MASTER'S THESIS 2022

Interactive Iterative Patent Search

Daniel Jogstad

ISSN 1650-2884
LU-CS-EX: 2022-11

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2022-11

Interactive Iterative Patent Search

Interaktiv Iterativ Patentsökning

Daniel Jogstad

Interactive Iterative Patent Search

Daniel Jogstad
tfy14djo@student.lu.se

March 23, 2022

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisors: Pierre Nugues, pierre.nugues@cs.lth.se
Fredrik Edman, fredrik.edman@innovation.lu.se
Henrik Benckert, henrik@mindified.com

Examiner: Jacek Malec, jacek.malec@cs.lth.se

Abstract

Searching for prior art in databases with millions of patent documents is a very time-consuming process. Programs use natural language processing, NLP, as an efficient way of finding similar documents to an input text to solve this. In the latest years, algorithms based on neural networks have become the brightest shining stars among these programs. Through NLP, they translate the words and sentences into numerical vectors with which they, in some sense, can describe the general meaning of a text.

In this Master's thesis, we have investigated the use of Sentence-BERT, a neural network computing a vector representation of a sentence, together with user input for gradual improvement. In several iterations, test persons have rated and divided the top results into good and bad matches that the program uses for re-ranking the search results. The final results are evaluated for patent quality, ordering of the patents, and how the method can heighten the rating of some good patents far down the list, at first dismissed as top candidates. Our results show that it is possible, through simple mathematical operations, to implement an interactive, iterative patent search that improves the initial search results of the neural network.

Keywords: natural language processing, semantic similarity search, artificial intelligence, patent, sentence embedding, iterative, interactive

Acknowledgements

This Master's thesis has been made possible thanks to several persons whose contributions to this project cannot be emphasized enough. First and foremost, I would like to express my gratitude towards my supervisor Henrik Benckert for the opportunity to do this project in collaboration with Mindified. I would like to thank him and the other people at Mindified for all the help I have received and the many things I have learned.

I would also like my supervisor Fredrik Edman for many useful insights as well as for being a crucial cornerstone in tackling the many practical issues this Master's thesis has faced.

Thirdly I would like to thank my supervisor Pierre Nugues for having patience with me when I have strayed from what has been most important and focused on other things, for giving me a nudge in the right direction, and for giving me plenty of useful articles and material to read.

Credit should also be given to the wonderful souls who offered their free time just to test my models and type numbers on a screen for several hours.

Lastly, I would like to give my sincere thanks to my friends, family, and my wonderful girlfriend for being who they are.

Contents

Chapter 1
Introduction

1.1 Problem

Today, large parts of the value of many companies lie in their intellectual properties and patents are important to protect them. There are many steps involved in the patenting process. Applying for a patent starts with a research stage. This step is crucial, because after the application has been done, third parties can object to the application. It is therefore essential for the companies to be able to defend their patents against objections. Likewise, a company could also be interested in objecting to rival patent applications if they do not have novelty or are not inventive solutions, for example if they are based on already existing techniques or patents. In any case, both the defending and the objecting side must use well-founded arguments, to which complete knowledge of prior arts is required. This is where doing similarity searches in patent databases comes in.

There are many reasons why one would like to search for similarities between documents in patent databases. When applying for a patent, the invention has to be a novel idea with an inventive step. That means that prior art, i.e. earlier works with the same ideas must not exist. If there is an interest in patenting an invention, it is therefore highly recommended to first search for earlier works in the same field before conducting the costly and time-consuming patent application process. If there is knowledge about some existing patents, a similarity search for these may be carried out, thus decreasing the likelihood of not having enough information about prior art.

Searching for similarities between documents can be done in many ways, but a cornerstone for most search methods is the use of keywords and/or phrases with which one can find documents with similar content. This may be problematic and give incomplete search results since many similar inventions can be described with different words. Even if one in some way could account for synonyms of words, there is no escaping the fact that something can be described using completely different words.

A way of solving this is to search for words or sentences that in their context have the

same meaning. In theory, this is a simple task for a human, but more difficult for a computer program. However, as we are working with databases with possibly millions of documents, the latter is preferred. Through the use of neural networks, algorithms can be created that can capture the general context of a text. However, while some of the results the programs find similar to an original document are good outputs for the user, we often get some documents that may be structured similarly and use similar words but do not really capture the context we want. This Master's thesis will explore ways of solving this problem, using feedback data from the user iteratively to improve the original search.

1.2 Previous work

Previous works have been written on the subject of using neural networks for semantic similarity searches among text documents. Among them, Reimers and Gurevych (2019) introduced the Sentence-BERT (SBERT) network, which we use in this Master's thesis, optimized for finding similarities between two sentences.

Several papers prior to this work have explored the use of BERT-based networks when working with patent document databases. Among other things, the papers differ in patent document representation for embedding creation. One way is to represent them with each patent's claims. For example, Lee and Hsiang (2020) classified patents belonging to specific CPC (Cooperative Patent Classification, a method of categorizing patents) subgroups with a fine-tuned BERT network.

A work that is closely related to ours is the Master's thesis by Navrozidis and Jansson (2020), where the authors explored different methods in NLP, including SBERT, for finding similar documents in a large patent database. In their work, as in ours, they use concatenated texts from each patent's title and abstract for document representation. However, while they evaluate the effectiveness of the network, we are trying to improve the network's output by user evaluation of the output and re-ranking the results.

Re-ranking with the use of BERT-based networks has been done before, for example by Khattab and Zaharia (2020). However, we believe our work to be novel in using user input together with SBERT embeddings for re-ranking.

1.3 Goal of project

The goal of this project is to explore the concept of interactive iterative patent search and try to find if it can improve the search results of an established NLP model such as SBERT. The search is associated with a re-ranking process by altering the original SBERT ratings of matches to input patents. This report describes our approach and evaluation process and reflects upon our findings.

Chapter 2
Theory

In this chapter, we will explain what needs to be known before proceeding to the experimental setup. Here we will talk about the general idea behind *interactive iterative patent search*. We will give some examples on how to represent text documents as numerical vectors and we will go through how neural networks can be used for semantic comparison.

With this knowledge, we will look closer at the models for interactive iterative patent search that we have used in this Master's thesis. Lastly, we will go through some statistical concepts and evaluations methods for testing.

2.1 Interactive Iterative Patent Search

Let us assume that we are to search through a database of patent documents and that we want to find similar matches to an input text. For this purpose, we use so-called word and sentence embeddings for document representation and cosine similarity as a measurement, which will be described more in detail in Sect. 2.5.

We can compare each of the documents in the database with our input to find the best matches according to the network that created the embeddings. However, the results may not all be satisfactory. The network could suggest matches that, while language-wise being very similar, differ in the specific topic that we want to find matches for.

An example could be searching for similarities to a design for a spark plug, using a patent for a spark plug in a lawnmower as input. Many of the results might be weighted in favor of lawnmower patents in general, instead of spark plug patents as the example in Table 2.1.

If that is the case, we could use the user's opinion on the better and worse results in the top matches in a feedback loop to re-weight the original similarities. If one of the original matches is similar to the better results, we should increase the score of that match. If it turns out that it is similar to the worse results, we should lower the score. If all goes well, this will result in better top matches. We can redo the weighting with new feedback from these top matches in an iterative process until we are happy with the final results. The re-weighting

Input: A new design for a spark plug in a lawn mower that optimizes fuel efficiency.

Ranking	Patent description
1	A lawn mower with low fuel consumption.
2	A new fuel for lawn mowers.
3	An inventive design for rotor blades in a lawn mower.
4	An optimal method for path finding for autonomous lawn mowers.
5	A spark plug with iridium that improves performance over traditional spark plugs.

Table 2.1: An example of a search with many bad results and the best result at 5th place.

can be done in many ways. In this Master's thesis, we will suggest and evaluate models for this purpose.

2.2 Document representation

2.2.1 Frequency-based methods

There are several methods to calculate a similarity between two text documents. One could, for example, simply compare the number of words that occur in both of them as a measurement. However, there are problems with this. For one thing, this method would not account for the length of the documents, as large documents with many words generally will produce a higher score and small documents the opposite. Neither does it account for the number of times a certain word is used. Also, the most common words as "the", "be", "to", "of", etc. are not the words we want to use for similarity checks when comparing the general subjects of two documents.

Methods exist that aim to reduce or eliminate some of these problems. One such method used is the bag-of-words model (BOW) that for all words in a list keeps track of the word Term Frequency (TF), that is how many times a given word was present in the document. A document can then be represented as in Table 2.2 with a numerical vector.

Text: Daniel is eating pesto. Daniel likes pesto.						
word:	daniel	is	eating	pesto	likes	chess
frequency:	2	1	1	2	1	0

Table 2.2: An example of document representation by TF. Notice that the words are not case-sensitive. The vectorized document representation is [2, 1, 1, 2, 1, 0].

A similar and more refined method is term frequency-inverse document frequency (TF-IDF), which uses term frequency (TF) and inverse document frequency (IDF). It accounts for the length of a document and multiplicity of words by using TF. It also accounts for filtering out the most common words with IDF by using a set of documents as a reference.

$$\text{TF-IDF (term)} = \frac{\text{multiplicity (term)}}{\text{lenght of document}} \cdot \log \frac{\text{count of documents in set}}{\text{nr of documents with term}} \tag{2.1}$$

From Equation 2.1, we deduct that if documents where the word is used are common, this word is given less weight to the general content of a document than a word that is not common. A document can then, for example, be represented by a list of words and their corresponding TF-IDF value similarly to what is done for BOW in Table 2.2

A problem that is not so easily solved when using these methods is that a certain word can have a different meaning given a certain context. "Hit", can for example be used both as a verb, noun, and adjective. Also, a hit can be a synonym for a punch, but in a different context it can mean a popular song. There are methods for solving this by using the surrounding words to determine a word's part of speech as well as in what synonymy context it is used. However, even if they use very large corpora and good statistical models, they will still fail to see a similarity between words that are not synonyms. One can argue that for example, the words "driving" and "car" are not very similar in a grammatical sense, however to a human, it is obvious that they can be used in a similar context.

2.2.2 Word Embeddings

Word embeddings are a way of representing words as real-valued numerical vectors in a fixed vector space. They are dense and low dimensioned in the sense that they are created from something very high dimensional and sparse (e.g. the space of *one-hot encoded* words from a corpus, meaning each word will be represented with the value 1 in one dimension and 0 in all others) to a dimension of some hundred or thousand values.

Ideally, each dimension of the vector will represent a certain syntactic or semantic feature of a word (Turian et al., 2010). Two vector representations of different words can then be said to be similar if they contain similar values in several of the dimensions, i.e. lie close to each other in the vector space.

2.2.3 Sentence Embeddings

Sentence embeddings, much like word embeddings, are a way of representing texts with real-valued numerical vectors. However, as the name suggests, we now want a representation for the complete sentence rather than for only a single word. One could argue that the vector representation of a text with BOW or TF-IDF is a sentence embedding, which would not be wrong. However, it is a very simple one, as each word in the text is connected to a single number. There exist much more complex embeddings, for example, the ones that are created from certain neural networks.

2.2.4 Practical limitations of patent documents

What is the optimal length of texts with which you want to find their individual similarities? For some neural networks, as we will go into detail later, it is not possible to use too long texts as inputs. However, one solution to this would be to split the text into many parts and use an average embedding for the parts as document representation. Therefore, it is easy to

think that the documents ideally should be as long as possible, as there is more information to extract.

However, one must realize that that is a problem. Too much information and too many sentences will make the text loose its meaning in some sense. As sentences do not tend to be repeated in the same text, we will get a great number of sentences carrying the meaning of a great number of topics. This implicates that the "average" meaning of the text will be very general and not very specific, making it harder to differentiate between different texts.

