text, even if the right document is found.

A method to limit the matches of a keyword query is to add classification as a parameter, i.e. if the search is for documents containing cameras the corresponding class would be entered as a parameter. However, this has the disadvantage that documents that mention cameras, even though they don't belong to the main content, might be overlooked since the classification would be based on the main subject of the patent document.

1.3 Our task

In this thesis we will examine different methods and evaluate them and find which ones that can be used to find descriptions containing information that matches a search query, even though the matching part of the description is small and not very related to the general topic of the description. The search query should have a length and structure similar to the first claim of a patent. In order to facilitate for the patent engineer, we would like our methods to find the descriptions that possibly contain similar information even if not written in the exact same words. Due to the length of the description we would also like to point out where in the text the match was found. With this we will assess how today's Natural Language Processing algorithm's perform on the task of finding relevant passages in patent documents.

1.4 Limitations

Due to the size of the patent database available to us, consisting of 2 019 795 publicly available european patent documents in english, we have chosen to limit our tests to only one class of patents. This is the international patent classification (IPC) level E, which contains patent documents regarding 'fixed constructions'. This is a well defined class and it reduced the number of patents in the database to 57 453 and the effect of this will be that we reduce computations and memory storage by a large factor while we still keep enough documents to be able to draw some conclusion about the effectiveness of our methods in the end.

1.5 Related work

1.5.1 Automating the search for a patent's prior art with a full text similarity search

In the report *Automating the search for a patent's prior art with a full text similarity search*, Helmers et al. use similarity comparisons of full texts and existing patents to find prior arts and information that could be relevant to a new invention. They achieved good results with both a Bag-of-words model and Neural Network Language Models such as word2vec and doc2vec.[1] For the full texts Bag-of-words performed best and for more limited parts of the documents, such as abstract or claims, doc2vec performed best. They reason that combining embeddings for too long texts will make the result too noisy and that the models are better adapted for smaller more coherent text passages. They used a relatively limited dataset comprising of patents belonging to the Cooperative Patent Classification scheme (CPC) category A61, 'Medical or veterinary science and hygiene'.

1.5.2 Using natural language processing to identify similar patent documents

A previously written work related to ours is the master's thesis *Using natural language processing to identify similar patent documents* by Hannes Jansson and Jakob Navrozidis. In their work they use the patents title and abstract and compare this with others to find patents that with a high probability contain related information. What they found is that the transformer based model SBERT performed best when finding semantically similar abstract.

1.5.3 PatSeg: A Sequential Patent Segmentation Approach

In the article *PatSeg: A Sequential Patent Segmentation Approach* the authors, M. Habibi et al., use language processing to segment a patent description into different categories such as summary, experimental data and lists among others. This is done in a two step process where they first segmented the description in an unsupervised manner and then they classified the parts into the right category. In both steps they used semantic word embeddings. The segmentation method that they found best was TextTiling.

Chapter 2
Theory

2.1 Patent theory

2.1.1 Structure of a patent

Every patent that has been granted follows a certain standardized structure. The first part is a page with bibliographic data. Among other things this page contains information about the inventor, the application number and filing date. Two other important items on the first page is the title and the abstract. The abstract is a short text describing the main idea and technology of the invention in a maximum of 150 words.

Following the bibliographic data comes the description. The description makes up the main bulk of text in a patent and can range between a few hundreds of words to several tens of thousands of words. The description itself has no standardized structure, but an arrangement of subsections such as background, summary, detailed description and if applicable, experimental data, is commonly used. In other words it is in the descriptions that most information about the invention can be found. The description also sets up the context for how to interpret the claims in terms of definitions of expressions and the like.

After the description comes the claims. There is no requirement of a specific number of claims but it is only what is in the claims that is protected by the patent, so usually there is a list of claims referring to the different aspects of the invention. The list is always written with the first claim being in the most broad terms, capturing the overall idea of the invention, and as the list goes on the claims become more narrow and specific about the details of the invention.

2.1.2 Searching for patents

There are many reasons for wanting to search for patents. Every patent or patent application that has been made public is a source of knowledge. This means that when you are investigating whether an idea is patentable or if you want to find proof that another patent should not have been granted, it's a good idea to consult with this vast source of information. There are different tools for a patent engineer to search for these documents and some of the most common are Espacenet [4], provided by European Patent Office and Total Patent provided by LexisNexis [5]. The basis of these tools is a keyword search, with the option to add different filters.

Even though the searches are aimed at previous patents it is not a requirement that the invention is published in a patent document for it to be known. Any type of publication, e.g. a scientific article, would be sufficient.

2.2 Embeddings

When human beings compare two texts we simply read them and we know what they are about. For a computer it is not as easy. A computer can read and handle text but it doesn't necessarily understand what it is about. Typically, computers work better with numbers, and therefore one method of automatic text comparison is to convert text into numbers. Then it is easy to use mathematical algorithms to check if the numbers are close together or far away.

There are different ways to associate certain numbers to a certain text. One way is to use a language model. A language model has to be trained on large amounts of text and can then be used for various tasks such as prediction of the next word in a sentence. From a language model like that it is also possible to extract the numerical representation of words or sentences. The way that an AI language model represents a word or text is heavily influenced by the training process. Many available models have often been pretrained on a large corpus (Wikipedia is pretty common) and then it is up to the end-user to fine-tune the model for a specific task.

The numerical representations can be made up of different features, each with its own value, and all together these numerical vectors are called embeddings. In this section we will cover the different methods we are going to use for embeddings in our thesis.

2.2.1 Bag-of-words

Bag-of-words (BOW) is a method to vectorize a text without capturing the semantics of it. A BOW model consists of a vocabulary that keeps track of all the individual words it has come in contact with during its training. When a vocabulary is in place, an embedding of a text can be generated by counting the words in the text and put the numbers in a vector. Each position in the vector corresponds to a specific word in the vocabulary. As a consequence the vector of the text can be very large and contain a lot of zeros. A short example of this for a corpus with two sentences can be seen in table 2.1. In table 2.1 the vector representation of the first sentence is $[1, 1, 1, 1, 1, 0, 0, 0]$ since it contains each of the words 'the', 'device', 'has', 'a', 'door' once and the other words in the corpus zero times.

Sentence\vocabulary	the	device	has	a	door	opens	outward	from
The device has a door	1	1	1	1	1	0	0	0
The door opens outward from the device	2	1	0	0	1	1	1	1

Table 2.1: Short example of Bag-of-words model

The method we will use in this thesis to generate the BOW embedding is based on the TF-IDF score for each word in the text. TF-IDF stands for term

frequency-inverse document frequency and is a measure of the importance of a word to a text in a set of several texts. The term frequency is computed as the number of occurrences of a word in a text divided by the total number of words in the text. The inverse document frequency is computed as the logarithm of the number of texts in the corups divided with the number of documents the current word is present in. The final score for a word w_i in document j, is the product of TF and IDF as seen in equation 2.1.

$$TF\text{-}IDF_{i,j} = \frac{frequency(w_i)}{(word\ count\ in\ document\ j)} \cdot \frac{\log(total\ number\ of\ texts)}{(number\ of\ texts\ with\ w_i)} \tag{2.1}$$

A problem with BOW is that the dimensionality grows with every new word and as mentioned before the vectors may contain a lot of zeros. There are several ways to restrict the number of words in the vocabulary and in our model we do the following restrictions. First we remove stop words (e.g. 'is', 'on' and 'in'), these are frequently used words that do not add understanding to the model. We use the stop words that are predefined in the python library Natural Language Toolkit (NLTK) [6]. Secondly we stem the words, meaning that we extract the root of the word. An example is 'thinking' and 'thinks' which both will be reduced to 'think'. Thirdly we will limit the number of words by setting a limit on the model's vocabulary since the memory usage it takes to store a large number of large non sparse vectors is too high for the hard drives we have available and the computation times would be too long to fit within the time scope of this project.

2.2.2 BERT based methods

A common way of creating language models is to build different kinds of neural networks. BERT stands for Bidirectional Encoder Representations from Transformers and it is a language representation model.[7] As the name suggests it uses a bidirectional encoder to condition on both left and right context. It has a vocabulary of 30 000 words and it learns an embedding for each input token (e.g. a word or a number) given its context. During pre-training BERT takes a sequence of text as input and masks 15% of the tokens and then tries to predict them. To get the embeddings out of the model we look at the last hidden state of the neural network, before the layer making the prediction. The values of the nodes in this state can be seen as the features of the word in a vector space.

BERT is also trained in next sentence prediction where it gets two sentences as input where the second sentence is tagged as either 'isNext' or 'notNext'. When the two sentences are passed into the network a score is calculated which could be seen as a semantic similarity score. While BERT could be used for similarity comparison between sentences it would be very slow. To give a similarity score, every pair of sentences would have to be passed through the network. For n sentences that means $n * (n-1)/2$, which is $O(n^2)$, computations which scales up very fast when we have a large amount of sentences.

SentenceBERT

SentenceBERT (SBERT) is a modification to the regular BERT that uses a siamese network to create semantically meaningful sentence embeddings.[8] This would give each sentence a unique representation in a vector space which would then facilitate operations such as clustering and information retrieval using semantic search, which is what we will be focusing on in this thesis. SBERT takes a sequence of text with a max length of 512 tokens, and by adding a pooling layer to the output of a BERT network, SBERT derives a fixed length vector for each sequence. Although the name suggests an input of sentences, the input to SBERT can actually be any sequence of text (e.g. many consecutive sentences) as long as the maximum length is not exceeded. Jansson and Navrozidis (2020), found SBERT to be a good option for matching a search query with abstracts and titles from patent documents, and therefore SBERT will be one of our main subjects for this thesis. This model has been trained with the datasets SNLI [9] and Multi-genre NLI [10].

SPECTER

Scientific Paper Embeddings using Citation-informed TransformERs (SPECTER) is another model based on BERT.[11] What makes it special is that it is trained on scientific papers where it takes the title and abstract as input text, and then it uses the citations of the papers as labels for how the papers are related. Unlike most other models, SPECTER is ready to be applied in a downstream task, e.g. embeddings or word prediction, without fine tuning. It collects the global information of the input into a special [CLS] token. It is the last hidden state corresponding to this token which is used as an embedding when comparing documents. Hopefully it is an advantage that the SPECTER network has been trained on scientific articles, since it is probably more closely related to the type of language in patents than other large corpora (e.g. Wikipedia).

2.3 Pre-processing the text

Before it is possible to generate an embedding, the text needs to be extracted from the files and segmented into parts suitable for the language models.

In the original data everything is stored in a format called Extensible Markup Language (XML) [15] and therefore the first step is to extract the patent text from the database and restore it as pure text. Since two of our models have a limit on how long the input text can be for it to produce an embedding, it means that we will have to split up the texts in smaller pieces. For one of our segmentation methods we will use the XML tags for the paragraphs to perform the splitting. In the rest of the methods we will use the text without XML formatting as input.

2.3.1 Segmentation

The BERT-based methods used in this thesis have a limitation of 512 tokens because the memory usage scale quadratically with the sentence length.[7] Due

to this limitation it is necessary to segment the texts into smaller parts. In this section the methods used in the thesis will be explained.

Greedy sentence split

The greedy sentence segmentation is a method that divide the text into chunks that are as close to the maximum number of tokens as possible but never greater. The chunk are generated by sequentially adding a sentence to a box, if this results in too many tokens for the box it is closed and the sentence is put into a new box instead. This is to guarantee that no box is overfull as well as no sentence is split into several parts. One exception is when a sentence has more tokens than the limit of the box. Then the sentence is put in its own box and when the embeddings for SBERT and SPECTER are computed only the first 512 tokens are used.

Paragraph split

A patent description is naturally divided into paragraphs defined by the author. These are often a good indication of where the text shifts focus and could therefore be used as individual segments when searching for relevant passages in a patent. A downside with this compared to the greedy sentence split is that the number of segments is much higher and therefore it could be too demanding to compute all embeddings, cluster them and then finding the right segments in a search. This segmentation is done in the same step as the extraction of the text from the XML code.

TextTiling

TextTiling is a more complex text segmentation method that is implemented in the NLTK library.[16] The complexity lies in that each sentence or paragraph is semantically evaluated to find the best places to split the text.

The algorithm traverses the text with a sliding window containing a number of sentences or paragraphs. It calculates and keeps a similarity score of the text contained in the window at each position. In the end it splits the document in positions where it maximizes the semantic similarity within each segment.

After a few initial tests with TextTiling on a few documents we discarded the segmentation method from further use. This was due to an unreasonable long computation time. The method also resulted in an immense number of segments for each document which meant that our memory capacity wouldn't be enough when later generating embeddings. Considering all this we decided it was not worth it and that for our evaluations, paragraph split would yield sufficiently similar splitting and semantic homogeneity of the segments.

2.4 Clustering

When dealing with a lot of data it can be ineffective to search through every data point. A way to get around this is to group data points together that share a similar structure on a desired feature. All clusters will have a cluster centre which is the mean feature vector of all the data points that is contained within a group.

An initial search is done on the cluster centers to find which clusters are most likely to contain relevant information. These clusters are then chosen for an extensive search over all data points they contain.

2.4.1 Initialization

There are mainly two different methods to initialize cluster centers, either randomly or using k-means++ [12]. Here random is straight forward, one simply takes as many random data points as one want to have as number of clusters. This could possibly lead to slow convergence due to lack of spreading of the centers. A more commonly used method is k-means++ which chooses the initial cluster centers in a way that generate diverse centers which leads to a faster convergence. However neither method guarantees convergence to the global minimum and they often fall into local minima when converged.

2.4.2 Mini-batch k-means

To speed up convergence when dealing with a lot of data it is possible to use the mini-batch k-means method.[13] With this the data is divided into several batches and the cluster centers are adjusted depending on one of these batches at a time. An example of this batching can be seen in figure 2.1. To the left in figure 2.1 a set of 100 samples can be seen and in the middle a subset of 30 points have been selected for the first epoch. To the right in figure 2.1 another subset of 30 samples has been selected for the next epoch. This means that in each epoch the number of computations will be (*batch size* $\cdot$ *nbr of clusters*) instead of (*total nbr of data points* $\cdot$ *nbr of clusters*). The speed up for each epoch will be bigger the more data points you have. In return the algorithm will need a larger amount of epochs to converge. An effect of not looking at the whole set of data points is that the cluster centers will adjust slightly in each epoch and therefore a tolerance will also have to be set for when to stop.