For this reason, it is desirable to keep the texts short while at the same time having enough information to describe the topic. Here, we are working with patent documents that all contain an abstract which in short describes the patent. To get as much information as possible, we will also work with the title of the patent as we believe that it also provides a compact description. Thus, we concatenate the title and abstract for each patent to use as representation. In the future sections, when we talk about patent texts, keep in mind that we most likely refer to this title-abstract representation.

2.3 Neural Networks

2.3.1 General Neural Networks

A neural network (NN), sometimes also called artificial neural network (ANN) is essentially a network of nodes that through mathematical transformations takes an input and provides an output. The nodes are connected in the sense that each node is fed information from the input or another node in the network, applies a mathematical function to it, and forwards the outcome to other nodes or the output.

When we talk about the training of a network, we indirectly talk about adjusting the transformations that are happening in the nodes. One can train a NN in many ways, but in general, we feed training data to it. The NN uses its current transformations to get an output and then uses a pre-determined loss function to see how successful (or unsuccessful) it is. The transformations are then adjusted to (hopefully) provide a better output.

We often talk about the different layers of a NN. Typically there are three, that is an input layer, a hidden layer, and an output layer. The hidden layer is where the transformations take place, and it often consists of many sub-layers, sometimes constructed differently. We will not go into detail here about all the different networks and layer designs that exist. Instead we will focus on the ones that are used in this project.

2.3.2 Tokenizers

Before we dive deeper into the field of neural networks and embedding creation, we will explain what a tokenizer is. A neural network that transforms words or sentences into embeddings often finds in what context a specific word is used by looking at the other words around it. However, we can not just feed the sentence into the network immediately as we need to tell the network where to find the different words, or more specifically, tokens.

It may be easy on first thought, the words are simply separated by empty spaces. However:

1. First of all, it should be noted that this is not the case for all languages.

2. Secondly, some words are so-called n-grams, where several words represent one thing. For example, “fire alarm” may be better if represented as a single word.

3. Thirdly, neural networks have no direct sense of what grammar is. It is obvious to us that “run” and “running” are two forms of the same word. To get the network to understand that, it is sometimes beneficial to create sub-words, in this case as “run-” and “-ning”.

4. Lastly, we will say that there are many more advantages of using tokenizers. For example, in a sentence, some words may exist only for grammar’s sake but do not make the sentence more understandable. These should thus be removed and not used as tokens that are fed into the network.

What finally will be input into the network is a list of tokens.

2.3.3 BERT

In 2018, Google released a paper about a new language representation model called BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019). The BERT network is a deep network based on the Transformer-network introduced in 2017 (Vaswani et al., 2017). The Transformer, contrary to earlier recurrent neural networks (RNNs) in NLP does not need the sequential data to be ordered. Therefore, Transformer allows for much more efficient computations as they can be done for several words in parallel. In BERT-networks, this architecture is used in two training tasks.

Task 1 The network is trained using a masked language model pre-training objective. Tokenized sentences are used as input where 15% of the words are replaced with a masked token. The network is trained by trying to predict the correct word where the masked token is using information from the context, i.e. the words around it Devlin et al. (2019). Traditionally this is done by a specific sequence order, often reading the surrounding words from left to right or from right to left. The BERT process can be said to be bi-directional in the sense that it can be done from both directions. However, this is not completely true as the surrounding words are used as input simultaneously and not from a specific direction (Horev, 2018).

Task 2 The second part of training a BERT network is done using a next sentence prediction task. Traditional language modeling is usually done for separate sentences. However, sometimes we need to understand texts with several sentences. It is important to understand the relationship between subsequent sentences as context can carry over from one sentence to the next. The BERT network is thus trained with pairs of sentences from a corpus, where it is to decide whether the first one is followed by the second in a corpus. Half of the time that is the case. In the other half, the second sentence is chosen randomly from the corpus (Devlin et al., 2019).

The network is trained on the two tasks simultaneously, aiming to minimize the tasks’ combined loss functions.

BERT networks are useful for creating representations of words in texts that can consist of more than one sentence. Thus, the input is not limited in this regard, but rather by how

many tokens the tokenized input text consists of. Too long texts will speak of many different things, which means that it is not possible to capture a general meaning. In BERT networks, there is therefore a default max-length of the input text, limited to 512 input tokens. If there are more than 512 tokens, only the first 512 are used as text representation. Each text is also preceded by a special classification token CLS. It is an aggregated representation of all tokens in the text (Devlin et al., 2019). The CLS token can be useful for representing the whole text as an embedding.

2.3.4 BERT for Patents

The architecture of BERT networks allows for fine-tuning on specific tasks and datasets. We could for example transform input data so that it can be used by the BERT network and feed the output data into a new network. This is then fine-tuned for the specific task (Devlin et al., 2019).

For example, a large dataset of patent documents could be used to train a BERT network specifically for patents. This was done in 2019 by fine-tuning a pre-trained BERT network on a patent database of over 2 million patents (Lee and Hsiang, 2019). In 2020, Google published a white paper on a BERT network trained from scratch on a database with over 100 million patent documents (Srebrovic and Yonamine, 2020) and also released the network on Github (Srebrovic, 2020).

2.3.5 Sentence-BERT

Comparing two texts with a BERT network tends to be costly computationally-wise, as each text can consist of a lot of words and each word has to be assigned a numerical vector representation depending on its surroundings. It will be even more costly if you have a lot of texts, and want to find the most similar matches among them.

In 2019, Reimers and Gurevych (2019) introduced Sentence-BERT (SBERT) which dramatically limited computation time for such procedures while keeping the accuracy of BERT. Instead of a traditional BERT structure, it uses two *Siamese* pre-trained BERT networks with tied weights as in Figure 2.1. It is then fine-tuned on pairs of sentences where the objective can be, for example, to calculate the cosine similarity between the sentences.

Contrary to traditional BERT networks this means that it has naturally been trained on semantic similarities between texts which is what we do in this Master's thesis.

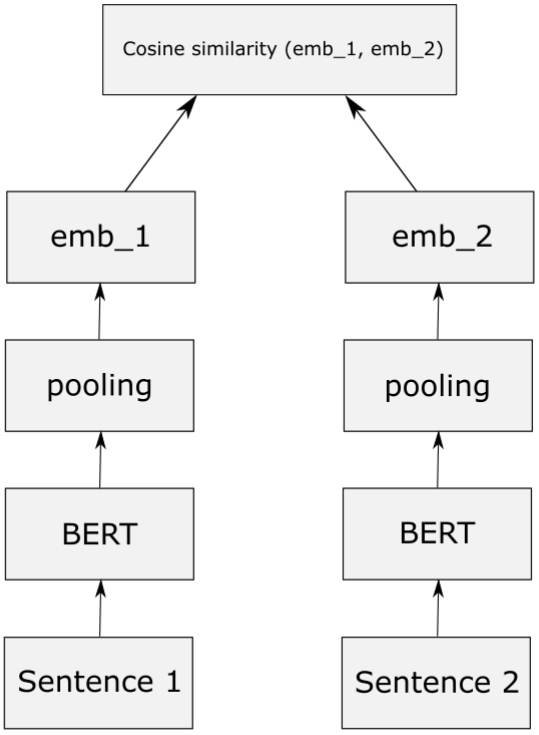


Figure 2.1: An example of the structure of an SBERT network. The two BERT networks are Siamese, i.e. they have shared weights. Here the SBERT network is used for sentence comparison with cosine similarity.

2.4 Similarity between numerical vectors

We have seen that texts can be represented by real-valued numerical vectors. However, we would still like to have a measurement of similarity between documents. The simplest form of calculating the similarity between two numerical vectors is to take the length of the line segment between them. This is called the Euclidean distance. The lesser the distance, the more similar the vectors are.

There are some drawbacks of using Euclidean distance for document comparison if we only care about the general meaning of the text. While different texts can be of different lengths, they may still have the same meaning. With Euclidean distance, this is not taken into account as the difference will be large.

However, there are more ways to measure similarity. Cosine similarity is a measurement of similarity between two non-zero vectors. Since we can express the dot product of two vectors as:

$$\mathbf{a} \cdot \mathbf{b} = ||\mathbf{a}||\,||\mathbf{b}|| \cos\theta, \tag{2.2}$$

where θ is the angle between the two vectors, we can deduct that

$$\cos\theta = \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}||\,||\mathbf{b}||}, \tag{2.3}$$

which is the cosine similarity.

Note that this does work for vectors of any dimension, even if the measurement the case for dimensions 2 and 3 are the easiest to understand intuitively as the angle between two vectors is harder to visualize in more dimensions. Also notice that as we divide with the length of vectors **a** and **b**, the length of the vectors does not matter for similarity measures. The single feature that decides the similarity is the angle between them. In Figure 2.2, it is clear that vector **a** is closer to **b** than to **c**, since the angle θ is smaller than α.

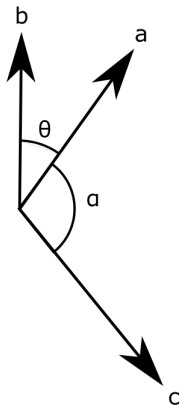


Figure 2.2: With cosine similarity, vector a is closer to b than c.

Reimers and Gurevych (2019) fine-tuned the Sentence-Bert network used in this work with minimization of the mean squared error of cosine similarity between sentence pairs. Thus, it is natural for us to use cosine similarity as the default measurement for similarity between embeddings. However, as all the embeddings produced by the Sentence-Bert are of the same size, Euclidean distance could also have been used. Previous experiments with this do not seem to make a significant change to the results (Reimers and Gurevych, 2019).

2.5 Re-ranking models

2.5.1 General model

Our focus in this project is to find methods that use an iterative, interactive re-ranking process to alter the original SBERT rating of matches for an input patent. This will be done by increasing the score of patents that are similar to ones the user has marked as “good” and lowering the score of patents that are similar to “bad” ones. This will be done in iterations.

In each iteration, the user will be able to choose the good and bad patents from top suggestions, originally produced by SBERT and in later iterations produced by our models. If all goes well, this will result in better top matches for each iteration. When the user is satisfied

with the results, no progress is being made, or a limit of time or a number of iterations has passed, the iterations stop.

The re-ranking can be done in many ways. In this Master's thesis, we will suggest and evaluate different models for this purpose. What our program does is described below in two simple steps:

Step 1: We start by performing an embedding search for similarity scores between an input text and patent texts in a database. We use an input text T_{in}, resulting in an ordered list of patent similarity scores S_{in}. For the first iteration $S_{new} = S_{in}$. Out of the top X results in T_{in}, the user will create two sets with what is believed to be good matches and bad matches. For each of these, we will create average embeddings E_{good} and E_{bad}.

Step 2:

1. First, we filter out the worst patents from S_{new} and S_{in} and only keep the top Z percent of patents.
2. We then perform similarity searches for E_{good} and E_{bad} on the database, resulting in lists of similarities S_{good} and S_{bad}.
3. Then, we re-weight the similarities in S_{in} according to the following formula for each patent with index i:

$$S^i_{new} = \text{memoryTerm} \cdot S^i_{new} + (1 - \text{memoryTerm}) \cdot \text{updateTerm} \tag{2.4}$$

where memoryTerm and updateTerm differ between different models.

We are now left with S_{new}, the re-weighted similarity scores, which we will order from best to worst. S_{in} will be reordered accordingly, e.g. an element that was previously in place i in S_{new} will have index k in both lists. From S_{new} the user will look at the top X matches. If the matches are all good, or if a number of max iterations j has been reached the iterations will stop here. Otherwise, we continue as follows:

1. The user creates two sets with good and bad matches from S_{new}, and as before creates the average embeddings E_{good} and E_{bad}.
2. We update iteration number and similarity scores: $j = j + 1$, $S_{old} = S_{new}$.
3. We redo **Step 2**.

2.5.2 Specific models

For this Master's thesis, we evaluated two models:

Model 1:

$$\text{updateTerm} = S^i_{in} \cdot \frac{S^i_{good}}{S^i_{bad}} \tag{2.5}$$

Model 2:

$$\text{updateTerm} = S^i_{in} + \text{weightTerm} \cdot (S^i_{good} - S^i_{bad}) \tag{2.6}$$

X
Z
weightTerm (only for model 2)
memoryTerm

Table 2.3: Parameters of the methods

Table 2.3 shows the parameters of the models that have to be decided before testing.

One should note that in model 1, we have a division in the update term. The original assumption was that this would not affect the worst matches among all patents as both S^i_{good} and S^i_{bad} were in the original top matches. Thus, they would be approximately equally "far away" from the bad patents and the division term would be 1. However, in early tests, this was not the case and very bad matches would sometimes surge to the top. Thus, while only keeping the top Z percent of results in every iteration limits the amount of data we have to search through, it is also necessary for good performance with model 1.

2.6 Evaluation concepts

Here we will introduce the concepts of a few methods of evaluation that we will later use for testing. Each method will have its benefits and drawbacks.

2.6.1 Averaging on a rating scale

In this Master's thesis, we will evaluate our results with several different methods. Our test persons will be working with patents that they will be able to rate on a scale from 1 to 5, with 5 being "a very good match" and 1 being "a very bad match" to an input patent. This simple rating is a good way of examining the quality of a patent match. To evaluate the quality of a set of patents, one can simply take the average of the patents in the set. The patents will be presented in a suggested order, with the best candidate first (according to the program). However, to evaluate if the ordering is good, we need something more than just the ratings of the patents.

2.6.2 Inversion number

When measuring how well a list is ordered, a common way is to use the inversion number of the list. The inversion number of a list $X = [x_1, x_2, ..., x_n]$ is defined as all possible permutations between the values in X for which the following is true: $i < j$ and $x_i > x_j$. This is for a list that ideally is ordered from lowest to highest values of X. When working with a list that ideally is ordered from highest to lowest values of X, the following condition should hold instead for the permutations: $i < j$ and $x_i < x_j$. The latter is what we will be working within this Master's thesis. In this case, the following list

$$X = [4, 3, 2, 5] \tag{2.7}$$

will have an inversion number of 3, since besides the last element the list is ordered, and 3 other elements ideally should be ordered after 5.

The downside of using inversion number for order evaluation is that in this case, as we will see later, we will be working with patent sets that partially will change. The ordering may worsen by adding a "better" new patent at the bottom of the set. However, it will be an improvement for the average rating of the set. Thus, the two evaluation methods might compete with each other.

2.6.3 Relevancy of patents and Average Precision

We would like to find a method of evaluation that can estimate the effectiveness of ordering without suffering from replacing a bad patent with a good one at any place in the set (or gain from replacing a good patent with a bad one).

In machine learning, there are many different ways to evaluate the findings. To map the outcome of a result, one can use positives and negatives to represent successes and failures. For example, if we want a network to predict if a patent belongs to a specific class of patents, we say that the result was a true positive (TF) if it was predicted to belong to the class and it does, while it is a true negative (TN) if it was predicted to belong to another class and it does.

This is of course what we aim for since the network in both cases predicted correctly. However, the network does not always do what we want, and the predictions can also be false. An example is if the patent is predicted to belong to the class and it does not. Then it is a false positive (FP).

Similarly, if the prediction was that the patent does not belong to the class, but it does, then we have a false negative (FN). These four outcomes can be used to create metrics for evaluation (Google Developers, 2020).

Accuracy is the simplest form of evaluation metric as one can think of in terms of how often the outcome is predicted correctly:

$$\frac{\text{Number of correctly predicted outcomes}}{\text{Total number of outcomes}},$$

or in other words: (2.8)

$$\frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}.$$

However, working with accuracy is not a good solution for our purposes. In this Master's thesis, we will be testing a program that presents the user with a few predictions of good candidate patents from a large database. In this regard, these predictions can be viewed as the positives TP + FP. The user will then rate these candidates, giving us the possibility to identify and separate which ones that are relevant (TP) and irrelevant (TN). However, for the rest of the patents that did not make the cut, the negatives TN + FN, we will not be able to identify and separate them. There is simply no time for the user to go through and rate all patents. Thus, accuracy becomes impossible to calculate, meaning that we must use another evaluation metric.

For our work, it is better to use what is called precision. Precision is defined as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.9}$$

In other words, what we are doing is measuring what percentage of positively classified results was positive. This is excellent, as we are only working with positives, and the user can separate TP from TN by rating their relevancy. As precision is calculated over a complete set, we could consider only the top k element in an ordered set. We then calculate something called Precision at k ($P(k)$). We can use this to create what is called Average Precision at k (AP@k):

$$\frac{\sum_{i=1}^{k} P(i) \cdot \text{rel}(i)}{k}, \tag{2.10}$$

where rel(k) equals 1 if the document at position k is relevant and 0 otherwise (Craswell and Robertson, 2009).

Finally, we have arrived at something that combines overall rating and ordering into a measurement. The rel(k) accounts for the former (only highly rated patents are relevant) while AP@k will be higher if the relevant patents are ordered first. This becomes clear if we view the two sets S_1 and S_2:

$$\begin{aligned} S_1 &= [\text{relevant}, \text{relevant}, \text{irrelevant}] \\ \text{AP@k}_1 &= \frac{\frac{1}{1} + \frac{2}{2} + 0}{2} = 1 \end{aligned} \tag{2.11}$$

$$\begin{aligned} S_2 &= [\text{irrelevant}, \text{relevant}, \text{relevant}] \\ \text{AP@k}_2 &= \frac{0 + \frac{1}{2} + \frac{2}{3}}{2} = \frac{7}{12} \end{aligned} \tag{2.12}$$

.

Thus, even though the two sets contain the same elements, the ordering matters.

Closely related to AP@k is Mean Average Precision (MAP). MAP is simply AP@k averaged over a number of different queries or tests.

The threshold for what is relevant or not might be biased as some users are more likely to, on average, find patents relevant. This can be accounted for, by choosing the threshold individually for each user according to their average rating.

One should note that "relevant" in this case does not necessarily mean that the user thinks the patent is relevant for some purpose. What patents one user thinks are relevant may be a much smaller subset of all patents than what another user thinks. Relevant in our measurements simply means that the user thinks more highly of the patent than they did of the average patent in this test.

Finally, we will give a warning when comparing MAP, especially between different test persons. First and foremost, some people will be more likely than others to find patents in general relevant. This will make the MAP score different for different people, even if they agree on which order the patents should be in.

We can compensate by using the average of a scaling rating as a personal threshold for relevancy. However, if we are using discrete values on this scale, as we are doing in this Master's thesis, the mean will be differently far away from an upper discrete value for some users than for others. For example, two persons may have almost the same mean and threshold, only one of them slightly over a discrete value and the other one just under. They might have almost the same rating preferences but will still get different MAP scores.

This could be solved by using float numbers instead of integers as possible ratings. However, that would make for some other problems, as ordering between the results would become more difficult to examine. If one result was a tiny bit better than another, their individual rating should not matter too much. More advanced evaluation systems would have to be constructed to prevent such ordering displacement not mattering too much. In the end, integer rating was the simplest solution, both for us and for the test persons.

2.6.4 MRSC and ARSC

Suppose we have two lists with the same elements in them but in a different order. To see how much one element has changed in rating order from one list to another is easy: If it was placed 5th in order in the first list and is placed 8th in the second list, it has moved 3 steps. This would also be true if it has moved in the reversed order from 5th to 2nd place.

In this project, we will be examining the maximum rating order change of any element in two lists, as well as the average rating order change for all elements. Hereafter, these two measurements will be referred to as Maximum Rating Step Change (MRSC) and Average Rating Step Change (ARSC).

Chapter 3
Experimental setup

In this chapter, we will describe the working process of our experiments. We will also try to give the reader some insight into our thinking process and why we wanted to do things the way we did.

3.1 Network testing

For similarity comparison, we wanted to use a network that scored high in similarity comparison tests and the choice fell on the SBERT network *bert-base-nli-stsb-mean-tokens* due to its high performance in STS benchmark tests as well as its high speed compared to traditional BERT networks (Reimers and Gurevych, 2019). However, there was an interest in using a network trained specifically on patent data, more specifically the BERT for Patents network by Google (Srebrovic, 2020).

To compare the two networks three random sets of 600 US patents were created from the Google Cloud patents-public-data (Srebrovic, 2021). Each set was chosen by letting each document be represented by its top 10 terms (unigrams or bigrams) and at random choosing 600 of these that contained a specific search term. The search terms for this test were “car”, “plastic” and “phone”. By comparing each patent in a set with all the other patents in all sets through cosine similarity on their embedding representation, one could then calculate the average similarity between each of the sets. The best performing network would be the one that easiest could differ between the sets.

3.2 Database creation

Calculating the embeddings for millions of document texts takes great computational power and time. It is highly impractical to do this every time a similarity search is performed. Thus,

this should be done beforehand, with the embeddings saved in a database where each patent's corresponding embedding can be referenced with the use of a unique index.

For this work, we are using a database from Mindified consisting of 66 352 156 patents, and a corresponding embedding database. The patents in the first database were filtered out from a larger set of patents from the European Patent Office World- wide Bibliographic database, with the constraint that they should be in English and each have a non-empty abstract and title. The embeddings database was then created from preprocessed and concatenated patent titles and abstracts.

The preprocessing was carried out by the elimination of noise in the texts, such as figure references and HTML tags that do not provide any information to the network. The concatenation then simply created a text for embedding creation by having the title as a separate sentence in front of the abstract. The embeddings were finally created using the SBERT network *base-bert-nli-stsb-mean-tokens* on these texts.

A database of 66 million patents will take a lot of time to search through, even with very efficient algorithms. With the resources available to us, this is simply too many patents, and with limited time for evaluations not possible. A solution here is to use clustering to organize and then to search through only a small, but relevant, part of the database, as was done by Navrozidis and Jansson (2020).

However, as we are interested in improving an assumed non-optimal output from the network, and keeping in mind that the network output is also used for organizing the clustered database in the first place, this may create some problems. Instead, we are working with a small subset of the database with patents that are classified in section E, class 06 (CPC), which are patents revolving doors, windows, shutters or roller blinds in general and ladders. This means that the final amount of patents to search through is thus 310,024.

3.3 Parameter settings

Tables 3.1 and 3.2 show the parameters we chose for the tests. Note that their values are set according to what seemed to produce reasonable results in early tests when working with the database we used for this Master's thesis. Also, some practical details had to be taken into consideration when choosing parameters, such as making the tests take a reasonably long time to run.

X = 5
Z = 10
memoryTerm = 0.85

Table 3.1: Parameters of model 1

3.4 Test input patents

For testing, we used randomly selected input patents from our database to perform similarity searches on. The only constraint for the input patents was that identical copies, or nearly

X = 5
Z = 10
weightTerm = 0.5
memoryTerm = 0.85

Table 3.2: Parameters of model 2

identical copies, could not exist in the database. This had been the case for a few patents in some early tests. In these cases, the outcome of the tests was scientifically irrelevant as we had "perfect" matches from the start.

The input patents themselves were not included as matches in the search results. The number of input patents was chosen to 5, as our resources were limited and our test persons had to finish their test within a reasonable amount of time.

The input patent texts that were used are listed in Appendix B.

3.5 Program for testing the methods

The program that is described in this section was written in Python.

The methods in Section 2.5 were implemented together with the database. For each configuration of user, input patent, and model, the performance of the model was tested over several iterations.

At the beginning of the program, the user was presented with an input patent which they had to read through, as seen in Figure 3.1. In each iteration, the user was given a set of top 5 candidate matches to the input patent, suggested by the specific method in use. 3.2 shows an example of this.