2.4.3 Different distance measures

A method to measure distance between data points and the cluster center is the euclidean distance as seen in equation 2.2 where $\mathbf{X^c}$ is the vector for a cluster center, $\mathbf{X}$ is the vector for the data point. x_i^c and x_i are the i:th coordinates of the cluster center and the data point respectively. This is the standard way to measure distances within clusters and is implemented by most libraries such as Scikit-learn [14]. However, when measuring similarities between texts a more common method to use is the cosine similarity where the vectorization of texts are projected onto the unit sphere. It could therefore be beneficial to cluster the data with respect

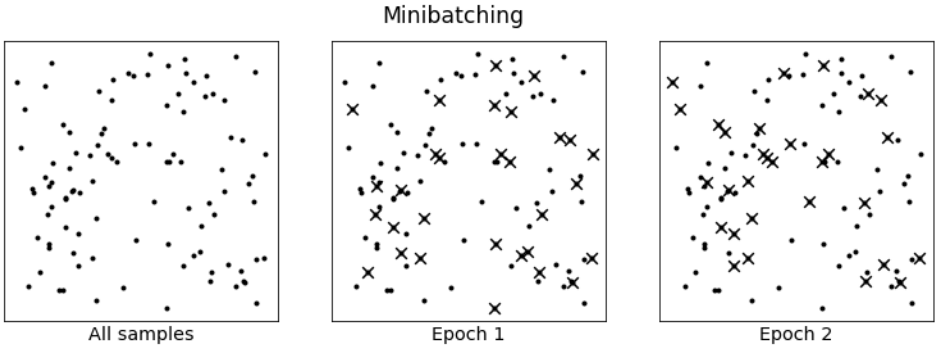

Figure 2.1: In a set of 100 samples a random mini-batch of 30 samples is selected in the first epoch. In the next epoch a new mini-batch is randomly selected.

to this instead. The cosine similarity is seen in equation 2.3 where $\mathbf{X^c}$ and $\mathbf{X}$ is the same as in equation 2.2

$$f(\mathbf{X^c}, \mathbf{X}) = \sum_{i=0}^{N} \sqrt{(x_i^c)^2 - (x_i)^2} \tag{2.2}$$

$$f(\mathbf{X_c}, \mathbf{X}) = \frac{\mathbf{X_c^T X}}{|\mathbf{X_c}| \cdot |\mathbf{X}|} \tag{2.3}$$

The cosine similarity, that ranges between -1 and 1, can easily be converted to a distance measure that ranges from 0 to 2 by taking 1-(cosine similarity). Using the cosine distance yields the same results as first projecting the points to the unit n-sphere and then using the euclidean distance measure. A simple 2D visualization of how the two distance measures will affect the clustering can be seen in figure 2.2.

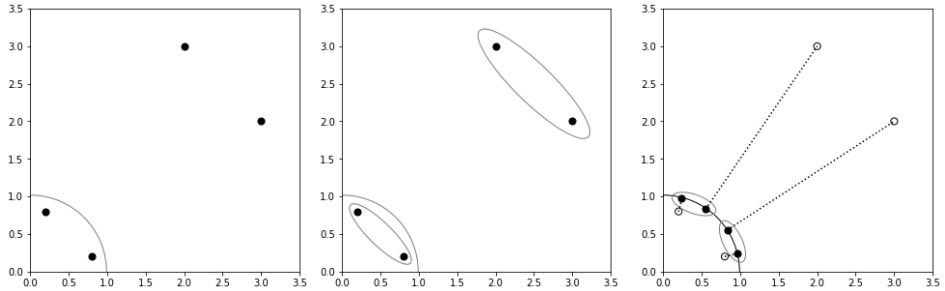

Figure 2.2: To the left, four data points and part of the unit circle can be seen. In the middle, the data points are clustered in two groups using the euclidean distance measure. To the right, the data points are clustered using the cosine distance instead and the clusters are not the same.

The different versions of the spherical clustering methods that we found did not implement mini-batching and was therefore not well suited to used together with paragraph split. This would have led to unfeasible computation times.

Chapter 3
Method

3.1 Architecture

In order for us to evaluate the different methods of retrieving text passages from descriptions that match an input claim we have created a simple program. The program takes a search query as input and returns the most relevant passages from different descriptions. For this program to run in a reasonable time we have to store a lot of the information in a database. By doing a lot of computations offline beforehand and storing as much information as possible in the database we can limit the number of computations that has to be done every time the program runs. This shortens the runtime for one search from approximately a day or two to a few minutes given that we only search in IPC level E. To evaluate our different methods we will have four kinds of evaluations performed at different stages in the search, which will be further explained in section 3.4. The workflow/architecture of our program can be seen in figure 3.1.

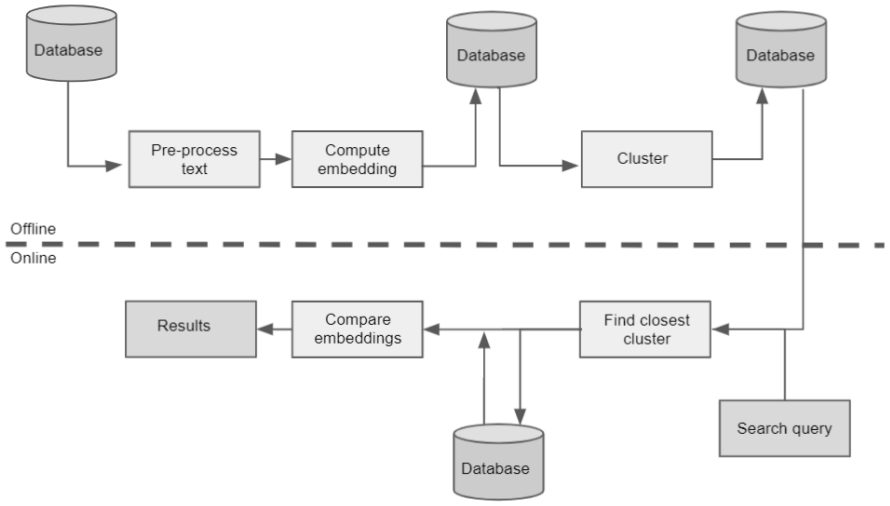

Figure 3.1: Pipeline of the project

In figure 3.1 we first access the raw data from the patent documents which is then pre-processed by cleaning the XML code and splitting the texts. From the pre-processed embeddings are generated and stored in the database together with the clean text segments. The embeddings are then retrieved from the database and clustered, and the information about the clusters is also stored in the database. With the database prepared a search query can be fed to the program and the closest clusters are located. The embeddings in these clusters are retrieved and a similarity comparison between the search query and the embeddings is performed. Lastly the results are presented in descending order.

3.2 Database

The database we use for our program contains all the needed information about the patents and their descriptions. The database was stored in a file system on disk, on a computer made available from Mindified. Categories of the patents that we store in the database are patent number, the full description and the first claim. We also went through every full description and segmented it into smaller parts and stored them as separate files using our different segmentation methods. The segments were stored in a manner so that the document they originated from could be found. In addition to this we also computed the embeddings, using the methods described in the theory section, for every text segment and full description and stored them as well for easy access during the search. The database also contains information about the clustering, e.g. cluster centers and which texts belong to which clusters in order to make the search more efficient.

3.3 Search

The search, visualized in the bottom part of figure 3.1, is done by taking an input query, supposedly a short text disposed as the first claim in a patent document, to be processed and used for computing an embedding. Then the program will search through our database for embeddings produced by the same model and find the best matches and fetch the corresponding texts. These will be shown in a ranked order together with their similarity score.

3.4 Evaluation

The evaluation is a very important step since it is used to decide if the search results provided by our methods are reasonable or not. It is also a way of estimating if the quality is high enough for it to be a tool that can be used professionally. We have chosen to create a chain of evaluations that can be seen in figure 3.2 where the first evaluation is performed directly after the embeddings are generated. The second evaluation is performed after the clustering to see how the clusters affect the results. The third and fourth evaluations are performed independent of each other where the third evaluation is focused on the positioning of the results and the fourth evaluations is a manual verification of the results by professionals.

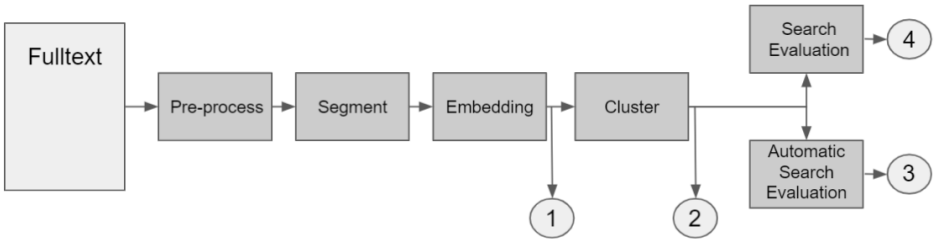

Figure 3.2: Pipeline of the evaluation process

Since evaluations performed by humans are time consuming and possibly subjective, we have aimed to do as much of the evaluations as possible automatic. Therefore only the fourth evaluation step will be performed manually. One of the difficulties with evaluating search methods is that there is no ground truth to compare with in order to know if the search was successful or not. It is possible to compare with search reports that has been made but these have some limitations. Firstly, they are often based on keyword searches and secondly, the person that performs the search often stops after a few examples even if more examples exists. Another difficulty with the evaluation is to separate true negatives, i.e. if we have no relevant results because the invention is new and no similar documents exists, from false negatives, i.e if we just miss the relevant documents.

3.4.1 Evaluation 1: Basic semantics capturing

The first stage of the evaluation will be a test of our models capability of giving similar embeddings to similar texts. For every document the embeddings of the claim and the description will be compared and the average cosine similarity will be calculated. Every claim will also be compared to a random description and the average cosine similarity of that is calculated as well. In the cases where we have segmented the descriptions into several parts, we will compare the claim with each part and then choose the maximum similarity as the similarity score for that document. Some of the random couples of claim and description will result in a good match, but on average it should be worse. If the models catch the semantics of the texts in the generated embeddings it is expected to get a higher similarity score on average when comparing claims and descriptions from the same document. As a result the two average similarity scores will hopefully diverge if tested on enough pairs. The evaluation will be performed on class E as previously stated. The test will therefore consist of 57 453 pairs with true match between the claims and the descriptions as well as an equal amount of random pairs.

In this stage of the evaluation we will two different BOW models with vocabularies of 2000 words and 10 000 words and they will be referred to as 'bow2' and 'bow10' respectively. These two models will be tested on the full-text similarity to get an indication on how the vocabulary size affects the results. If not specified and in later stages of the evaluations BOW will always refer to 'bow2', which is the model we will continue with.

3.4.2 Evaluation 2: Clustering

In order to improve the efficiency and keep down the computation times during runtime the texts will be clustered. To make sure that not too much relevant information is lost due to the clustering some tests will be performed.

First a cluster study will be performed for a set of different number of clusters and the inertia, which is a kind of measure of how the data within the clusters are spread, will be measured and plotted. This is to get an overview of the impact of the number of clusters. After the cluster study, a small set of number of clusters is chosen to evaluate the impact of the number of clusters the search is performed in as well as if the clusters are based on euclidean or cosine distance. For these combinations the average of the maximum self-matching score for claims towards their own descriptions is computed. These results will be presented together with a reference to both 100% and 90% of the result from evaluation 1 to easier visualize the information loss.

With the results from this evaluation a decision will be made for each configuration (language model + segmentation method) on a combination of the number of clusters in the clustering and the number of clusters to search in for the following evaluations. For each configuration we will also decide if euclidean or spherical clustering will be used.

3.4.3 Evaluation 3: Ranking

In this part of the evaluation a search will be performed on the first claim from each document. The average position in the ranking of the search results, from highest to lowest similarity, will be computed for the best found match from the corresponding description. From this we will get a sense of how the best self-match compares to other matches.

3.4.4 Evaluation 4: Manual control

The fourth and last stage of evaluation will consist of making a number of searches with our different methods ranking the results. This will be done manually with the help of employees at AWA since there is no list of correct answers to compare our results to.

For each configuration we will make a number of searches based on the first claim from a patent document belonging to our subset of IPC level E. We will pick out the results in places 1, 2, 3, 5, 10, 25, 50, 75 and 100. To these results we will add one result from a search report on the same patent. The order of the results will be mixed and presented to the employees at AWA and Lund University to be ranked against each other in relevance for a invalidity search. The search results will also be individually classified as 'X' (all features of the claim (input) is disclosed in the text), 'Y' (all features are not presented, but when combined with another document (also an Y document) all features are presented by the combination of documents) or 'A' (other). An example of the form used can be seen in Appendix A.1. The results from this evaluation will give us a measure both on how relevant the search results from our methods are and how they compare

to those results found by a person doing a search report. We will also see if our conclusions from previous evaluations seems to be in the right direction.

Chapter 4
Results

Here we will review the results of each method in the different evaluation steps described in the previous section. All the computations have been performed on a computer provided by Mindified with a 24 core AMD 2970WX Threadripper CPU, a Gigabyte GeForce RTX 3090 24GB TURBO GPU and 128 GB RAM. The operative system was Linux and all code was implemented in Python.

The two segmentation methods we used resulted in different lengths of the segmented texts which lead to a different number of total texts as well. In table 4.1 some statistics of the text splitting methods and their results can be seen. The paragraph split results have a lot more text segments than the greedy sentence split.

Statistic\Splitting method	**Greedy splits**	**Paragraph splits**
Total number of documents	57 453	57 453
Total number of parts in all documents	735 142	3 235 450
Max number of parts in one document	411	2 457
Mean number of parts per document	12.8	56.3

Table 4.1: Results from the two text segmentation methods.

4.1 Evaluation 1: Basic semantics capturing

The first evaluation method gives a measure of each model's basic ability of giving similar embeddings to texts that have a similar meaning. The results of this first evaluation for different combinations of model and segmentation method can be seen in table 4.2. The models 'bow10' and 'bow2' are BOW models restricted to a vocabulary of 10 000 and 2000 words respectively. 'True mean' is in here defined as the mean similarity score of a claim and description belonging to the same document and 'random mean' is defined as the mean score for a claim compared to a random description.

Model	Segmentation method	True mean	Random mean	Diff (true-rnd)
bow10	no segmentation	0.4779	0.0545	0.4234
bow2	no segmentation	0.5726	0.0893	0.4833
bow2	greedy	0.7337	0.0752	0.6585
bow2	para	0.8529	0.0954	0.7575
sbert	greedy	0.8583	0.6671	0.1912
sbert	para	0.9066	0.6851	0.2215
specter	greedy	0.9258	0.7448	0.1810
specter	para	0.9496	0.7575	0.1921

Table 4.2: Results from first evaluation for different models and segmentation methods.

In figure 4.1 and 4.2 the distributions of the results can be seen for both the true and the random matches when methods for splitting up the descriptions where used. In figure 4.3 we can see the same results when comparing 'bow2' with 'bow10' on the full length descriptions.

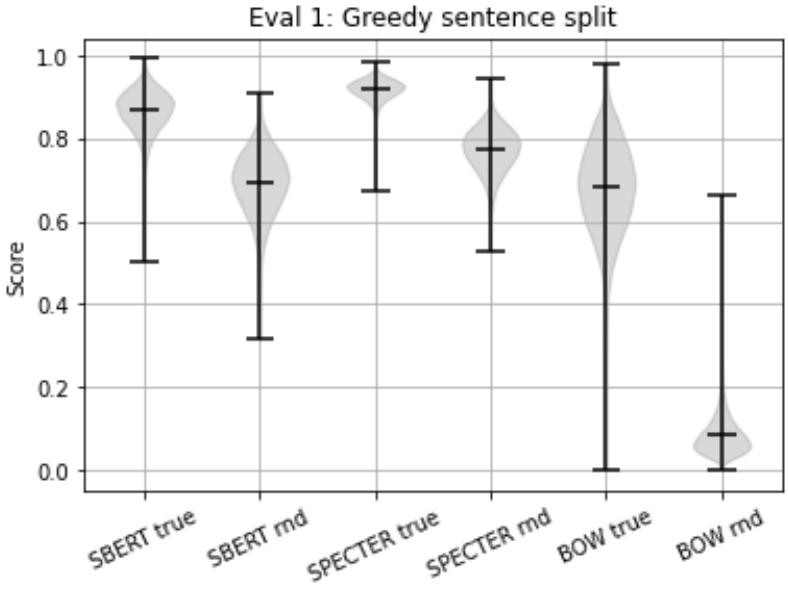

Figure 4.1: Distributions of results from the first evaluation using greedy sentence split. Here 'true' is short for true matches and 'rnd' is short for random matches.

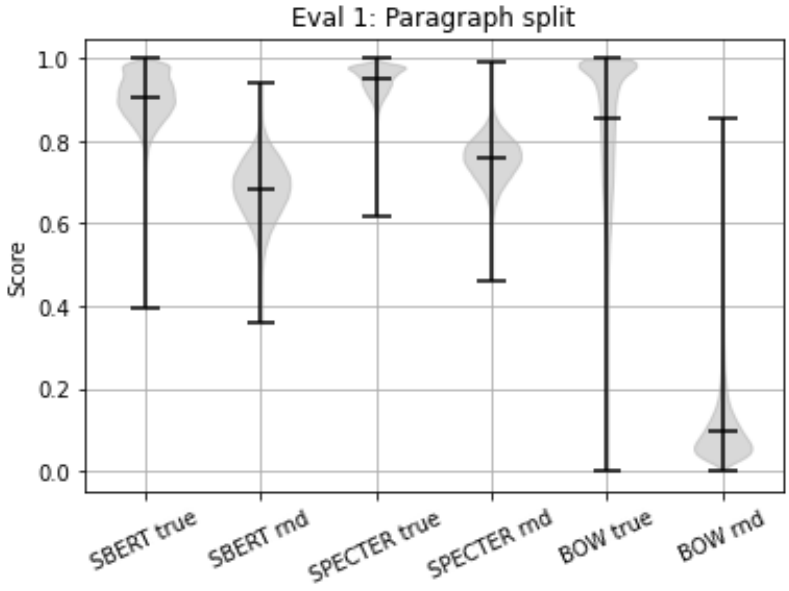

Figure 4.2: Distributions of results from the first evaluation using paragraph split. Here 'true' is short for true matches and 'rnd' is short for random matches.

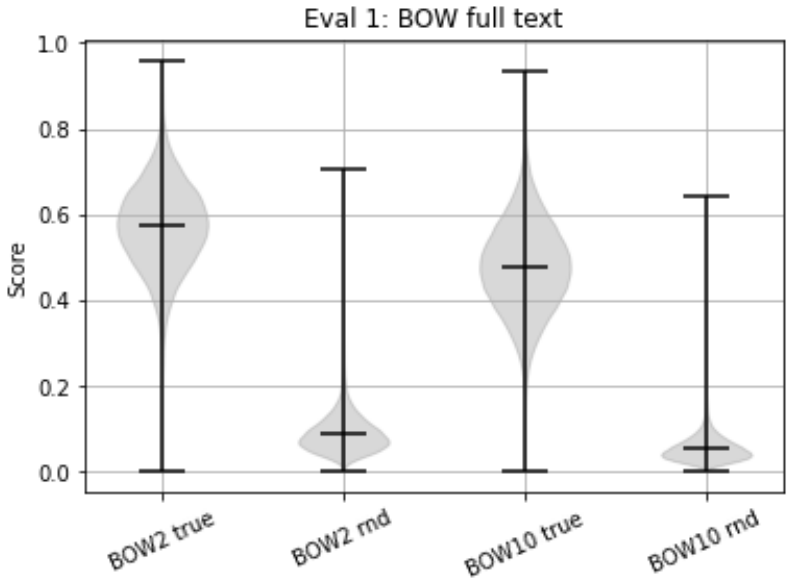

Figure 4.3: Distributions of results from the first evaluation using bow2 and bow10 on full length texts. Here 'true' is short for true matches and 'rnd' is short for random matches.

4.2 Evaluation 2: Clustering

The figures 4.4, 4.5 and 4.6 shows the result from the cluster study performed on the SBERT embeddings. Figure 4.4 shows how the inertia of the clusters change with number of clusters for spherical clusters when performed on the greedy split sentences and it can clearly be seen that the inertia goes down as the number of clusters increases. Figure 4.5 shows the corresponding curve for euclidean clusters with the same trend of decreasing inertia for more clusters. In figure 4.6 we see the inertia for euclidean clusters performed on the paragraph split sentences and the inertia decreases with more clusters in this graph as well, even though the inertia seems to follow two possible curves.

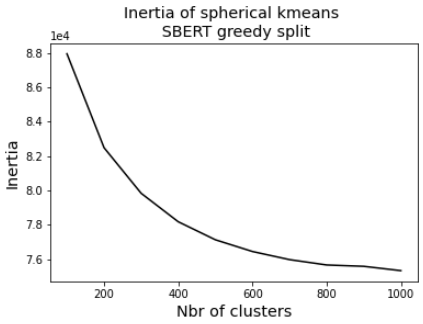

Figure 4.4: Cluster study on spherical clusters with greedy sentence split

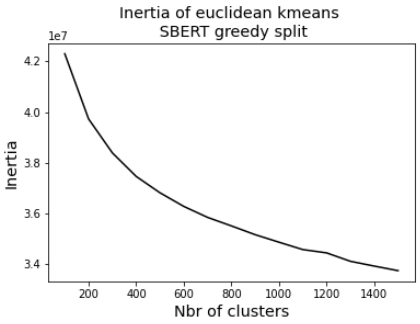

Figure 4.5: Cluster study on euclidean clusters with greedy sentence split

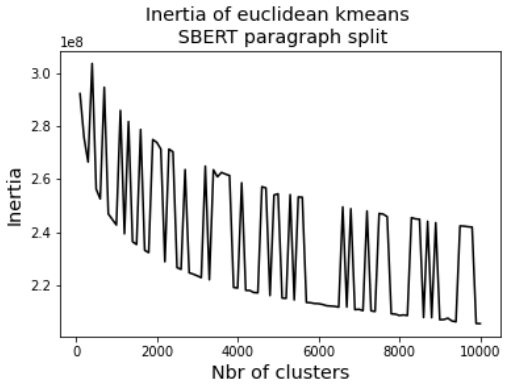

Figure 4.6: Cluster study on euclidean clusters with paragraph split

Figures 4.7, 4.8 and 4.9 show the results from the second evaluation on the configuration with model BOW and greedy sentence split. In figure 4.7 the results from evaluating with euclidean clusters are shown for 300, 600 and 1000 clusters. After approximately 5% of the clusters have been searched through the lines coincide, and at approximately 15%, the score has nearly reached to the score from

evaluation 1. In figure 4.8 the results for 600 and 1000 cluster coincide at approximately 15% at the score of evaluation 1. Already at 5% the results for 1000 clusters has reached the score of evaluation 1. The results for 300 clusters constantly lies at a lower level than the others and never reach up to the score of evaluation 1.

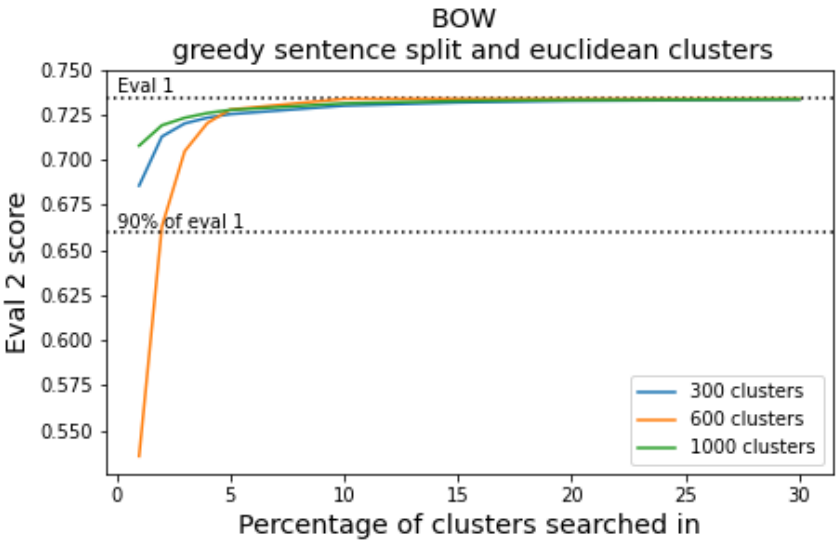

Figure 4.7: The results from the second evaluation for the BOW model with the greedy sentence split and euclidean clustering.

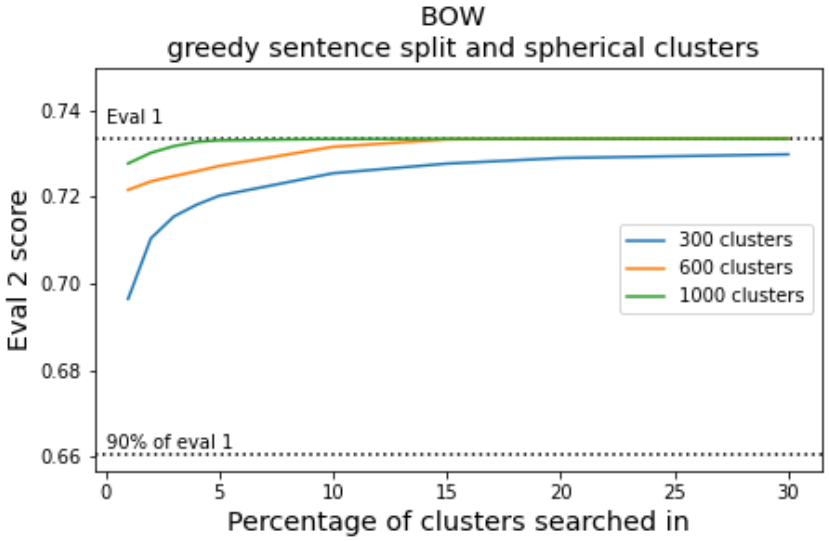

Figure 4.8: The results from the second evaluation for the BOW model with the greedy sentence split and spherical clustering.

The results in figure 4.9, where we compare 1000 clusters for each method, shows that for results searched in fewer than 20% of the clusters the spherical clusters perform better and for evaluation 3 and 4 the cluster configuration will be 1000 spherical clusters with a search ratio of 5%.

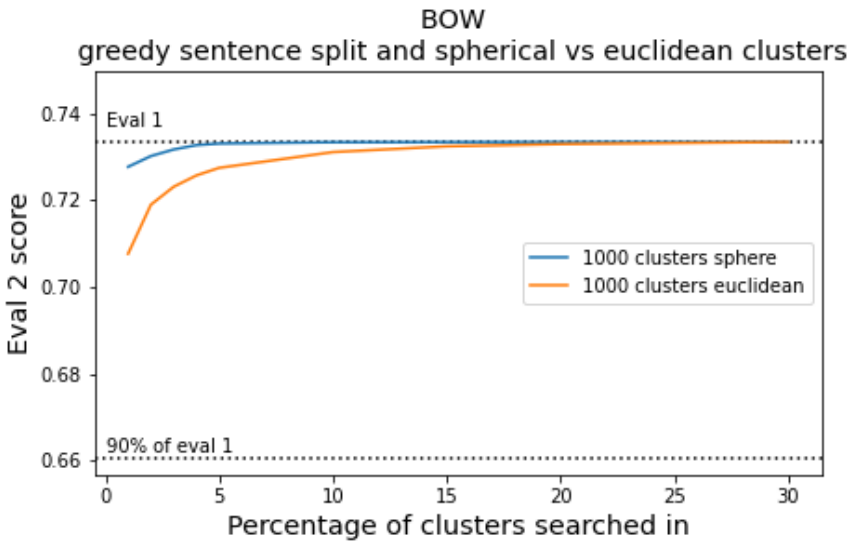

Figure 4.9: A comparison from the second evaluation for the BOW model with the greedy sentence split and spherical clustering vs euclidean clustering for 1000 clusters.