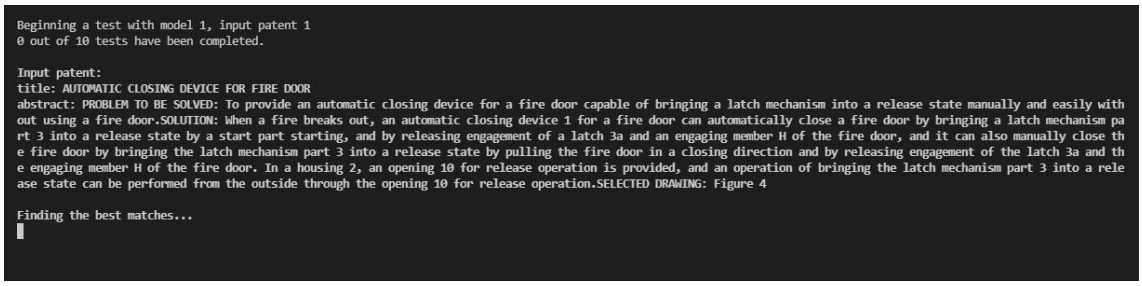


Figure 3.1: The interface the users worked with, here presenting the input patent for this test.

In each iteration, the user was given 2 tasks:

1. Give all of the candidates a rating on a scale from 1-5[1], with 5 being a "very good match" and 1 being "a very bad match".

2. Divide the candidates into a good set, and a bad set.

[1]In the first iteration, it was only possible to rate the patents 2, 3, or 4. This was to guarantee the possibility of seeing improvement/deterioration over the iterations should they happen.

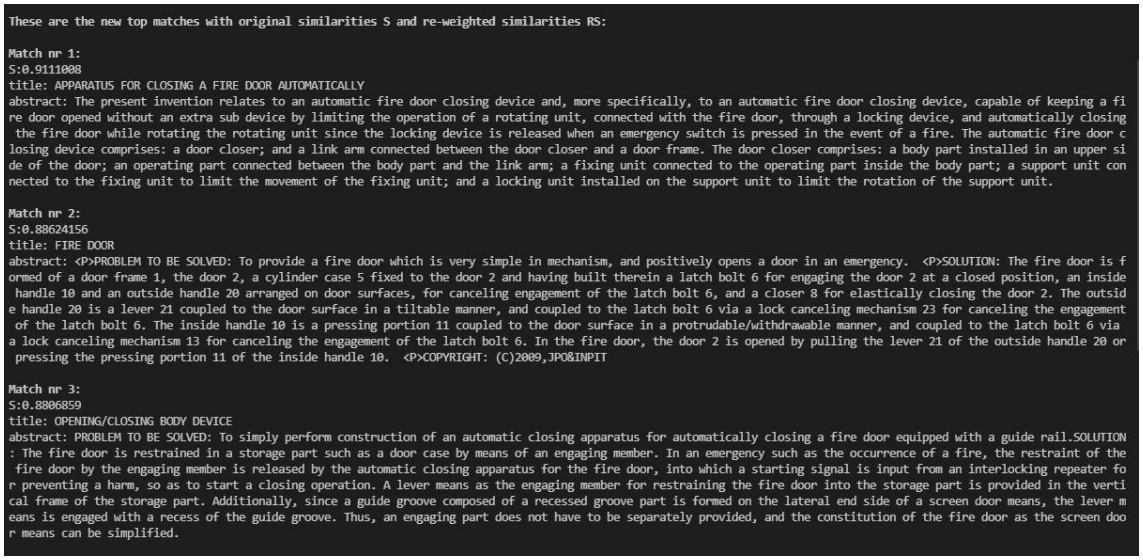


Figure 3.2: The suggested patent candidate matches to the input patent in figure 3.1 given by the program.

One should note that the rating of the patents given by the user is only used for evaluation of the tests in this report. The program itself does only change its candidate ordering according to what the user chooses as the good and bad set, as described in Section 2.5.

For the sake of consistency, the user was not able to give a different rating for the same patent if it returned among the top 5 candidates in different iterations. If the users were able to change the rating of a patent, it would be hard to interpret improvement or deterioration over the iterations.

What the user can change, however, is their opinion about what should be in the good set and the bad set. For example, a patent that is the best candidate in one iteration should belong to the good set. But if the method has presented new, much better candidates for the next iteration, the patent might be the worst in the group. Thus, it should now belong to the bad set. This makes it possible, given better and better candidates by the program, for the user to return better and better information to the program about how the ideal match should look like. If all goes well, we have a loop with improving candidates for each iteration until progress can be made no more.

The users were given some limitations to what patents the user can choose to be in the good and bad sets:

- All the patents had to be chosen in some group, and only in one. This is not strictly necessary for the model as one could technically work with “indifferent” patents that are neither good nor bad. It was implemented this way to make it simpler for the user.

- At least one patent must be assigned to each group. Otherwise, the methods in Section 2.5 would be impossible to use.

- The previously given ratings were used to make sure that no patent in the good set could be lower rated than any patent in the bad set (and vice versa).

3.6 Evaluation methods

As the procedure of evaluating the similarity between text documents is highly subjective, we decided that the best way to obtain an objective metric for evaluation would be to test the different methods using results from several different test persons and averaging them. We wanted to test the different methods in five areas:

1. How similar the top output candidates are to the input patent.
2. How the quality of the output candidates changes over the iterations.
3. How good the raw output candidates of the network are compared with the re-ranked ones.
4. How well the final output candidates are ordered.
5. How good the methods were at suggesting candidates that were not originally highly rated but still decent.

How well a method performs in suggesting patents that originally were rated quite low is perhaps what we desire most when searching through large databases (in combination with producing output patents of good quality). There are many methods out there that are designed to find similarities between patents. If our methods can find good patents that other methods do not rate highly, then there is indeed some value to it.

Equivalently, if our method does not promote low-rated patents, why should we bother and not instead just read through the default top suggestions manually? We will hereafter refer to this ability to promote low-rated patents as "effectiveness" and the patents themselves as "bubblers".

The different evaluation methods, together with their scope, a description, and corresponding area of purpose are listed below in Table 3.3.

Evaluation Method	Scope	Description	area of purpose
Rating (on a scale from 1-5)	Top 5[2]	Patent quality	1, 2 & 3
Inversion number (IN)	Top 5	Ordering	4
Mean Average Precision (MAP)	Top 5	Ordering	2[3] & 4
Max Rating Step Change (MRSC)	Top 50	Effectiveness	5
Average Rating Step Change (ARSC)	Top 50	Effectiveness	5

Table 3.3: List of evaluation methods

When working with MAP, the threshold for a patent being relevant (TP) or not (FP) was set to the average rating for all patents in all tests for each user separately.

The evaluation methods including Rating, IN and MAP were carried out on the top 5 candidate suggestions of the methods. However, when examining the effectiveness of the

[2]For the most parts, the results are presented as an average over the top 5 candidates. However, for iteration 0, we will present all ratings for comparison between the test persons.

[3]MAP is imprecise for comparison with rating results between different users because of several reasons.

methods through MRSC and ARSC, a wider scope of 50 was used. To see if large changes in similarity score order are happening, there is no need to have read through and rated the patents, thereby not limiting us to only study the top 5. Top 50 was chosen as it is a wider range that gives more data points.

It is important to also consider the rating as well when examining the effectiveness. Promoting low-rated patents is easy if we are not suggesting good candidates. High effectiveness is only useful if the ratings are improving.

3.7 Testing process

The evaluation tests were performed by employees from the intellectual property rights companies AWA in Malmö and Zacco in Copenhagen as well as an employee from Lund University. In total, a test was run for each of 5 different input patents. With two different methods, this resulted in 10 tests in total. All in all, 10 tests took approximately 2 hours to complete for each test person.

Chapter 4
Results

In this chapter, we will present the results of our work. We will first present the comparison between SBERT and the BERT for Patents network.

In the second part, we will study the results for the different interactive iterative patent search models when used together with SBERT. More specifically, we will go through results regarding user ratings, rating changes, and original network similarity score.

4.1 Network testing

In this experiment, we used three different sets, wherein each set, each patent had to contain one of the terms *car* (set 0), *phone* (set 1) or *plastic* (set 2) among its top terms. We then averaged the similarity scores from all possible combinations of two patents from two sets. This was done between all sets. Table 4.1 shows the results we obtained for our two networks.

	SBERT			BERT for Patents		
	0	1	2	0	1	3
0	**0.573**	0.437	0.456	**0.535**	0.491	0.471
1	–	**0.628**	0.403	–	**0.574**	0.466
2	–	–	**0.580**	–	–	**0.494**

Table 4.1: Average similarity scores between all patents in the 3 sets.

Both SBERT and the BERT for Patents network manage to find higher similarities between a set and itself than the set and other ones. However, the difference in similarity is much larger for SBERT, which suggests that this is a better network for semantic similarity comparison between patents.

4.2 Method evaluation

4.2.1 User ratings

During the tests, the test persons rated the output of the two models on a scale from 1 to 5, with 5 being the best and 1 the worst. This was done over several iterations, with iteration 0 being the default output of the SBERT network.

Average ratings for top 5 candidates

Model	1				2			
Iteration	0	1	2	3	0	1	2	3
Input patent 1	3.25	3.35	**3.40**	3.35	3.25	3.35	**3.40**	3.35
Input patent 2	2.80	2.85	**3.05**	3.00	2.80	2.95	**3.10**	3.00
Input patent 3	2.45	3.05	3.20	**3.35**	2.45	3.05	3.20	**3.35**
Input patent 4	2.95	2.90	3.10	**3.20**	2.95	2.95	2.95	**3.35**
Input patent 5	2.35	2.70	2.70	**2.95**	2.35	2.55	2.65	**2.9**
Average	2.76	2.97	3.09	**3.17**	2.76	2.97	3.06	**3.19**

Table 4.2: The ratings (averaged over top 5 candidate patents for all test persons) of the top 5 results in a search, for different models, iterations, and input patents.

As we can see in the ratings in Table 4.2, there is an increasing score from the default output of the SBERT network when compared to working with our models in the other iterations. It is not strictly increasing for all patents and iterations, but on average the trend is clear. As we can see in Figures 4.1 and 4.2, this trend is especially visible from iteration 0 to iteration 1, when the difference in ratings is especially large. The trend does not seem to stop after 3 iterations.

The ratings are very similar for model 1 and model 2. Model 2 has a slightly larger increase in ratings on average after 3 iterations, but it is very small.

Noticeable is that different patents have different rates of improvement. For example, patent 1 being the originally highest rated patent, has only gained 0.10 in average rating from iteration 0 to iteration 3. Meanwhile, patent 5, while being the originally worst-rated patent, has increased 0.55 on average.

When studying the scores for the individual test persons, as shown in Table 4.3, there is also a difference in the way the different test persons rated the patents. In particular, test person 2 had a much lower improvement rate of the ratings than the others. Test person 2 also gave lower scores than the other ones. Noticeable is that test persons 3 and 4 had the highest original ratings and also the highest final ratings in iteration 3. However, as test person 1 started with a lower original rating but still made an improvement similar to numbers 3 and 4, all of these 3 test persons can be said to have successfully improved their results when working with the models.

In Table 4.4, we see the ratings given by the test persons on iteration 0 for the top 5 candidates to each input patent. Notice that they all always have access to the same candidates in this iteration. Also, notice that the ratings in iteration 0 always will be the same for the models.

Average ratings for top 5 candidates

Model	1				2			
Iteration	0	1	2	3	0	1	2	3
Test person 1	2.68	2.88	2.92	**3.08**	2.68	2.92	2.96	**3.16**
Test person 2	2.56	**2.76**	2.72	**2.76**	2.56	2.68	**2.72**	**2.72**
Test person 3	2.92	3.12	3.32	**3.44**	2.92	3.16	3.24	**3.44**
Test person 4	2.88	3.12	**3.40**	**3.40**	2.88	3.12	3.32	**3.44**

Table 4.3: The ratings (averaged over the top 5 candidate patents for all input patents) for each model, iteration, and test person.