Next configuration is model SPECTER with greedy sentence split and in figures 4.10, 4.11 and 4.12 the results for this configuration are shown. In figure 4.10 the results for the euclidean clusters are shown where one can see that the curves for all three cluster sizes are separated, though 600 and 1000 are close to each other. All curves reach over 90% of the evaluation 1 score but never reach to the full score because some of the best self-matches is not found in the clusters we have searched through.

In figure 4.11 all cluster sizes are close together and coincide at approximately 10% and they reach up to the score from evaluation 1 at approximately 20% of searched clusters. For a lower search percentage we see that 1000 clusters are slightly better and we will compare this with the 1000 clusters from the euclidean case in figure 4.12.

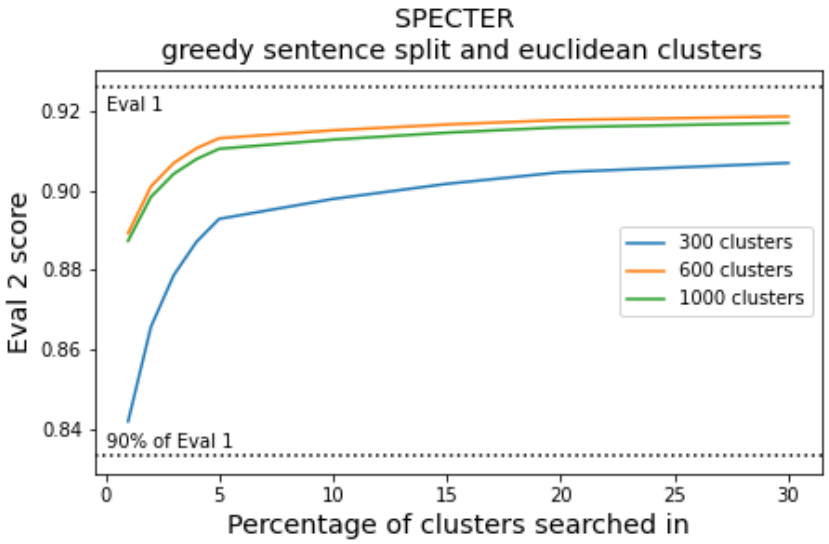

Figure 4.10: The results from the second evaluation for the SPECTER model with the greedy sentence split and euclidean clustering.

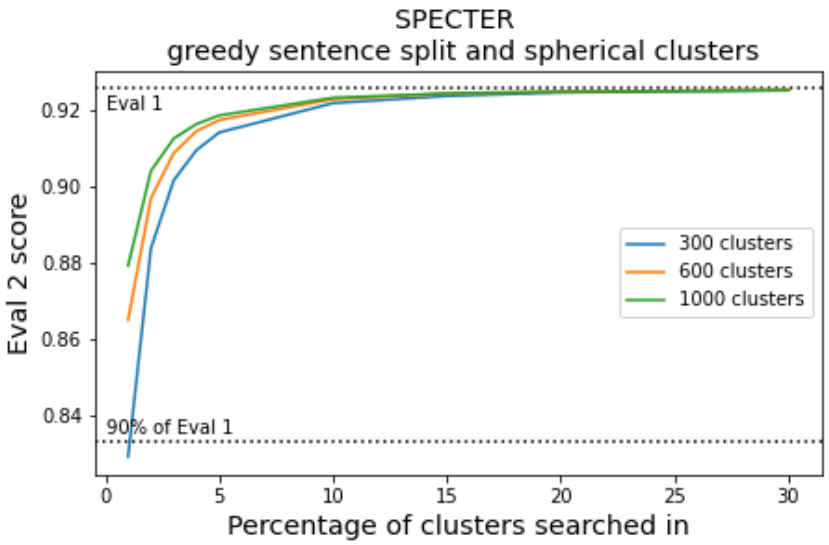

Figure 4.11: The results from the second evaluation for the SPECTER model with the greedy sentence split and spherical clustering.

The comparison of the euclidean and spherical clusters in figure 4.12 clearly shows that the spherical clusters perform better than the euclidean for 1000 clusters. The configuration going forward with evaluation 3 and 4 will be 1000 spherical clusters at 10%.

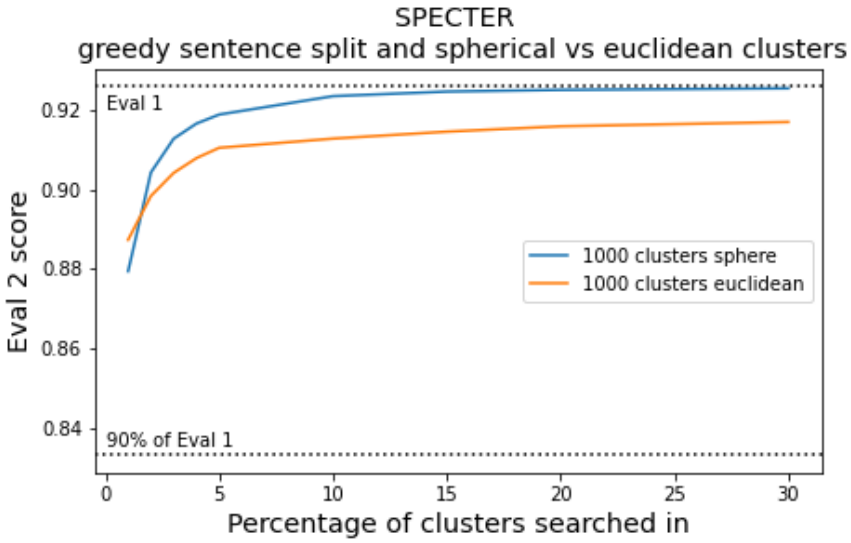

Figure 4.12: A comparison from the second evaluation for the SPECTER model with the greedy sentence split and spherical clustering vs euclidean clustering for 1000 clusters.

In figures 4.13, 4.14 and 4.15 the results for the configuration of model SBERT and greedy sentence split are shown. In figure 4.13 the results for the euclidean clusters are shown, they are separated from one another and have 300 clusters as worst and 1000 clusters as best. All three cluster sizes reach over 90% of the score from evaluation 1 but neither come very close to the full score.

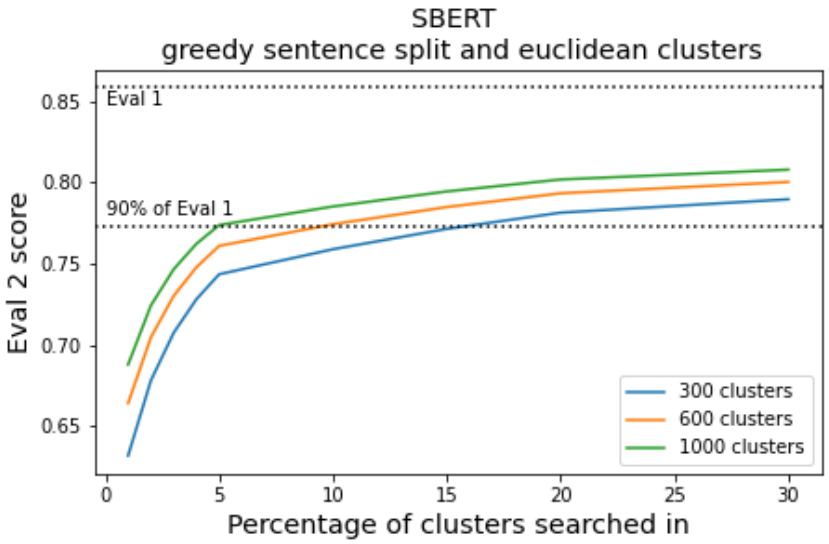

Figure 4.13: The results from the second evaluation for the SBERT model with the greedy sentence split and euclidean clustering.

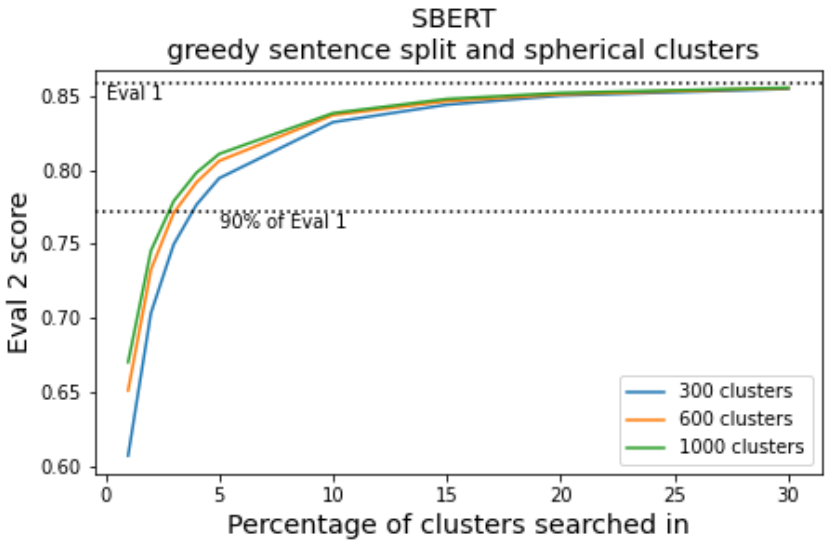

Figure 4.14: The results from the second evaluation for the SBERT model with the greedy sentence split and spherical clustering.

Looking at the results for the spherical clusters in figure 4.14 all sizes are closer together and eventually coincide. However, 1000 clusters perform slightly better than the others at early percentages and comparing this with the the euclidean results for 1000 clusters in figure 4.15 one can see that the spherical clusters perform better. In evaluation 3 and 4 we will therefore use 1000 spherical clusters with a 10% search ratio.

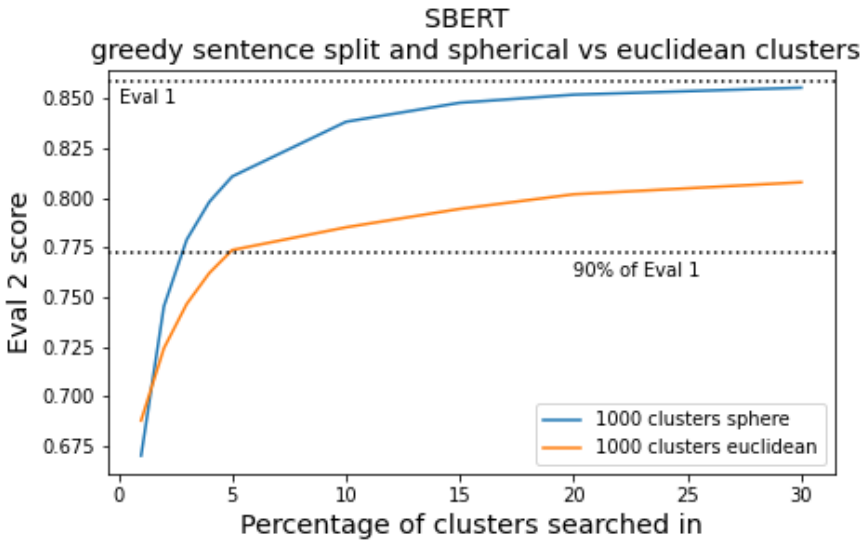

Figure 4.15: A comparison from the second evaluation for the SBERT model with the greedy sentence split and spherical clustering vs euclidean clustering for 1000 clusters.

For the combination with paragraph split and the three models the only cluster configuration is euclidean. Therefore the interesting result in figures 4.16, 4.17 and 4.18 is how many clusters to have and how large percentage of these it is necessary to search in, and not which clustering method that performs best. In figure 4.16 the results for BOW is shown and already at 5% the different cluster sizes are coinciding and almost at the score from evaluation 1. For evaluation 3 and 4 we use 9000 clusters and have a search ratio of 3%.

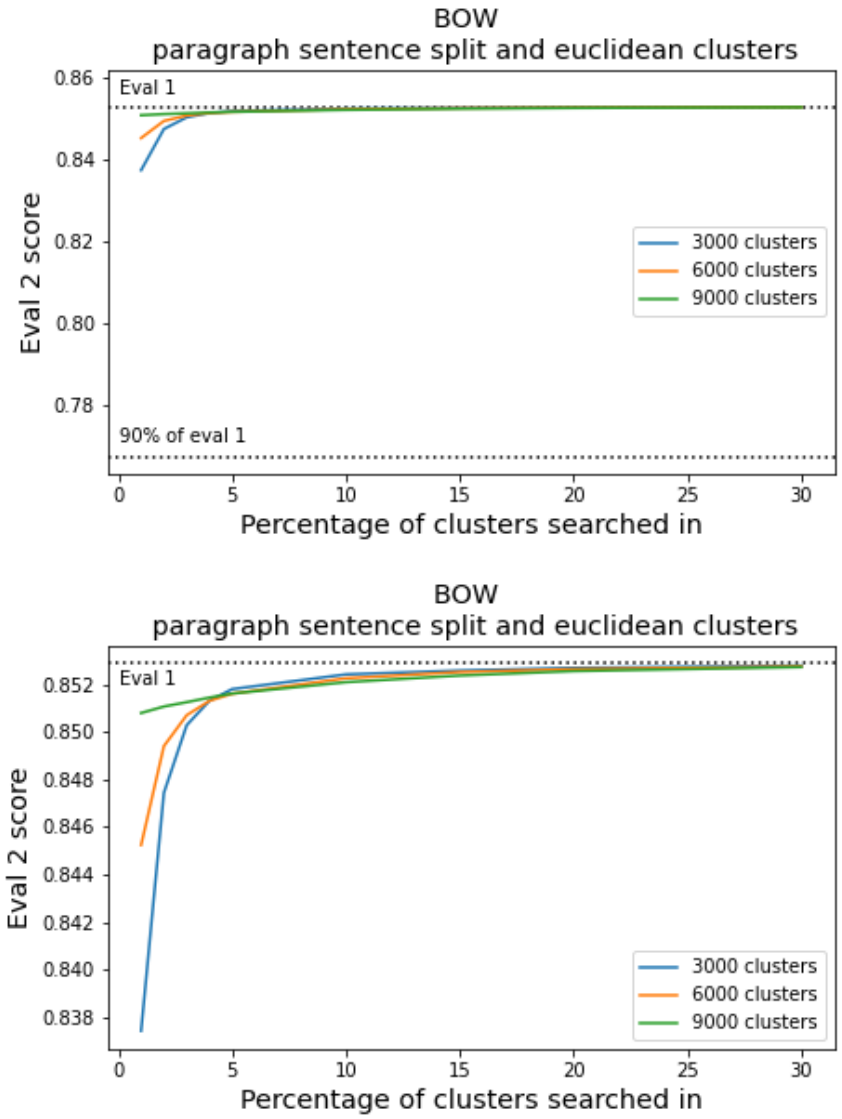

Figure 4.16: The top image in the figure is the results for the second evaluation for BOW with paragraph split. The bottom image in the figure is a zoomed in version of the top figure.