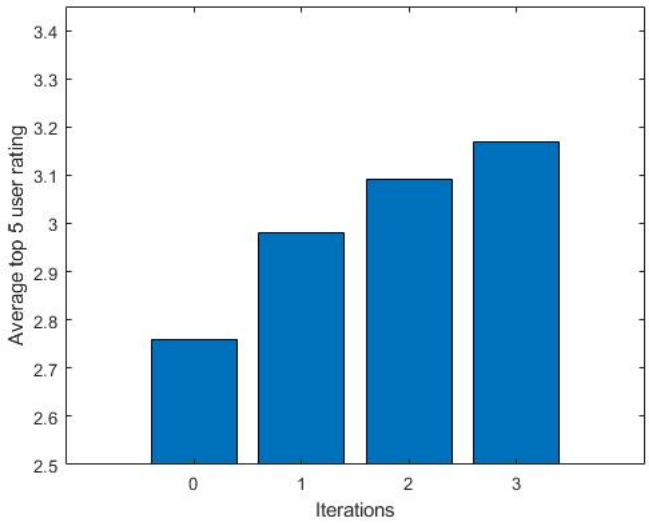


Figure 4.1: Average for all test persons and patents: the user rating for the top 5 results over iterations for model 1. A clear improvement can be seen.

The results clearly suggest that most often, the test persons agreed on the best and worst candidates.

4.2.2 Rating order changes

Here, we present the results regarding change in the order of the results. When talking about changes in order, we mean absolute change in order placement.

In Table 4.5, we can see changes in IN, MAP, MRSC, and ARSC averaged over all test persons. The overall averaged result can be seen for all of these, and we can see a consistent decrease for IN and an increase for MAP, MRSC, and ARSC for all iterations.

Studying the results for different patents, what we can see is that the same decrease/increase is present most of the time, but with some exceptions for IN and MAP.

The best-ordered patent in terms of IN and MAP in iteration 0 (patent 5) was still the best-ordered patent at iteration 3 considering MAP, but not IN. This was true for both models.

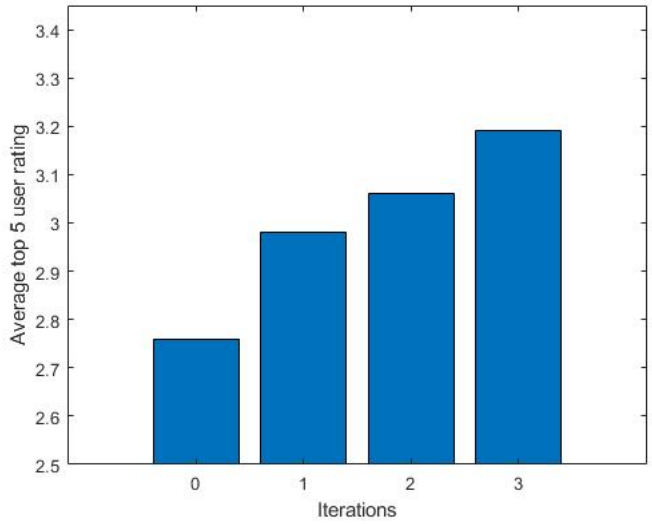


Figure 4.2: Average for all test persons and patents: the user rating for the top 5 results over iterations for model 2. The results are quite similar to those for model 1.

The least ordered patent in iteration 0 was not the same for IN (patent 4) and MAP (patent 1). Also for iteration 3, they were not the same (IN: patent: 1, MAP: patent 3). This was true for both models.

In Table 4.6, we can see IN, MAP, MRSC, and ARST averaged over all patents. Also here, there are a few exceptions from the overall trend. This time, apart from IN and MAP, one exception is also present in MRSC. The two models seem to perform similarly with a few exceptions (test person 1: MRSC and ARSC, test person 4: IN)

When studying the different test persons in terms of IN and MAP, we can that they differ a lot (particularly interesting is iteration 0, where they had access to the same patents). All of the test persons experienced an overall improvement over the iterations, with one exception being test person 4 with IN for model 1.

The test persons had very different scores in terms of MRSC and ARSC. Test person 1 and 4 had relatively similar scores here in terms of MRSC and ARSC for model 2, although they surprisingly differed quite a lot for model 1 (where test person 4 had similar results for models 1 and 2).

4.2.3 Original network similarity score

In Figure 4.3, we list the similarity score for the top 50 candidates for all input patents.

As expected, there are some differences between the candidates.

- To begin with, for input patent 1, we have a few patents that have a high similarity score compared with the other input patents. However, the score drops quickly and after the first 5 patents, we almost have a linear decay in similarity score for a while. Around index 30-50 the candidates almost have the same similarity score.

- For input patents 2 and 3, the few best candidates are not that much higher rated

Original ratings of candidates in iteration 0 for different test persons and input patents.

Input p.	TP 1					TP 2					TP 3					TP 4					#B	#W
1	3	4	4	2	2	4	3	4	4	2	4	4	3	3	3	2	4	2	4	4	3	3
2	3	2	2	3	2	3	2	3	3	2	3	3	4	2	4	4	4	3	2	2	3	3
3	3	2	2	2	3	3	2	2	2	2	3	2	2	2	3	4	4	2	2	2	4	4
4	3	2	4	3	3	3	2	3	3	2	4	2	4	3	4	3	2	4	3	2	4	4
5	4	2	2	2	3	2	2	2	2	2	3	2	2	2	2	4	2	3	2	2	4	4

Table 4.4: Candidate patent ratings in iteration 0 for the different test persons. #B and #W stand for the highest number of test persons that agreed on a candidate as the best/worst match. Commonly agreed upon best/worst matches are marked with an underline/overline for the test persons that agreed.

than the other ones. Thus, for the first 30 patents, we have an approximately linearly decreasing curve. After that, the candidates almost have the same similarity score. Compared with the ones in input patent 1, these candidates lie in a closer range.

- Input patent 4 is very linear in its appearance. Also, it has the closest range of all input patents.
- Input patent 5 is similar to input patent 2 and 3, perhaps a little more closely approximated to a linear curve.

Ordering results for top 5 ratings

Patent 1								
Model	1				2			
Iteration	0	1	2	3	0	1	2	3
IN	2.25	2.5	**1.25**	1.5	2.25	2.5	**1.25**	**1.25**
MAP	0.53	0.58	**0.67**	0.62	0.53	0.58	**0.67**	**0.67**
MRSC	0	15.25	26.00	**50.50**	0	16.50	34.50	**48.50**
ARSC	0	3.74	6.38	**9.40**	0	4.07	6.49	**8.89**
Patent 2								
Model	1				2			
Iteration	0	1	2	3	0	1	2	3
IN	2.25	1.00	1.00	**0.00**	2.25	0.75	0.75	**0.00**
MAP	0.33	0.39	0.56	**0.65**	0.33	0.51	**0.66**	0.65
MRSC	0	65.25	106.00	**180.00**	0	54.75	98.25	**148.75**
ARSC	0	10.65	21.06	**31.49**	0	9.81	22.01	**30.2**
Patent 3								
Model	1				2			
Iteration	0	1	2	3	0	1	2	3
IN	1.50	3.00	1.50	**0.75**	1.50	1.00	1.25	**0.00**
MAP	0.22	0.44	0.60	**0.73**	0.22	0.46	0.60	**0.74**
MRSC	0	58.25	235.25	**253.00**	0	84.25	106.50	**114.25**
ARSC	0	11.75	19.62	**22.59**	0	10.01	16.47	**20.62**
Patent 4								
Model	1				2			
Iteration	0	1	2	3	0	1	2	3
IN	3.25	1.00	1.25	**0.00**	3.25	1.00	**0.25**	0.75
MAP	0.41	0.51	0.61	**0.70**	0.41	0.51	0.54	**0.72**
MRSC	0	84.25	106.50	**114.25**	0	102.50	111.75	**150.25**
ARSC	0	10.01	16.47	**20.62**	0	13.45	20.34	**27.74**
Patent 5								
Model	1				2			
Iteration	0	1	2	3	0	1	2	3
IN	1.00	2.25	1.00	**0.75**	1.00	1.00	**0.50**	1.00
MAP	0.12	0.22	0.21	**0.30**	0.12	0.16	0.21	**0.33**
MRSC	0	39.00	55.25	**116.00**	0	43.00	57.25	**114.25**
ARSC	0	7.95	12.72	**19.09**	0	8.50	13.33	**19.32**
Average								
Model	1				2			
Iteration	0	1	2	3	0	1	2	3
IN	2.05	2.1	1.2	**0.65**	2.05	1.65	0.85	**0.75**
MAP	0.32	0.43	0.53	**0.60**	0.32	0.44	0.54	**0.62**
MRSC	0	49.50	105.50	**143.90**	0	55.00	107.40	**142.95**
ARSC	0	8.73	15.14	**20.96**	0	9.52	16.35	**21.87**

Table 4.5: IN, MAP, MRSC and ARST, averaged over all test persons.

Ordering results for top 5 ratings

Test person 1								
Model	1				2			
Iteration	0	1	2	3	0	1	2	3
IN	2.80	2.00	**1.40**	**1.40**	2.80	1.80	1.60	**1.20**
MAP	0.42	0.57	0.74	**0.76**	0.42	0.58	0.73	**0.8**
MRSC	0.00	43.40	**83.80**	73.40	0.00	57.40	95.80	**100.40**
ARSC	0.00	8.12	12.96	**13.92**	0.00	10.67	16.67	**19.50**
Test person 2								
Model	1				2			
Iteration	0	1	2	3	0	1	2	3
IN	1.20	2.40	0.60	**0.40**	1.20	1.40	0.60	**0.40**
MAP	0.39	0.51	0.53	**0.60**	0.39	0.51	0.57	**0.60**
MRSC	0.00	54.60	177.80	**213.00**	0.00	61.20	170.80	**209.60**
ARSC	0.00	10.21	17.44	**20.71**	0.00	9.48	16.46	**21.80**
Test person 3								
Model	1				2			
Iteration	0	1	2	3	0	1	2	3
IN	2.40	2.00	1.00	**0.40**	2.40	1.60	**0.40**	**0.40**
MAP	0.20	0.25	0.35	**0.50**	0.2	0.30	0.41	**0.50**
MRSC	0.00	52.00	92.20	**184.40**	0.00	52.60	93.20	**154.00**
ARSC	0.00	8.88	17.14	**30.19**	0.00	9.92	18.98	**28.01**
Test person 4								
Model	1				2			
Iteration	0	1	2	3	0	1	2	3
IN	1.80	2.00	1.80	**0.40**	1.80	1.80	**0.80**	1.00
MAP	0.28	0.38	0.49	**0.54**	0.28	0.38	0.44	**0.59**
MRSC	0.00	48.00	68.20	**104.80**	0.00	48.80	69.80	**107.80**
ARSC	0.00	7.72	13.01	**19.02**	0.00	7.99	13.31	**18.17**

Table 4.6: Inversion number (IN), Mean Average Precision (MAP), Max rating step change (MRSC) and Average rating step change (ARSC), averaged over all patents.

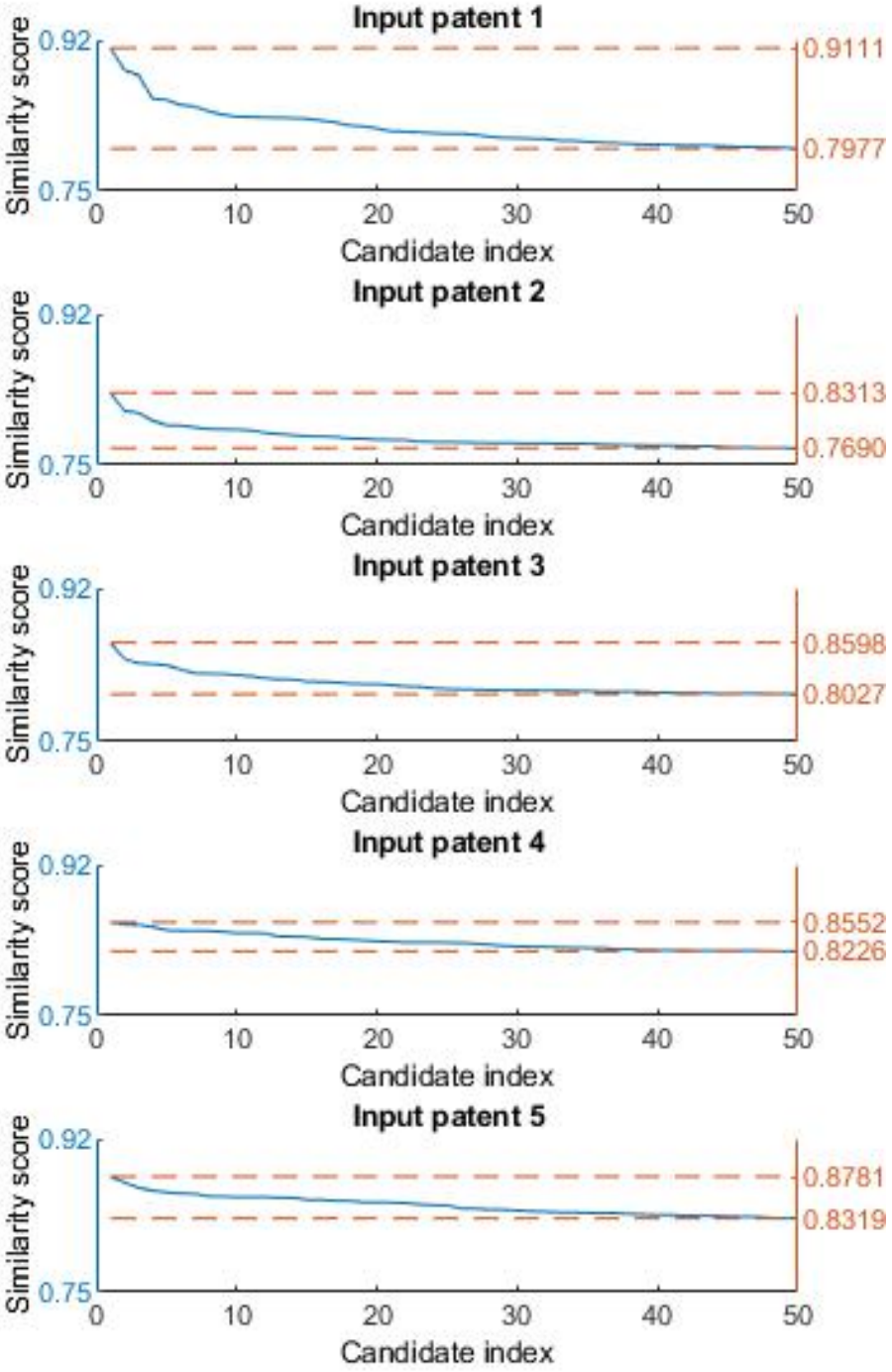


Figure 4.3: The similarity score for the original top 50 candidates for all input patents.

Chapter 5

Discussion

In this chapter, we will first shortly discuss our findings from the comparison between SBERT and the BERT for Patents network. In the second part, we will discuss the results for the different interactive iterative patent search models when used together with SBERT. Lastly, in the third part, we will try to pinpoint possible improvements in our work and suggest areas of interest for future experiments.

5.1 Network testing

This part of the project was a much smaller part than the rest of it. We did it out of curiosity to see how a network that was trained on a large database of patents would perform when compared to a more conventional network such as SBERT. The patent embeddings used in this is a BERT network that is optimized for word prediction and detection of pairs of subsequent sentences, while SBERT specifically has been trained to find the cosine similarity between pairs of sentences. Thus, it was no surprise that the individual differences between each of the classes were much larger for SBERT than for BERT for Patents. However, it strengthened our beliefs that SBERT was a good choice as a network for semantic comparison with cosine similarity.

It would have been interesting to see how an SBERT network trained or fine-tuned specifically on patent texts would perform in comparison with the conventional one. However, it is a much harder task to put together a large training database with only patent texts than general texts. Thus, the results of such a test would greatly depend on the training database.

5.2 Evaluation

5.2.1 Overall results

From our results, it is clear that the quality of the top patents does indeed improve on average when working with an interactive iterative feedback loop as we have done in this Master's thesis. The final result of iteration 3 has a top 5 average similarity of 3.17 and 3.19 for models 1 and 2 when compared to the original input patents, as seen in Table 4.2.

Since the scale from 1-5 is very subjective in the way of how one would describe two patents' similarity in words (one person could say that a 4 is very similar, while another would have 4 as a threshold for being relatively similar), it is not so easy to say what the ratings mean in reality. However, what can be said is that they are improving over the iterations.

Studying Figures 4.1 and 4.2 makes it clear that the two methods we have tried are not only both valid, they are also very similar in the quality of the results they produce.

In Table 4.6, we see that the average IN and MAP does go down, which is an indication that the top 5 results are better ordered than from the original SBERT output. It is clear that the change in MRSC also for the most part matches the change in ARSC.

Regarding MRSC and ARSC it is clear from Table 4.6 that they increase on average over the iterations. This is very good for the usefulness of our models, at least when keeping in mind that the ratings also does improve.

In the next following sections, we will draw more in-depth conclusions from the results.

5.2.2 Differences in results for test persons

As seen in Table 4.3, test person 2 had the worst results in rating with only a small increase. However order-wise, as seen in Table 4.6, IP and MAP were still decent. Even if this test person gave noticeably smaller ratings for the same starting candidates in iteration 0 as the other test persons, this does not affect the program in the next iteration. What it does affect, however, is the way the person chose the good and bad set. From studying the test logs, it is clear that this test person more often than the others chose 1 candidate as either the good or bad set, and put 4 in the other. For test person 1 this was done 2 times. For test person 2 this was done 15 times. For test person 3 this was done 7 times. For test person 4 this was done 5 times.

Since we are working with averaging the embedding of the good and bad sets, the more candidates there is in one group will make the "meaning" of that embedding more blurred. In turn, the re-ranking will probably become more random. Rankings will be updated differently if their corresponding patents are more similar to the good or the bad set. With blurred embeddings, the result of the comparison is more arbitrary. Thus, this will lead to more randomly updated similarity scores, with worse results following.

What is interesting is that while this user did not have so much improvement, an improvement was still clearly present from iteration 0 to 1. It is hard to say what this is caused by as the before mentioned unbalanced set splitting occurred also here. As we have it, there is a limited amount of data to read from. Perhaps, if more extensive tests were done, it would become more clear.

Test person 2 also had a relatively large average MRSC and ARSC. Possibly, this is also

caused by more random updates in similarity scores. It is easy to get larger changes at the top if we make arbitrary changes across all candidates. However, it is also possible that this test person simply made "good choices" for finding "bubblers" far down in the candidate list. It is hard to say which it is.

The changes in IN and MAP were still decent for test person 2. This probably goes hand in hand with what we mentioned in Section 2.6.2 that IN should naturally increase if the new candidates at the bottom of the top suggestions are bad. Also, bad new entries replacing bad old ones at the bottom of the list do not affect MAP negatively. With the unbalanced set splitting of 4:1 or 1:4, the one patent should at least move up or down in ratings, making for better ordering.

Rating-wise the other 3 test persons all had similar improvements. As we have stated earlier, we should be very careful with comparing ordering. What we will compare is MRSC and ARSC. Far highest improvement here was achieved by test person 3. This, together with having one of the best improvements in rating makes test person 3 the one that had the most success overall. Appendix C contains some examples of the "bubblers" of test person 3.

5.2.3 Effect of choosing parameters

If we recall, the set of parameters for our models was:

X
Z
weightTerm (only for model 2)
memoryTerm

Table 5.1: Parameters of the methods

.

While the influence of the weight-term and memory-term can be discussed (how large adjustments we should make to our default similarity scores and how much we should keep from the last iteration), these are mostly parameters that were set through trial and error. With more testing, one would surely come up with better values for these. However, we should add that the best parameters probably are chosen by looking at the distribution of original candidates for the specific input patent, as we did in Sect. 4.2.3.

Since it seems like we sometimes have a larger spacing in similarity scores between candidates, we should perhaps change the parameters so that the updates of the scores are larger (or not, sometimes it can be good to not have the scores change too easily). Another way of doing this would be to apply some sort of transform to the distribution, although this would take a lot of computing power for the vast amount of data.

More easily discussed are the X and Z values, i.e. how many top candidates are shown to the viewer for rating and set splitting and how many percent of the top candidates we should keep in the next iteration.

X should not be too small, since the user is more likely to miss out on some of the best candidates that might be just below the top candidates. Also, if it is too small we will not be able to gradually tweak our results in a specific direction as easily since the "correct" direction

might be "between" some candidates. For example, if we want a search profile that matches some distinct parts of several different candidates.

The downside of X being too large, however, apart from it taking much longer to do the test because of the increased text body, is that the average embedding of the good and bad sets will be blurred. We have discussed this before, and blurred embeddings lead to more random updates in similarity scores.

Our value of $X = 5$ seemed like a good choice. One thing that we have not researched is if it is better to have an equal splitting, i.e. X being even. This could make for better results, although our best performing test person, number 3, had a few unbalanced splittings of 1:4 or 4:1.

The method of creating averaged embeddings was something we tried very early in this project and have not been reflected on after choosing parameters. However, one should be able to implement our methods without this. One could, instead of computing the similarity between the input patent and the averaged embeddings, compute the similarity between the input patent and all of the candidates, then take the average of the similarity. It is very possible that this also would produce good results. For one thing, it makes for the possibility of using larger values of X, although averaging many different similarities could still theoretically blur out the meaning.

Having a Z-value effectively makes the code run a lot faster after a few iterations, eliminating the worst results each time. We have not really tried to optimize this since what we mainly wanted to do was to eliminate the worst candidate matches. This is because, as discussed in Section 3.6, they caused problems for method 1. However, for method 2 Z should not be necessary, although also not harmful.

5.2.4 Differences in results for input patents & distribution in original similarities

For the most part, the ratings improved for all patents over the iterations. The biggest abnormality was for patents 1 and 2 where iteration 2 had slightly better ratings than iteration 3 for model 1. No improvement between these iterations was seen for model 2. The same pattern was present when looking at the ordering results for MAP where small deterioration with model 2. However, the IN improved quite a lot here.

By studying the results in Appendix A, it is likely that relatively few new patents were replaced in the top candidates for most users in the first iterations for patents 1 and 2. However, their individual ordering still changed for the better. Some replacements came in in iteration 3, and they were on average bad. This of course affects MAP negatively but may very well improve the IN score, as discussed in Sect. 2.6.2. This is clearly visible for model 2. Why model 1 still improved its MAP score regardless of the rating dropping probably had to do with a better ordering while the lowering of rating was not enough to go below the threshold.

The reason for not finding any good replacement candidates, and eventually finding some bad ones, can be understood if looking at the original distributions in Section 4.2.3. What we see for patent 1 is a much steeper curve, with large rating differences between the candidates, especially for the first 5. Of course, this will make it harder for introducing new patents, but as we discussed in Section 5.2.3, this may also be good. The results, in this case, showed that

by replacing the worst of the top candidates, the ratings got worse. Sometimes, the original candidates might simply be the best!