In figure 4.17 the results for SPECTER is shown and here all cluster sizes seems to coincide from the start and lies at the score of evaluation 1. But when zoomed in, we can see that 9000 clusters are actually slightly better and therefore we will use 9000 clusters with a search ratio of 5% in evaluation 3 and 4. We chose this setting because we think it is a good trade off between the score and the computational time.

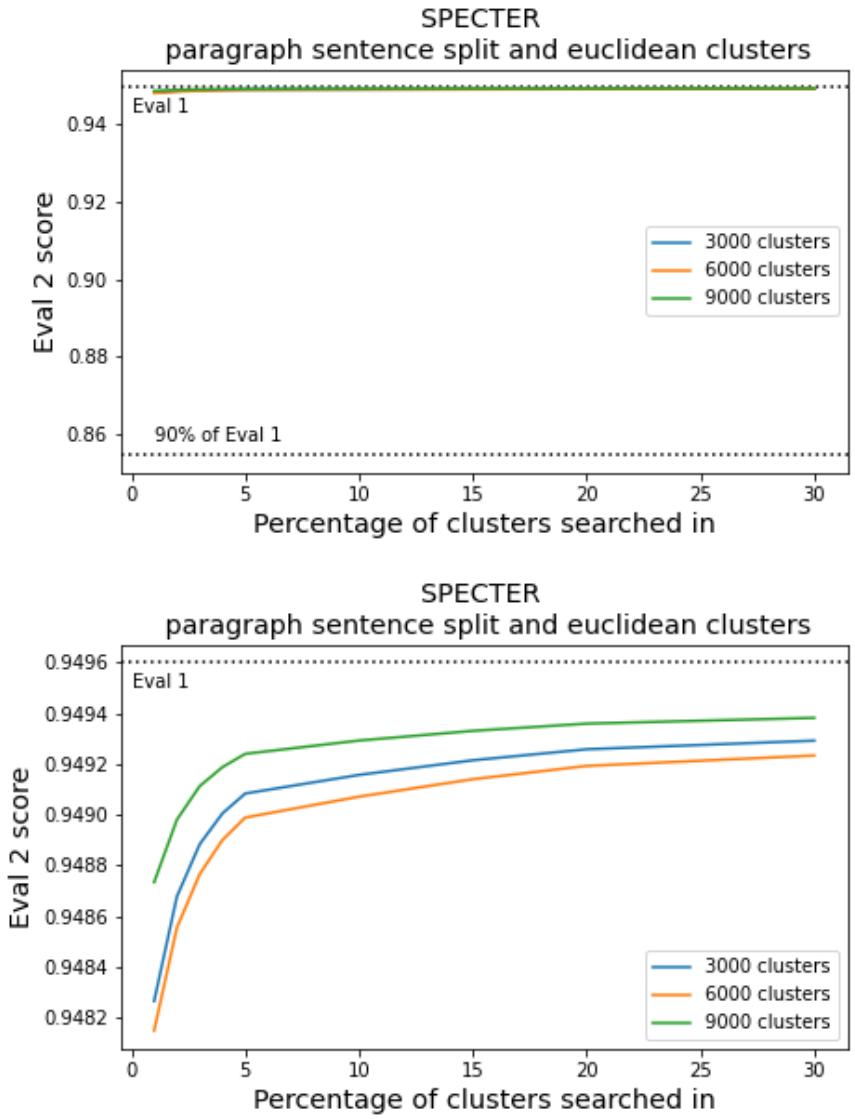

Figure 4.17: The top image in the figure is the results for the second evaluation for SPECTER with paragraph split. The bottom image in the figure is a zoomed in version of the top figure.

In figure 4.18 the results for SBERT is shown and after 5% the curves start to flatten out and after 10% the different cluster sizes have coincided and lies very close to the score of evaluation 1. In the first 5% the three cluster sizes differ somewhat with 3000 clusters as the best and 6000 clusters as the worst. In evaluation 3 and 4 we will use 3000 clusters with a search ratio of 5%.

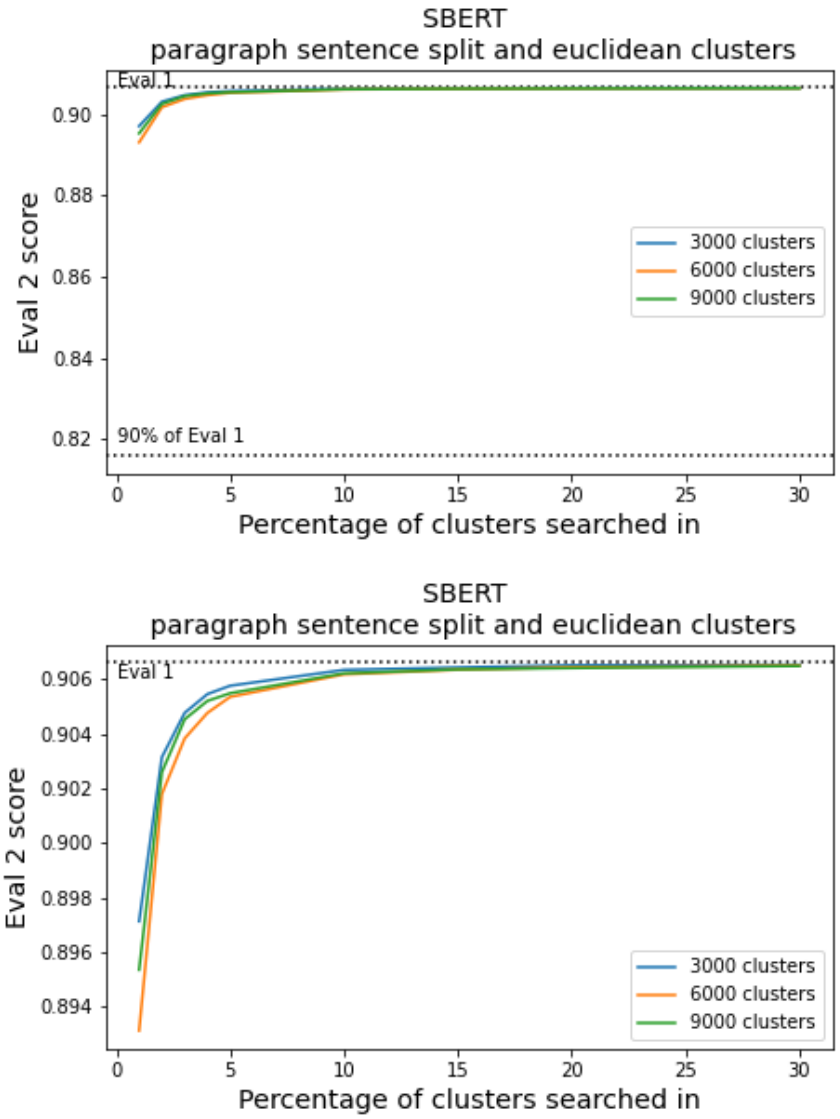

Figure 4.18: The top image in the figure is the results for the second evaluation for SBERT with paragraph split. The bottom image in the figure is a zoomed in version of the top figure.

Due to a mix up of results, two additional configurations made it to evaluation 4. These are SPECTER and SBERT with greedy sentence split and euclidean clustering. For SPECTER we used 600 clusters and searched in 5% of them and for SBERT we had 1000 clusters and searched in 20% of them. The decision for these search settings were taken before we had the correct results for the spherical clustering.

4.3 Evaluation 3: Ranking

In the third evaluation we made a search on a claim from our database and noted how the best match from the same document as the claim was ranked compared to the other matches. The search was performed with the cluster settings that was found in evaluation 2, and it was done on 2000 claims. Some statistics from these searches can be seen in table 4.3 where 'g.' and 'p.' shows if the search was done on the texts that was split with the Greedy sentence split or Paragraph split respectively. 'Best', 'Worst', 'Mean' and 'Median' are qualitative measures of how the best self-matches were ranked and the other statistics are quantitative measures of how the best self-matches were distributed in the rankings of search results.

Statistic\Model	SBERT g.	SBERT p.	SPECTER g.	SPECTER p.	BOW g.	BOW p.
Best	1	1	1	1	1	1
Worst	58124	76121	22279	3217	952	57450
Mean	774.45	264.95	46.08	16.63	3.03	33.31
Median	6	1	1	1	1	1
Position 1	694	1234	1402	1614	1823	1846
Top 10	1053	1512	1741	1828	1974	1954
Top 100	1434	1761	1917	1949	1994	1989
Over 100	506	238	79	51	5	11
Not found	60	1	4	0	1	0

Table 4.3: The values of some statistical measures from the third evaluation.

The median result for the six different configurations is 1 except for 'SBERT greedy' where it is 6. Half of the search results were at the median results or above, which means that in most cases there is higher similarity between a claim and a text from the same document than from other documents. Looking at the mean position of the best self match, 'BOW g.' performed best with a value of 3.03 and the two SBERTs performed much worse than the others with 264.95 for 'para' and 774.45 for 'greedy'. The number of self matches in position 1 are superior for the BOW configurations and they only rank 5 and 11 matches worse than top 100. After BOW, SPECTER performs better than SBERT and for both the BERT-based models we can see that they perform better when using the paragraph split instead of greedy split.

4.4 Evaluation 4: Manual control

In the fourth evaluation we let employees from AWA and Lund University look at the results from seven different searches for each configuration of model and splitting method. The results were distributed so that each person evaluated the search results from three different claims, one for each model. In total seven claims were used for the searches and the mean and median ranking (between 1 and 10) of the results for the different configurations can be seen in table 4.4 where 'g.' and 'p.' stands for greedy sentence split and paragraph split and 'e.' and 's.' stands for euclidean and spherical clustering. In the following results SR stands for the result from the search report.

Result\Config.	BOW g.	BOW p.	SPECTER g.e.	SPECTER g.s.	SPECTER p.	SBERT g.e.	SBERT g.s.	SBERT p.
SR	4.33 / 3.5	7.4 / 7	4.2 / 3	7.2 / 8	6 / 6.5	3.5 / 1.5	5.67 / 6.5	4 / 3
1	5 / 5	4 / 4	5.8 / 6	4.4 / 5	2 / 2	2.25 / 2.5	3.17 / 3	4.83 / 3.5
2	3 / 2	5 / 3	5.2 / 5.5	4.8 / 4	4.67 / 5	5.25 / 5	4.5 / 5	4.5 / 4
3	5.83 / 5.5	4.2 / 4	4 / 3	4.8 / 4	4.33 / 4.5	5.75 / 5.5	4.17 / 3.5	5.67 / 5.5
5	5.5 / 4.5	5.6 / 6	5.6 / 6	5.2 / 5	7.5 / 8.5	6.5 / 6	2.83 / 1.5	5.67 / 5
10	6.67 / 7.5	8.6 / 10	6.6 / 7	4.4 / 5	6.17 / 6	7.5 / 8	6.67 / 6.5	5.67 / 6.5
25	7.33 / 7	4.6 / 6	6.4 / 8	6.4 / 6	5.17 / 5.5	7 / 7	6.5 / 5.5	6.17 / 7
50	6 / 5.5	4.4 / 3	3.8 / 4	7 / 8	6.33 / 6.5	4 / 3	5.5 / 5.5	6 / 6.5
75	4.67 / 5	6.2 / 6	5.8 / 5	5.6 / 6	7 / 7.5	5 / 4.5	9 / 9.5	5 / 5.5
100	6.67 / 6.5	5 / 4	7.6 / 9	5.2 / 4	5.83 / 6	8.25 / 8.5	7 / 7	7.5 / 8.5

Table 4.4: A table showing the (mean / median) results from the fourth evaluation stage for each configuration.

Each of the search results were also classified as either 'X', 'Y' or 'A', and the distribution of these for each search can be seen in figures 4.19-4.26. In these figures the blue column to the left for each search result is the number of answers that were classified as 'X' or 'Y' and therefore considered relevant to the claim. The orange column to the right for each search result is the number of answers classified as 'A' and therefore considered irrelevant to the claim. A more detailed presentation of the individual rankings of search results can be seen in appendix A.

To further analyze the results one can see the results for BOW with the greedy sentence split in figure 4.19. Here, a mixture of relevant and irrelevant classifications throughout the search results can be seen. However, there is a bit higher frequency of relevant documents for the top five search results which was expected due to the higher similarity score towards the search input. The relevant documents at places 50, 75 and 100 gives a notion that within the IPC level E lots of similar documents can be found.

Looking at the results for BOW with paragraph split in figure 4.20 one can see a similar distribution as for the greedy split but here an even more leveled variation of relevant and irrelevant documents over the search results. This is not what we expected but it's still an interesting result.

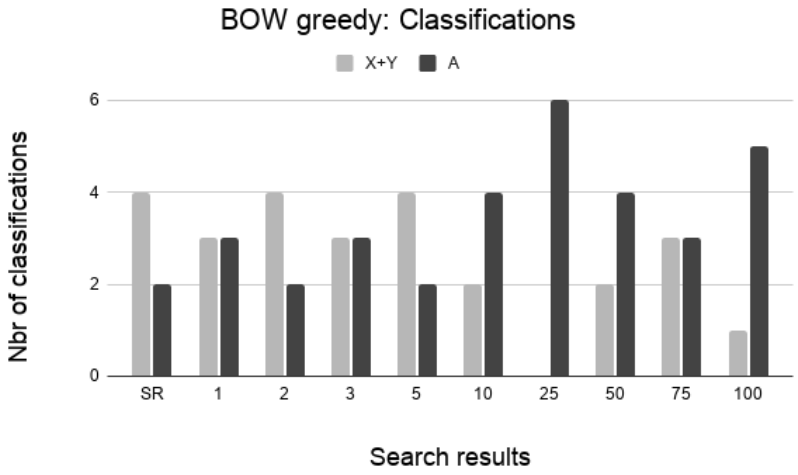

Figure 4.19: A column chart for the BOW greedy classification with two classes, X+Y which are relevant documents and A which are irrelevant documents.