5.2.5 The subjective measurement of similarity

We ask ourselves the same question that we did in Section 3.6:

> How similar are the top output candidates to the input patent?

The answer to that question may be simple. For the most part, the best patents are rated 4 on a scale from 1 to 5 where 5 is the best. However, what does that mean in reality? Is 4 good enough for it to be used in a real-life situation when looking for prior art? The answer to that is probably that it depends. As many test persons commented about in Appendix D, it would have been easier to understand the finer details of the patent when using images as a complement to the patent text. Then one would be able to see if these finer details as for example "the latch mechanism" in input patent 1, Appendix B.1, were similar and the patents thus being relevant to each other. This is something that we did not have the resources to achieve, but it should be perfectly possible to do in an application.

When giving the instructions for the tests, we stated that "5 was a very good match" and "1 was a very bad match". We had two instances of 5s come up in the tests, and for these two matches we can say that they are very similar to the input patents. A few 1s were also found that we can say were very bad. Would we have done these test again, the instructions would have stated in words, how similar a patent would be for each rating.

What we also would have made different is to be able to give a ranking of 5 or 1 on iteration 0. We did not do this to guarantee a possibility of seeing deterioration/improvement but at the same time it makes it hard to say if for example some of the patents rated 4 was "a very good match". It makes sense that the default network simply finds the best patents on some occasions, but now we cannot see that.

As the similarity between text documents is so subjective, we cannot say exactly how similar the output candidates are to each input patent. But what we can say is that for each iteration we have improvements. To answer the question, we will have to say that our methods were able to bring forth many 4s, i.e. many new patents that were close to being "very good matches".

5.2.6 IN vs. MAP

We have concluded that the results for IN and MAP in Table 4.6 go down on average, indicating better orders. This is a good result, especially for the IN considering that the ratings for the candidates are improving on average at the same time.

As discussed in Section 2.6.2, there is a risk of getting worse scores for IN when new good patents emerge at the bottom of the top candidates. Although we see a worsening score on some occasions, we still see an average improvement in order. This is confirmed by the MAP which is a more reliable tool for evaluation in some sense. This is because, as discussed in Section 2.6.3 it will never worsen its rating when better patents emerge.

However, in some sense MAP is also a worse tool for evaluation as it, in this case, thresholds a rating on a scale into a binary rating of relevant or not. With such a subjective task as

comparing similarities between longer texts, it is not so easy to place a text in one of just two categories, and a rating scale from 1-5 should give us more information.

The two methods thus complement each other, and when discussing the order results, both of them should be included.

5.3 Possible improvements and future work

The largest improvements that can be made to our models are quite possibly adjustments to the parameters, which we discussed in Section 5.2.3. Extensive testing and fine-tuning here should make the models better. That being said, these models are very simple in purely mathematical terms. There exists an infinite pool of variants for these models, each one more complicated than the other. Realistically, the optimal model should be much more complicated than what we have done. However, what we set out to do was not to achieve perfection, but simply to see if it was possible to improve the results in an interactive iterative process, and it was!

There are a few things that should have been changed in this project but were not. For example, the ratings on a scale should not have been further limited on iteration 0. What also would be an improvement is the way the test persons did the splitting of good and bad sets. If presented with the same candidates in the same iteration, but for two different models, the splitting should have been identical. This was very uncommon in our tests and did likely not affect our results, but for consistency's sake and test comparison, it would have been nice.

One thing that should be noted about this work is the limitations in resources for testing. What should be done in the future are tests with more test persons, input patents, and iterations. The last one is especially interesting since we, on average, saw improvements on all of the iterations. At some point, it is reasonable to believe that improvement should stop or deterioration of the ratings should take place. At what point in time that one could perhaps change the parameters, of course depending on if perfect results already have not been achieved.

One thing that could be examined in the future is for example the splitting of the candidates into good and bad sets. Further testing should say if it would have been better to have a predetermined size of the splittings into more even sets.

As said before, extensive testing and fine-tuning of the parameters would be good to do in the future. For a functioning application in the patent industry, one would have to include images and possible links to the full-text patent for each title-abstract pair.

It would be interesting to explore the use of other neural networks as we have done with SBERT. At the time of writing this, SMART-RoBERTa Large is a network with very high benchmark scores on semantic similarity tests (paperswithcode.com, 2021).

Perhaps one of the most promising aspects of interactive iterative re-ranking is the potential for the methods to find bubblers. As seen in Appendix C, they seem promising, even when quite far down in the list and never seen by the test persons. As that is, future works could test methods where the user can read and choose among the patents with the best improvement ranking-wise, not only the highest ranking.

One thing that is very important for the practical use of the models is to get them to run on a larger database. This will pose problems in terms of execution time. This could be solved with clustering and only picking the candidates from the top clusters. However, in

some sense, this is counter-productive as the clustering would be made with some similarity score from the beginning. Then the candidates would only be pulled from what already is considered to be the most similar ones, thus preventing "bubblers" from far down in the scores to later rise. That being said, clustering might still be the way to go given enough clusters and a large number of candidates from each of them. Another option would be to create many smaller databases and, as we have done, search the appropriate database or databases for matches to the input patent. However, the drawback of this would be that similarities between patents in different databases would impossible.

Chapter 6
Conclusions

When we began this project, we set out to explore ways of using neural networks for semantic similarity searches in patent databases in an iterative, interactive process. We had two simple mathematical models for re-ranking the results that we wanted to test in five different areas:

How similar the top output candidates are to the input patents? Although there is some problem translating our rating scale into words, we can say for that the most part, the best patents were close to, or very similar to the input patent. For some worse candidates, we can say that they were not very similar to the input patents.

How the quality of the output candidates change over the iterations? While a few iterations showed signs of no change or small deterioration, on average a clear improvement could be seen. Overall, the quality of the candidates changed for the better.

How good the raw output candidates of the network are compared to the re-ranked ones? As we saw a steady improvement in each of the iterations, one can only state that the final output is of higher quality than what we started with.

How well the output candidates are ordered? This was the most difficult part of the tests to evaluate since new results were introduced to the top candidates for most of the iterations. In the end, the IN and MAP scores together point to the ordering on average becoming better over the iterations. Depending on which one is used, some minor deterioration could be seen in specific tests.

How good the methods were at suggesting candidates that were not originally highly rated but still decent? This might be the most promising result of our work. Our methods frequently produce "bubblers" that are not originally ranked very highly, but in the end, rise to the top candidates. If the user experiences better patents at the same rate that we introduce bubblers to them, it means that we can find patents that otherwise would have remained hidden!

After all of this, we can state that we have explored new territory in the field of textual semantic similarity search. Hopefully, our testing and our conclusions will make it easier for future researchers to employ similar methods to ours. That is for finding prior art in patent databases and for more general semantic similarity searches alike.

References

Craswell, N. and Robertson, S. (2009). Average Precision at n. In LIU, L. and ÖZSU, M. T., editors, *Encyclopedia of Database Systems*, pages 193–194, Boston, MA. Springer US.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Google Developers (2020). Classification: True vs. False and Positive vs. Negative. Accessed: 11.05.2021
URL: https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative.

Horev, R. (2018). BERT Explained: State of the art language model for NLP. Accessed: 17.05.2021
URL: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270.

Khattab, O. and Zaharia, M. (2020). Colbert: Efficient and effective passage search via contextualized late interaction over BERT. *CoRR*, abs/2004.12832.

Lee, J.-S. and Hsiang, J. (2019). PatentBERT: Patent classification with fine-tuning a pre-trained BERT model. *CoRR*, abs/1906.02124.

Lee, J.-S. and Hsiang, J. (2020). Patent classification by fine-tuning BERT language model. *World Patent Information*, 61:101965.

Navrozidis, J. and Jansson, H. (2020). Using natural language processing to identify similar patent documents. Master's thesis, Lund University, LTH.

paperswithcode.com (2021). Semantic Textual Similarity on STS Benchmark. Accessed: 04.10.2021
URL: https://paperswithcode.com/sota/semantic-textual-similarity-on-sts-benchmark.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *CoRR*, abs/1908.10084.

Srebrovic, R. (2020). BERT for patents. GitHub Repository. Accessed: 08.06.2021 https://github.com/google/patents-public-data/blob/master/models/BERTforPatents.md.

Srebrovic, R. (2021). patents-public-data. GitHub Repository. Accessed: 04.10.2021 https://github.com/google/patents-public-data.

Srebrovic, R. and Yonamine, J. (2020). Leveraging the BERT algorithm for patents with TensorFlow and BigQuery. Accessed: 08.06.2021 https://services.google.com/fh/files/blogs/bert_for_patents_white_paper.pdf.

Turian, J., Ratinov, L.-A., and Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden. Association for Computational Linguistics.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Appendices

Appendix A
Test results

Table A.1 shows all the ratings given by all test persons for all input patents, methods and iterations.

Table A.1: All ratings.

Test person 1, model 1

Input p.	Iteration 0					Iteration 1					Iteration 2					Iteration 3				
1	3	4	4	2	2	3	4	4	4	2	4	3	4	4	3	4	3	4	4	2
2	3	2	2	3	2	3	3	2	2	2	3	3	2	3	1	3	3	3	2	1
3	3	2	2	2	3	3	3	2	4	4	3	4	3	4	2	4	3	3	4	2
4	3	2	4	3	3	4	3	2	3	2	4	3	3	3	1	4	3	3	3	2
5	4	2	2	2	3	4	3	2	2	2	4	3	2	3	1	4	3	3	5	2

Test person 1, model 2

Input p.	Iteration 0					Iteration 1					Iteration 2					Iteration 3				
1	3	4	4	2	2	3	4	4	4	2	4	3	4	4	3	4	4	3	4	3
2	3	2	2	3	2	3	3	2	2	2	3	3	2	3	1	3	3	3	2	1
3	3	2	2	2	3	3	3	4	2	4	3	4	3	4	2	4	3	3	4	2
4	3	2	4	3	3	4	4	2	3	2	4	4	3	1	3	4	4	3	3	2
5	4	2	2	2	3	4	3	2	2	2	4	3	2	3	1	4	3	3	5	2

Test person 2, model 1

Input p.	Iteration 0					Iteration 1					Iteration 2					Iteration 3				
1	4	3	4	4	2	4	4	3	4	2	4	4	3	4	2	4	3	4	4	2
2	3	2	3	3	2	3	3	2	3	2	3	3	3	2	2	3	3	3	2	2
3	3	2	2	2	2	3	2	2	3	2	3	3	2	3	2	3	3	3	2	2
4	3	2	3	3	2	3	3	3	2	2	3	3	3	2	2	3	3	3	3	2
5	2	2	2	2	2	2	2	3	4	3	2	4	2	2	2	4	2	2	2	2

Test person 2, model 2

Input p.	Iteration 0	Iteration 1	Iteration 2	Iteration 3

1	4	3	4	4	2	4	3	4	4	2	4	4	3	4	2	4	3	4	4	2
2	3	2	3	3	2	3	3	3	3	2	3	3	3	3	2	3	3	3	2	2
3	3	2	2	2	2	3	2	3	2	2	3	3	2	3	2	3	3	3	2	2
4	3	2	3	3	2	3	3	3	2	2	3	3	3	2	2	3	3	3	3	2
5	2	2	2	2	2	2	2	2	2	3	2	3	2	2	2	3	2	2	2	2

Test person 3, model 1

Input p.	Iteration 0					Iteration 1					Iteration 2					Iteration 3				
1	4	4	3	3	3	4	4	3	3	3	4	4	3	3	3	4	4	3	3	3
2	3	3	4	2	4	3	4	3	4	3	4	4	3	3	4	4	4	4	4	3
3	3	2	2	2	3	3	3	2	3	4	3	4	4	3	3	4	4	3	4	4
4	4	2	4	3	4	4	4	4	2	3	4	4	4	2	3	4	4	4	3	2
5	3	2	2	2	2	3	2	3	2	2	3	3	3	3	2	3	3	3	3	2