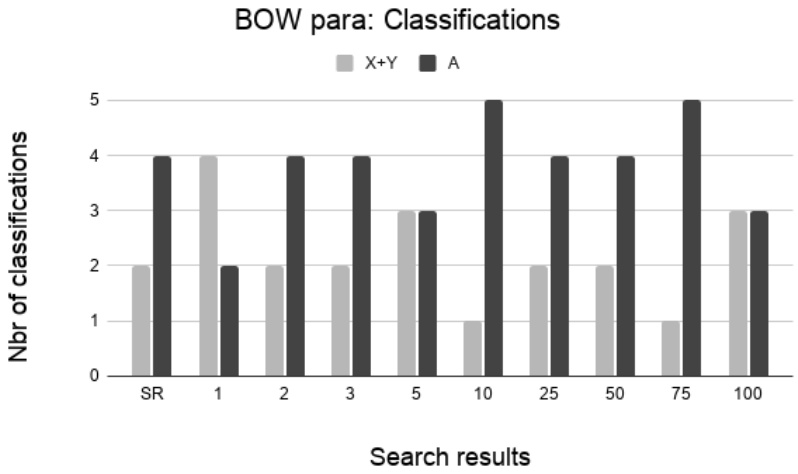

Figure 4.20: A column chart for the BOW para classification with two classes, X+Y which are relevant documents and A which are irrelevant documents.

The results for SPECTER with greedy sentence split and euclidean clusters can be seen in figure 4.21. Here one can see that all search results have at least one case of a relevant passage. What is interesting here is that for each of the top three search results there is only one relevant passage which is lower than the corresponding search results for both BOW configurations where there are up to four relevant passages.

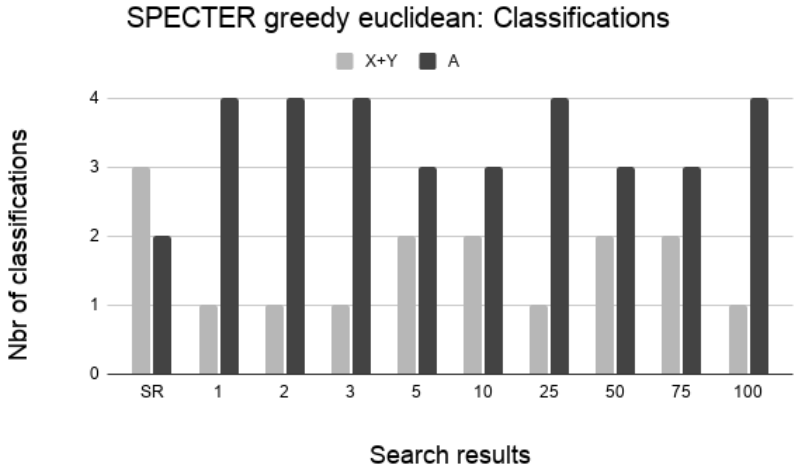

Figure 4.21: A column chart for the SPECTER greedy euclidean classification with two classes, X+Y which are relevant documents and A which are irrelevant documents.

In figure 4.22 the results for SPECTER with greedy sentence split and spherical clusters can be seen. The results show that this configuration is very good at finding relevant passages, especially in the top five result where four out of five were classified as relevant. To see that the people doing these evaluations did not classify texts as more relevant than the people doing the other evaluations we can look at the classification of the search report. Since this column have a similar look as the corresponding for the other configurations, with some relevant and some irrelevant classifications, the good result for the top five scores seems okay and are probably not too influenced by personal preferences.

The last SPECTER results can be seen in figure 4.23, which is for the paragraph split. The results here look a bit more like what we expected from this evaluation with a larger number of relevant passages in the earlier search results and more irrelevant results further back. We did not however, expect the results to stay relevant for so long but it is interesting to see that the possibility of good results is not limited to top 10.

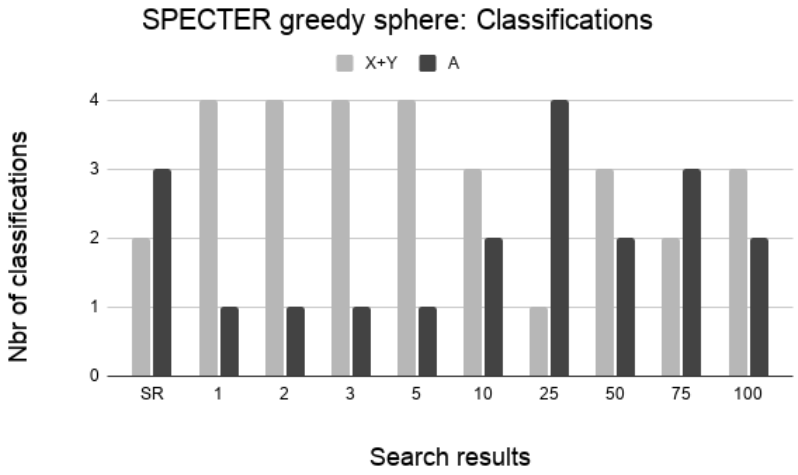

Figure 4.22: A column chart for the SPECTER greedy spherical classification with two classes, X+Y which are relevant documents and A which are irrelevant documents.

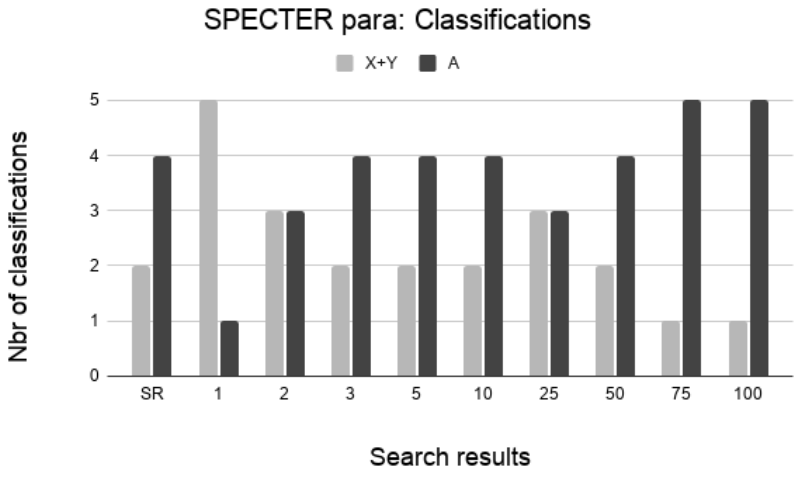

Figure 4.23: A column chart for the SPECTER para classification with two classes, X+Y which are relevant documents and A which are irrelevant documents.

The results for SBERT with greedy sentence split and euclidean clusters can be seen i figure 4.24. These results look a bit different than the others with fewer relevant passages, especially from search result 10-100, where all passages where classified as irrelevant with an exception of search result 50. That the results were poor is not unexpected since we could see in evaluation 2 that this configuration had the worst performance of all greedy sentence split configurations.

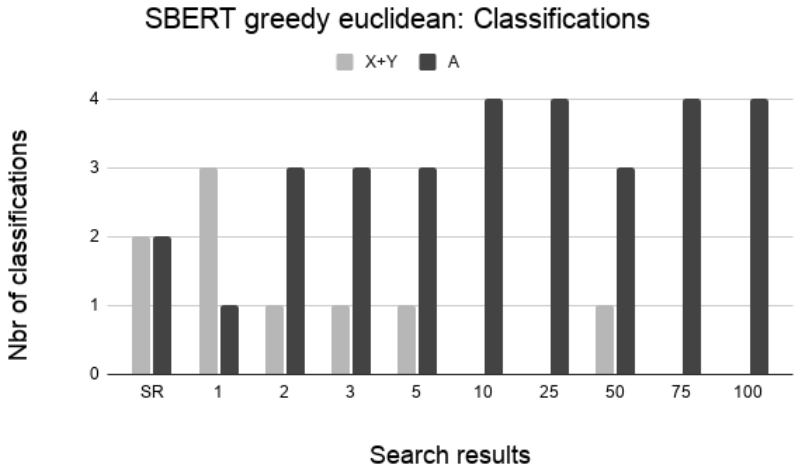

Figure 4.24: A column chart for the SBERT greedy euclidean classification with two classes, X+Y which are relevant documents and A which are irrelevant documents.

The results for SBERT with greedy sentence split and spherical clusters can be seen in figure 4.25. It shows, like the previously analysed results, that we have a variety of relevant passages over the search results.

The final result to analyse for evaluation 4 is the one for SBERT with paragraph split and can be seen in figure 4.26. Here one can see a fairly even level of relevant passages up to the last search result where it has dipped down to one. These results are in line with most of the other configurations where we found at least some relevant documents for every search result.

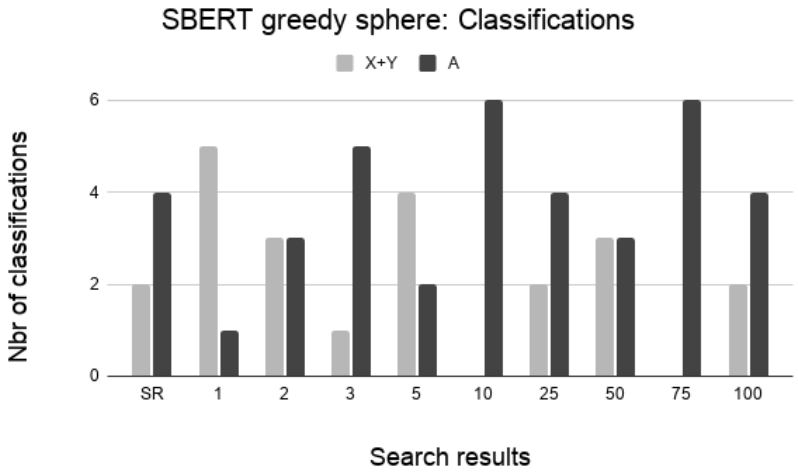

Figure 4.25: A column chart for the SBERT greedy spherical classification with two classes, X+Y which are relevant documents and A which are irrelevant documents.

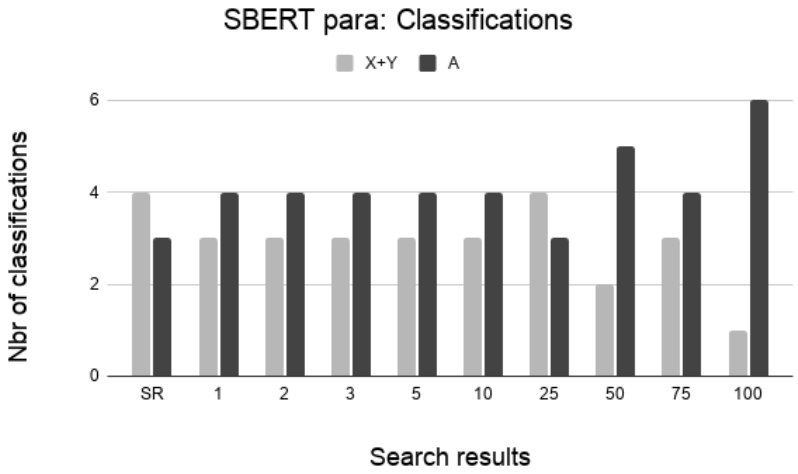

Figure 4.26: A column chart for the SBERT para classification with two classes, X+Y which are relevant documents and A which are irrelevant documents.

Chapter 5
Discussion

5.1 Evaluation 1: Basic semantics capturing

Looking at the results from the first evaluation step in table 4.2, there are a number of things to note. If we start with comparing the first two rows of 'bow10' and 'bow2' when tested on full length descriptions, we can see that there is a higher similarity between a claim and description from the same document for the embeddings based on a vocabulary restricted to 2000 words. The score for the random matches are also a bit higher, but the result is still a larger difference between true and random matches. According to our hypothesis this is a good thing since it suggests that the model can differentiate between a relevant and a random document. This is another advantage of 'bow2' compared to 'bow10', in addition to memory and computation times.

If we examine the data for 'bow2' further, we can see an increase in the 'True mean' score when the text is split into smaller chunks while the 'Random mean' stays at about the same level. For the greedy sentence split this yields the highest difference between true and random across all configurations.

The results for the two BERT-based models are a bit different than BOW. They give the highest similarity scores for the 'True mean', but they also give a lot higher similarity for the random matches which results in the difference between true and random being much lower than for BOW.

A possible explanation for the big differences between true and random matches for BOW is that the embeddings are based on the exact words that are used in the text. Since the description from the same document as the claim is written by the same author, it is likely to be written using the same vocabulary and definitions as the claim. A random text could instead be written by someone else using other words and expressions, and therefore not be detected as 'similar' by BOW.

The BERT-based models on the other hand does not focus on the specific words when creating embeddings of the texts. Since the semantics of the text is in focus, a high score for the true matches are to be expected. A reason that 'Random mean' also get a very high score (higher than 'True mean' for BOW) could be that even though the descriptions are randomly selected, they are still picked from a pool where all documents belong to the same IPC level, and therefore a similarity to some degree is very likely.

According to the values in the last column of figure 4.2, one could believe that

SBERT is to be preferred due to a bigger separation between relevant and random results. But if we study figures 4.1 and 4.2 we can see that the distribution of results are more centered around the mean values for SPECTER compared with SBERT which suggests that there is less overlap between the two categories for SPECTER.

It is important to remember that we don't actually have the ground truth of what the different scores from the first evaluation should be. We can only reason and make assumptions according to our hypothesis that a better model has a greater divergence between a true match and a random match.

5.2 Evaluation 2: Clustering

From the cluster studies made on the SBERT models shown in the figures 4.4, 4.5 and 4.6 we chose that the range we wanted to explore further was between 300 and 1000 clusters for the greedy sentence splits and between 3000 and 9000 for the paragraph splits. We based this decision on the slope of the decreasing inertia. As long as the slope is steep we believe there can be benefits of adding more clusters. The results in figure 4.6 does not look like the other two figures. Since the algorithm is not guaranteed to converge to a global minimum and each new number of clusters will generate different starting points we believe this could lead to two different levels of local minimum. Nevertheless, both levels decrease in the same general manner when more clusters are added.

Another thing that was noticed during this part of the work is that for the amount of data when using the paragraph splits it was not possible to use the spherical cluster method. This was due to the memory usage being too large when converting the data to the necessary sparse structure. An attempt to get around the need for sparse data structure was performed by using an adapted version of scikit-learn's k-means algorithm. However, this was still not feasible due to the long computation time and we aborted the attempt after approximately 12 hours without the first epoch being completed. To be able to perform this type of clustering with the models used in this thesis it would be necessary to implement a mini-batch k-means as used for the euclidean clusters.

Going forward to the results from the second evaluation, the figures 4.7, 4.8 and 4.9 shows the bag-of-words model with the greedy sentence split, which is used as a baseline for the other models. In these figures one can see that even though we had results from both euclidean and spherical clusters that reached up to the score from evaluation 1, the spherical cluster size of 1000 reach up fastest. This is what was expected, we compare the embeddings for the claims and the description parts with the cosine similarity which is the same measure that spherical clusters use.

The results for SPECTER yields a similar result when looking in figures 4.10, 4.11 and 4.12. As described in the results section we have the spherical clusters outperforming the euclidean clusters. In figure 4.11 one can see that at 10% we have gotten very close to the score from evaluation 1 and searching in a larger amount of clusters would only marginally increase the score. The trade off between computation time and score gain has in our opinion reached a good level and that

is why we chose the 10% ratio going forward.

The last results on the second evaluation for the documents divided with the greedy split method are the ones for SBERT. These can be seen in figures 4.13, 4.14 and 4.15. Here the separation between the euclidean and spherical clusters is even larger than for the SPECTER model. However, the spherical clusters seems to not perform as well because none of the cluster sizes completely reach up to the score from evaluation 1 but they all come very close at 30%. Even though the score does not reach all the way we still feel that 10% of the clusters is enough to search through for the size 1000 clusters. This is because 1000 clusters is well past the 90% of evaluation 1 score and we want to easily compare with the SPECTER method, which at this stage seems to have the upper hand, since both will search in 10%.

Continuing to the results for the paragraph split descriptions we have seen that they all perform outstandingly well in finding the most similar piece from a claim's own description. This is probably because of two reasons. Firstly, the segments of texts are smaller and therefore they contain less noise and are more topically homogeneous. Secondly, the number of clusters they are divided into are many, and therefore smaller, which leads to the cluster centers being more representative of the embeddings contained in the clusters. This makes it easier to find the relevant clusters for a claim. It is interesting that for SBERT, 3000 clusters perform slightly better than the others in the first 10%. A separation can also be seen in SPECTER where all curves start at a higher level but in the zoomed in version there is a clear distinction between all sizes. BOW on the other hand has for the first 5% a clear separation between the different cluster numbers with 1000 clusters clearly better from the start, even though 3000 clusters are slightly better after 5%

Over all the results behaved as believed, especially the spherical clusters where more clusters generated a better result. The euclidean clusters however surprised in some cases for the greedy split where more clusters did not always behave better like for SPECTER in figure 4.10 where there is a large gap between 300 clusters and the others, and where 600 and 1000 clusters go close together with 600 as the best number of clusters. The reason for this is unknown, and further investigations needs to be done to understand the cause of this. Another interesting finding was how well the paragraph split descriptions performed for all methods which indicates that these might be better suited at finding special similarities between texts. A probable cause for this is that the text segments in the paragraph split is more homogeneous topic wise, and the semantics of the over all shorter segments is better captured by the models.

5.3 Evaluation 3: Ranking

Some things to note in the results from evaluation 3 is that the BOW results are superior to the BERT-based models in the number of matches that position themselves on first place in the search and in the number of documents that were not found. As discussed earlier we believe the BOW model has a strong bias towards texts written by the same author on the same topic as the input claim

due to choice of words and phrases being similar.

For SBERT and BOW we see that the worst result (of those that were found) are much worse for paragraph split than for greedy sentence split. This could be due to there being many more texts in the search, so that even though the distributions are similar there will be a higher value for paragraph split. But looking at SPECTER we see that the worst result is much better for para than for greedy, which is the opposite. This could be a sign that SPECTER is in fact much better at giving 'meaningful' embeddings to shorter texts, possibly due to there being less noise. This seems to be the case for both BERT-based models when the mean value and distributions of top 1, 10 and 100 are studied.

5.4 Evaluation 4: Manual control

The first thing to note about the fourth evaluation is that the data we gathered is very limited. We set out to get evaluations on 7 searches, based on different claims, for every configuration, which would mean a total of 42 evaluations. But due to our mix-up of results we ended up asking for 56. There were a limited number of employees that had both the time and interest to do the evaluations and we got 45 results back a bit unevenly distributed between the configurations.

With the limited data it is difficult to draw any certain conclusions. Looking at the distribution of the internal rankings of search results, a slight trend can be seen that the first search hits (1, 2, 3) are often a bit better than the last ones (75, 100). A comment from persons doing the evaluations were that often 6 or even more results were all unrelated or bad and thus difficult to rank among themselves. This could lead to a random internal ranking of those hits.

Some of the claims that we chose for the evaluation were very short. This led to the people evaluating sometimes finding it difficult to gather enough information from the claim to be able to evaluate the search results. We believe this could be a reason that sometimes not even the text from the search report was classified as 'X' or 'Y'. Other reasons for this could be that the claim (input) has been machine translated which affects the language, or that the 'X' in the search report can be based on multiple paragraphs in the document but we only had one of these in the evaluation.

We think that the classifications of the search results as 'X', 'Y' and 'A' are more interesting than the internal rankings of the results. Here we can see that we found relevant texts across most of our search results. The numbers of 'X' and 'Y' classifications varied a lot between the different searches, and in some cases this could be due to our models' limited capability. But another likely reason is that our search is done in a very limited subset of patents (57 453 documents) belonging to a single IPC class. This means that there is a big risk of not finding relevant documents since they are not included in our subset. The reason for presenting the 'X' and 'Y' documents as a joint category in the classification charts is that they both indicate some relevance in the text as opposed to 'A'. But since the texts are relatively small, an 'X' might turn into a 'Y' when the whole document is studied, and vice versa.

The results are not conclusive enough to point out one configuration as bet-

ter than the rest, but they indicate that these methods could be of interest and contribute with a new non-human perspective in invalidity searches for patents.

5.5 Time complexity and memory usage

Here we will make a few comments about the complexity of the computations and how much memory that is needed for the database.

5.5.1 Creating embeddings

For the BOW models the computation complexity for creating the embeddings with a predefined vocabulary is O(n*v*l), where n is the number of texts, v is the size of the vocabulary and l is the length of the texts. For the BERT-based models the computation complexity is unknown.

5.5.2 Clustering

The basic k-means clustering method has a time complexity of O(t*k*n*d) where t is the number of iterations, k is the number of centroids, n is the number of data points and d is the dimension of the data points.

While the mini-batch k-means theoretically has O(infinity) since it never converges, it is still able to speed up the algorithm significantly since we can set a threshold value for when to stop. When we clustered the embeddings for the paragraph splits, we had to use the euclidean clustering method since the spherical clustering methods we found did not implement mini-batching and thus took too long to finish.

5.5.3 Search

The search consists of first creating an embedding for the input and then comparing it with the embeddings in the closest clusters. This means there will be k+p*n comparisons. The list of similarities to the different embeddings will also be sorted which is of O(p*n*log(p*n)).

The complexity for a search with our program is approximately O(d*p*n*log(p*n)) where d is the dimension of the embeddings, p is the percentage of the clusters that are searched and n is the number of data points.

5.5.4 Storage

In the database there are two big bulks of data being stored, the text parts and their corresponding embeddings. The text parts are saved as text files and are not very big and memory will scale up as O(n) where n is the number of documents. The space complexity for the embeddings is O(n*d), where n is the number of embeddings and d is the embedding dimension, and they are saved as pickle files.

5.5.5 Scaling up

The two parameters that affects the complexity that differs between our methods are n and d. For n, paragraph split yields about 4.4 times as many data points as the greedy sentence split. For d, the dimension of the BOW embeddings is 2000 while it is only 768 for SBERT and SPECTER. This means that when more documents are added to the database the memory scales up almost three times as fast for BOW compared to the BERT-based models and 4.4 times as fast for paragraph split compared to greedy sentence split. All the computations will also take a longer time when BOW is used due to the larger dimension.

With even better and dedicated hardware together with optimized code, it is possible that it could be feasible to scale up to a more extensive database of patents than we could due to the limited time frame of our thesis. Then the results could perhaps be even better as more documents are included in the search.

5.6 Future work

This thesis shows the potential of segmented whole text search in patents and future works on this topic could be to develop and test spherical clustering for the paragraph split documents. This would also be beneficial if the database where to be expanded into more IPC classes and there generally would be more documents. Another future work could be to train or fine tune a model specifically for patent documents and see if it would have an impact on the result. In the evaluation part a future work could be to develop better test and measurement to evaluate the models. In our thesis the only control evaluation where the results for a claim was evaluated was in the manual part. This part is hard to quantify and it would therefore be of use to have a similar evaluation made automatic or semi-automatic. It would also be interesting to include the images of the patents in the comparison to better understand the text as well as exploring the possibility of trans language search if a patent has versions in other languages.

Chapter 6
Conclusion

The goal with this thesis was to explore the possibilities of doing full-text search in patent documents and how the search is affected by different clustering and segmentation methods.

From the first evaluation we found that segmented texts are better suited for creating the embeddings as they are more semantically homogeneous and less noisy. Drawbacks of the full-text embeddings are that they include more noise and aren't able to pinpoint where in a long text the similarities are found. Due to the limit of 512 tokens the BERT-based models we examined it was not possible do the full-text embeddings.

The segmentation methods leads to a lot more texts/embeddings but we found that clustering worked very well and by searching through only a fraction of 5-10% we could speed up the search with a very low loss of accuracy. The spherical clustering method outperformed the euclidean clustering in every instance.

Both the greedy sentence split and the paragraph split gave good results in our evaluations. There seem to be a slight advantage to the paragraph split, possibly due to the semantics of shorter text passages being better captured by our models and also being more homogeneous due to following a segmentation done by the author. In that regard the paragraph split is also more efficient computation wise, whereas the greedy sentence split has the advantage of creating fewer texts/embeddings which is more efficient memory wise.

The automatic testing is done in evaluations 1 and 2, where a claim is tested against the description of the same document, and in evaluation 3 where the self-matches are compared with other search results. This gives us a limited understanding about the performance in a 'real' search and also there is no ground truth to how the models should perform. In the fourth evaluation a better indication of how the models performed in a real search was given. The data there was limited and we can't conclude which configuration works the best, but we can say that the results indicate that the methods have potential to find interesting documents and could be used for finding a list of documents to further examine when doing an invalidity search.

Overall we have used a limited set of data and it would be interesting to see more experiments done for more documents and with improved testing methods. For scaling up to a really large database we think that fewer and shorter embeddings would be preferred, i.e. greedy sentence split and one of the BERT-based models.

References

[1] Helmers L, Horn F, Biegler F, Oppermann T, Müller K-R (2019) Automating the search for a patent's prior art with a full text similarity search. PLoS ONE 14(3): e0212103. doi.org/10.1371/journal.pone.0212103

[2] Jansson H, Navrozidis J (2020). Using natural language processing to identify similar patent documents (Master's thesis), lup.lub.lu.se/luur/download?func=downloadFile&recordOId=9008699&fileOId=9026407

[3] Habibi M, Rheinlaender A, Thielemann W, Adams R, Fischer P, Krolkiewicz S, Wiegandt D, Leser U (2019) PatSeg: A Sequential Patent Segmentation Approach, doi.org/10.1016/j.bdr.2020.100133

[4] Espacenet patent search, worldwide.espacenet.com/

[5] Lexis Nexis, www.lexisnexis.com

[6] Natural Language Toolkit, www.nltk.org

[7] Devlin J, Chang M-W, Lee K, Toutanova K (2018) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, arxiv.org/pdf/1810.04805.pdf

[8] Reimers N, Gurevych I (2019) Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, https://arxiv.org/pdf/1908.10084.pdf

[9] Bowman S R, Angeli G, Potts C, Manning C D (2015) A large annotated corpus for learning natural language inference, www.aclweb.org/anthology/D15-1075

[10] Williams A, Nangia N, Bowman S (2018) A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference, www.aclweb.org/anthology/N18-1101

[11] Cohan A, Feldman S, Beltagy I, Downey D, Weld D S (2020) SPECTER: Document-level Representation Learning using Citation-informed Transformers, arxiv.org/pdf/2004.07180.pdf

[12] Arthur s, VAssilvitskii S (2007) k-means++: The Advangages of Careful Seeding, https://theory.stanford.edu/~sergei/papers/kMeansPP-soda.pdf

[13] Sci-kit learn: Mini-Batch K-Means Clustering, scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html

[14] Sci-kit learn scikit-learn.org

[15] Extensible Markup Language, www.w3.org/standards/xml/core

[16] Hearst M A (1997) TextTiling: Segmenting Text into Multi-paragraph Subtopic Passages , www.aclweb.org/anthology/J97-1003.pdf

Appendix A

Extra material on evaluation 4

A.1 Forms for results of evaluation 4

For every search the results were randomly assigned a letter from A to J and the following two tables were filled:

	1	2	3	4	5	6	7	8	9	10
A										
B										
C										
D										
E										
F										
G										
H										
I										
J										

Table A.1: The form for ranking the search results against themselves.

	X	Y	A
A			
B			
C			
D			
E			
F			
G			
H			
I			
J			

Table A.2: The form for classifying each search result as 'X', 'Y' or 'A'.

A.2 List of claims used in evaluation 4

A search on each of the following claims was done for each configuration in the fourth evaluation

Claim 1: A vehicle control device (4) including a communication means (11, 11a, 12, 12a, 13) for performing wireless communication with a portable device (2) and a control means (14) for controlling the communication performed by the communication means (11, 11a, 12, 12a, 13), communicating with the portable device (2) to authenticate the portable device (2), and entering an enable mode for enabling starting of an engine (23) of the vehicle (3) when the portable device (2) is authenticated, the vehicle control device (4) being characterized in that : the communication means (11, 11a, 12, 12a, 13) is one of a plurality of communication means respectively comprising a first transmission antenna (11a) in a center console (3a) and a second transmission antenna (12a) in a rear seat (3b) and forming a plurality of different communicable regions (A1, A2) in a passenger compartment; the control means (14) is configured to detect whether or not a back door (5e) of the vehicle (3) is open and refrain from entering the enable mode when that the back door (5e) is open; and the control means (14), when detecting that the back door (5e) is open stops communication between the portable device (2) and the second transmission antenna (12a) as the communication means (11, 11a,) of which the communicable region (A2) corresponds to the open back door (5e), while the control means (14) enables the starting of the engine even if the back door (5e) is open when receiving an ID code from the portable device (2) in response to a request signal transmitted from the first transmission antenna (11a).

Claim 2: A borehole apparatus (1) capable of autonomously estimating its position in a borehole and autonomously controlling the actions of a downhole tool (5) located in the borehole, the apparatus comprising: a body; at least one measurement device (4) capable of measuring a parameter of the borehole or the distance travelled by the device; a computer system located in the body, the computer system comprising; a processor (2) arranged to receive data from the measurement device and to calculate the position of the apparatus in the borehole; and a data storage device capable of storing data that have been processed by the processor; and a power system (3); wherein the computer system is configured to process the data gathered from the measurement device to estimate the position of the borehole apparatus using a Bayesian approach and characterized in that the data storage device is capable of storing instructions to control the actions of the downhole tool and that the borehole apparatus (1) is configured to provide output signals to control an action of the downhole tool (5), the action being dependant on the position of the apparatus (1) in the borehole and the instructions stored by the data storage device.

Claim 3: Construction and installation of a rainbow highway on existing roads or new roads by applying paint in rainbow colors (red, orange, yellow, green, blue, indigo and violet) to such roads.

Claim 4: A sound attenuating laminated panel having a laminate structure, comprising a decorative layer, a core layer, a backing layer and a wear layer and wherein one or more layers comprised in said laminate structure, has been treated with an elastomeric material, so as to form a treated layer, in order to reduce the noise generated by said panel, wherein the core layer is an MDF or HDF wood core, wherein said treated layer has been formed by pre- impregnating a selected layer with said elastomeric material prior to production of said laminated panel, and wherein said selected layer is said decorative layer, backing layer and/or wear layer, characterized in that said selected layer is a paper layer that has been pre-impregnated with an elastomeric polymer that penetrates into the paper and in that said elastomeric material is selected from polyurethane, polyolefin (TPO), modified melamine-based thermoset resin, ESI - ethylene styrene interpolymer or any of the styrene acrylic copolymers, rubber based material; NBR (nitrile butadiene), SBR (styrene butadiene), or CR (chloroprene); or a carboxylated, natural or synthetic latexes, and in that said wear layer comprises a paper layer which has been preimpregnated with said elastomeric material.

Claim 5: Wall covering having a relief and formed from a plurality of strips applied to a wall next to one another, wherein each of said strips comprises a base (2), a layer of heat-expandable material (5) applied to said base (2), wherein this layer has a pattern of expanded and less expanded or unexpanded locations, characterized in that said pattern is different for each strip, wherein said different patterns create a relief pattern in said wall covering exceeding the width of one single strip and extending across various strips.

Claim 6: A method for designing a cement system for placement in a well having

a borehole penetrating subterranean formations, at least one casing string and at least one cement sheath, comprising: (i) selecting a candidate cement system such that the cement sheath has a known Young's modulus, Poisson's ratio, tensile strength and a variable linear thermal expansion coefficient; (ii) determining the well geometry and casing geometry; (iii) using a computer simulator to determine cement-sheath integrity and tangential stress upon application of heat, pressure or both in the well; (iv) if the simulation indicates cement-sheath failure, modifying the candidate cement system to adjust the variable thermal expansion coefficient, and repeating step iii.; and (v) if no failure is indicated, selecting the candidate cement system as a final design.

Claim 7: A sound insulator comprising a sound-absorbent core (1) within a flexible membrane (2) characterised in that part of the external surface of the resultant composite structure is curved around a longitudinal axis.

A.3 Extra images from evaluation 4 results

In this section some extra figures from the results of evaluation 4 is presented. In the figures the answers made by our participants are marked as a grey dot. Grey dots can lie on top of each other and give the impression of fewer answers. The median of the answers are shown with a black dot and the mean of the answers is shown with a black 'X'.

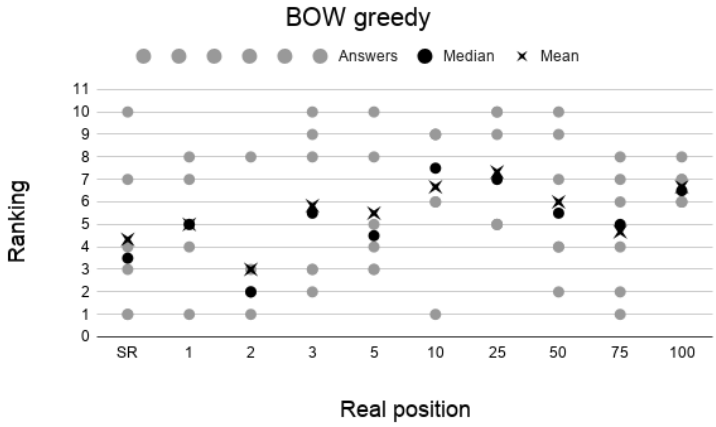

Figure A.1: The result for BOW greedy where the answers are marked as grey dots, the median as a black dot and the mean as a black 'X'.

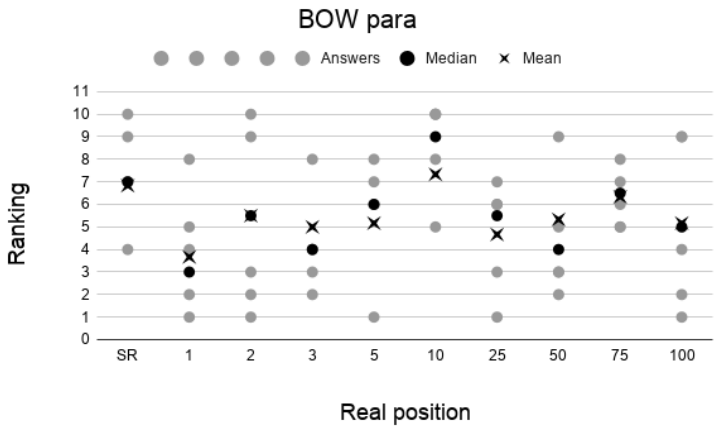

Figure A.2: The result for BOW para where the answers are marked as grey dots, the median as a black dot and the mean as a black 'X'.

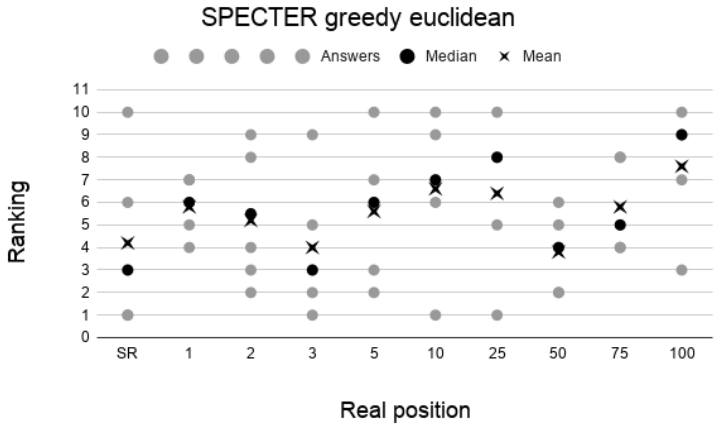

Figure A.3: The result for SPECTER greedy euclidean where the answers are marked as grey dots, the median as a black dot and the mean as a black 'X'.

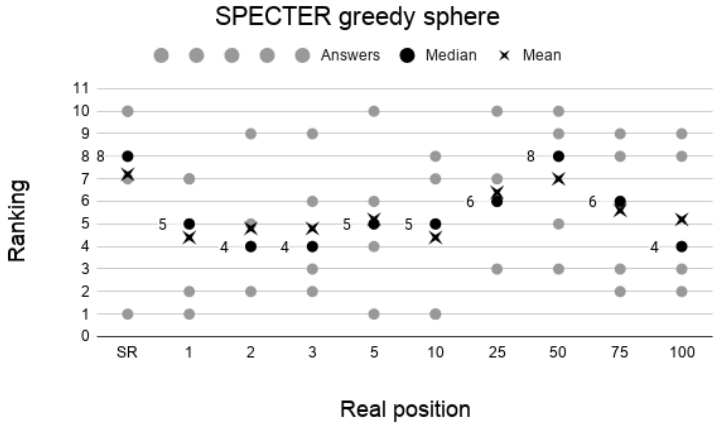

Figure A.4: The result for SPECTER greedy sphere where the answers are marked as grey dots, the median as a black dot and the mean as a black 'X'.

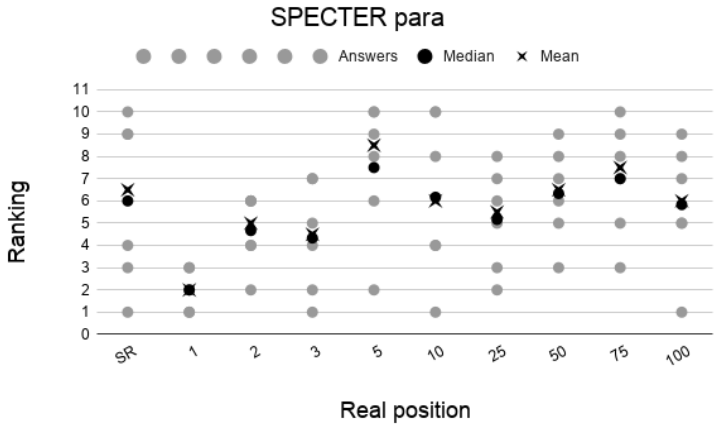

Figure A.5: The result for SPECTER para where the answers are marked as grey dots, the median as a black dot and the mean as a black 'X'.

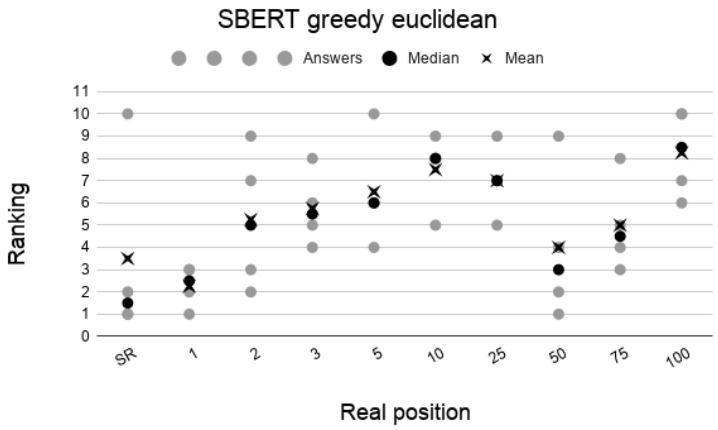

Figure A.6: The result for SBERT greedy euclidean where the answers are marked as grey dots, the median as a black dot and the mean as a black 'X'.

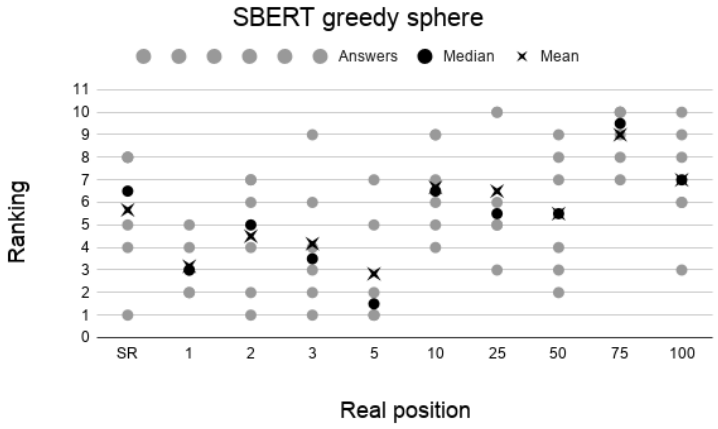

Figure A.7: The result for SBERT greedy sphere where the answers are marked as grey dots, the median as a black dot and the mean as a black 'X'.

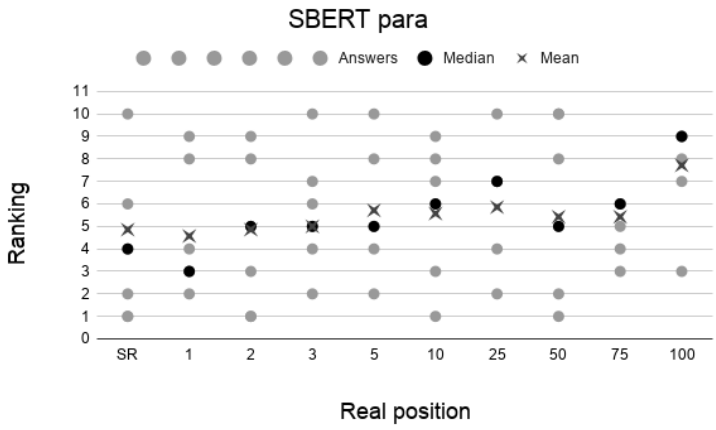

Figure A.8: The result for SBERT para where the answers are marked as grey dots, the median as a black dot and the mean as a black 'X'.

Printed by Tryckeriet i E-huset, Lund 2021

LUND
UNIVERSITY

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2021-837
http://www.eit.lth.se