Test person 3, model 2

Input p.	Iteration 0					Iteration 1					Iteration 2					Iteration 3				
1	4	4	3	3	3	4	4	3	3	3	4	4	3	3	3	4	4	3	3	2
2	3	3	4	2	4	4	3	4	3	4	4	4	4	4	2	4	4	4	4	3
3	3	2	2	2	3	3	3	3	2	4	3	4	4	3	3	4	4	3	4	3
4	4	2	4	3	4	4	4	4	2	3	4	4	4	2	1	4	4	4	3	4
5	3	2	2	2	2	3	3	2	2	2	3	3	3	3	2	3	3	3	3	2

Test person 4, model 1

Input p.	Iteration 0					Iteration 1					Iteration 2					Iteration 3				
1	2	4	2	4	4	2	4	2	4	4	4	2	4	4	2	4	2	4	4	2
2	4	4	3	2	2	4	4	3	2	2	4	4	3	4	3	4	4	4	2	2
3	4	4	2	2	2	4	4	4	4	2	4	4	4	4	2	4	4	4	4	3
4	3	2	4	3	2	4	3	2	2	3	4	3	3	4	4	4	4	4	3	3
5	4	2	3	2	2	4	3	3	2	3	4	3	2	3	3	4	3	3	3	3

Test person 4, model 2

Input p.	Iteration 0					Iteration 1					Iteration 2					Iteration 3				
1	2	4	2	4	4	2	4	2	4	4	4	2	4	4	2	4	2	4	4	2
2	4	4	3	2	2	4	4	3	2	2	4	4	3	3	4	4	4	4	2	2
3	4	4	2	2	2	4	4	2	4	4	4	4	4	4	2	4	4	4	4	4
4	3	2	4	3	2	4	3	2	2	3	4	3	3	3	3	4	4	3	3	4
5	4	2	3	2	2	4	3	3	3	2	4	3	3	3	2	4	3	4	3	2

Appendix B
Input patents

Table B.1 shows all the input patents used in this Master's thesis. Please note that the input patents were "cleaned" before being transformed to embeddings. This meant for example that the text was converted to lower case and figure references and tags were removed. References and tags are removed here as well for readability.

Table B.1: Input patents in their original form before cleaning.

1	AUTOMATIC CLOSING DEVICE FOR FIRE DOOR PROBLEM TO BE SOLVED: To provide an automatic closing device for a fire door capable of bringing a latch mechanism into a release state manually and easily without using a fire door. SOLUTION: When a fire breaks out, an automatic closing device 1 for a fire door can automatically close a fire door by bringing a latch mechanism part 3 into a release state by a start part starting, and by releasing engagement of a latch 3a and an engaging member H of the fire door, and it can also manually close the fire door by bringing the latch mechanism part 3 into a release state by pulling the fire door in a closing direction and by releasing engagement of the latch 3a and the engaging member H of the fire door. In a housing 2, an opening 10 for release operation is provided, and an operation of bringing the latch mechanism part 3 into a release state can be performed from the outside through the opening 10 for release operation. SELECTED DRAWING: Figure 4
2	Outdoor pavilion

	The utility model relates to an outdoor pavilion. The kiosk comprises a base, supporting columns and a kiosk roof. The supporting column is fixed to the base. The pavilion roof is fixed to the end, away from the base, of the supporting column. A roller shutter is arranged between every two adjacent supporting columns. The roller shutter comprises an installation base, a reel used for winding the curtain, a pull disc and the curtain. Wherein the mounting base is fixedly connected with the supporting column, the scroll is rotatably connected with the mounting base, the top of the curtain is fixedly connected with the scroll, the balance weight lever is fixed to the bottom of the curtain, the pull disc is fixed to the end of the scroll, a pull bead rope used for driving the pull disc is arranged on the pull disc, and a fixing piece used for fixing the balance weight lever is arranged on the supporting column. The curtain is hung on the pavilion by pulling the bead pulling rope, the situation that sunlight obliquely irradiates into the pavilion or rainwater floats into the pavilion can be effectively reduced, and the curtain is high in practicability and good in comfort.
3	Lock matched with louver curtain stag cord The utility model relates to a lock set matching with a stretching wire of the window blind curtain, which comprises of a cubic casing and a box without a cover and a bottom, a gear rack of ladder shaped in the cubic chamber as well as a toothed wheel of copper alloy, a fixed-point rotary wheel, i.e. sliding wheel, the copper gear is the stripe gear hole that parallels with the echelonment gear rack, the outer of the chamber entrance wall on the other end of the cubic chamber that facing with the ladder shaped gear rack is provided with a corresponding sliding track. The utility model is simple in structure, easy to process and equip and grand in outline and can be arranged on either the right side or the left side, which gives the customer more choices. Because the utility model has to burden bigger pressure, the casing toothed wheel and firmly locked wheel are all cast and preceded by non-ferrous alloy, the utility model is rigid and wearable, and has a long service life. In addition, the utility model is provided with installed sliding track, and can be integrated with other window blind hardware components when equipping, which will take good use of the space.
4	VENTILATION DEVICE ventilation devices for doors and windows. SUBSTANCE: device is movable between frame of door and separate frame. Frame are provided with sealing strips which may be pushed in and out by means of actuating mechanism, thus opening air gaps between beams of frames. Each actuating mechanism of each sealing cover plate or strip is connected with all actuating mechanisms by means of drive and gates may be placed in at least three positions. EFFECT: facilitated control of device and its mounting. 8 cl, 17 dwg
5	Intelligent monitoring entrance guard control safety protection device convenient for heat dissipation

The utility model relates to the technical field of intelligent monitoring entrance guard control safety protection. The utility model relates to a security protection device, in particular to an intelligent monitoring entrance guard control security protection device facilitating heat dissipation. Door frame, a door plate is hinged to the inner side of the door frame; a positioning hole is formed in the front end surface of the door plate on the left side; the inner side of the door plate is fixedly connected with an electric lock; a baffle is arranged on the front end surface of the door plate on the left side; the baffle penetrates through the door plate on the left side; the baffle plate is slidably connected with the door plate; a handle is fixedly connected to the front end face of the baffle. The top end face of the baffle is fixedly connected with a second spring. According to the utility model, through the arrangement of the baffle, the baffle is matched with the elastic force of the second spring to the baffle, the elastic force of the first spring to the clamping block and the sliding connection between the baffle and the door plate located on the left side, the card reader can be shielded when the device is not used, and therefore the phenomenon that the card reader is damaged due to external collision and other factors is avoided, and the cost investment is reduced.

Appendix C
Bubblers

Test person 3 clearly had the highest values in MRSC and ARSC while also being one of the test persons that had the most improvements in rating. Thus, some of their "bubblers" will be shown here as an example. Some of these have not been seen or rated by the user but would rate higher for this author than some candidates that got rated 4 in the tests.

Table C.1: Some examples of "bubblers" from test person 3. IP is corresponding input patent number, RC is Rating Change (absolute number of steps in rating order), FR is Final Rating (order), R? is either - or a rating if the user has rated the patent.

IP	Patent	RC	FR	R?
1	OPENING PROMOTION LOCK	55	42	-

	PROBLEM TO BE SOLVED: To provide an opening promotion lock capable of eliminating differential pressure (difference in atmospheric pressure) between the inside and the outside of a room without providing a window hole and an opening and closing plate and opening a door being difficult to open due to difference in atmospheric pressure easily without reducing beauty of appearance, airtightness, crime prevention property, and safety. SOLUTION: This opening promotion lock is provided with a latch bolt advancing and retracting from a door opening and closing end 19 on the opposite side to a hoisting origin, an operation shaft making the latch bolt advance and retract, and a strike plate 33 which is provided in a door frame 11 and in which the latch bolt advances to assist opening movement of the door 13 from a condition in which the door 13 is in a closed condition and the latch bolt escapes from the strike plate 33. This lock is provided with a reaction force member 39 provided in the door frame 11, a forcibly opening latch 83 provided to advance and retract from the door opening and closing end 19 and is abutted on the reaction force member 39 by advance movement to pry the door 13 in the direction of opening, and a prizing cam fixed to the operation shaft and makes the forcibly opening latch 83 advance and move by rotation operation of the operation shaft after retracting the latch bolt. COPYRIGHT: (C)2005,JPO&NCIPI			
2	Energy-saving curtain An energy-saving curtain belongs to a curtain and comprises two supporting rods, that is, an upper supporting rod and a lower supporting rod which are parallel; a plurality of curtain leaf blades are vertically connected between the upper supporting rod and the lower supporting rod; the upper supporting rod is connected with a rotating shaft of an electric motor; the upper supporting rod is connected with each curtain leaf blade through a bevel gear pair; a supporting groove for splicing rotating shafts of the curtain leaf blades is formed on the lower supporting rod; and a mirror surface is arranged on the same side of the curtain leaf blades. When in use, the power supply is connected so as to start the electric motor to drive the curtain leaf blades to rotate, and the positions of the curtain leaf blades are adjusted according to the requirements, so as to achieve the required intensity of reflected light, thereby achieving the effects of uniform indoor illumination and energy conservation by adjusting the required illumination intensity through the curtain no matter how the high the illumination intensity is.	310	41	-

4	Curtain lock The utility model discloses a curtain lock, which comprises a lock shell with an inner cavity. A free gear and a rotating part are arranged in the lock shell; the central axis direction of the free gear is substantially parallel to the rotation axis direction of the rotating part; the inner surface of a side wall of the lock shell is provided with a toothed surface on which the free gear can roll up and down; the teeth of the toothed surface are arranged to be matched with the teeth of the free gear; when the free gear is in a clamped state, the free gear is clamped between the rotating part and the toothed surface; and when the free gear is in a loose state, the free gear is separated from the rotating part or the toothed surface; and the two ends of the rotating part are connected to the lock shell by bearings. The rotating part resists a small rolling friction force during the rotation, so a stay can be operated easily with a small force; the heat generated by friction is so little as not to influence the normal service life of the lock shell; and the rotating noise is low because of the small friction force during the rotation of the rotating part, so the influence on the surroundings is avoided.	50	2	4

Appendix D
Questions for test persons and answers

D.1 Questions

The following questions were asked to the test persons after taking the tests:

1. Was the program easy to work with? Were there any parts that were harder to understand?

2. How difficult was it to understand the patents? Were there any parts that were harder to understand?

3. Did you notice an improvement over the iterations?

4. Did you notice any difference in final results due to how similar the top candidates were from the beginning for each input patent?

5. Did you notice any significant difference between the models?

6. Was there something about the program that did not work so well?

7. Is this method something that you would be open to work with in the future?

8. Any other thought?

D.2 Answers

The answers to the questions were often similar between the users:

1. Most answers were that was easy, and that it was good to have the possibility to redo your actions if you did something wrong.
2. Some answers were that it was difficult, others thought the difficulty to vary between patents. Especially difficult were the patents that were mechanical to their nature, where images would be handy.
3. The common answer was yes, or generally yes.
4. Two test persons answered no. One test person answered that good starting patents implicated fewer new good patents later. One test person answered that the best improvements happened when the original candidates were neither good or bad, while also being easy to separate into one bad and one good set.
5. Two test persons answered no. One person thought that the second model was slightly better. One test person expressed worries that you did less work for model 2 since you already had rated most of the patents since working with model 1 and that this could effect your judgement.
6. The answers included being able to see images in patents if available, and a better user interface.
7. The answers were either yes, or yes as a complement to existing methods.
8. One test person expressed concern about patents where a very small part was very relevant. Does the method still work? One user expressed an interest in being able to rate the patents on two different criteria: one for being in the right technical field and one for being relevant as prior art.

Appendix E
Populärvetenskaplig Sammanfattning

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2021-10-28

EXAMENSARBETE Interactive Iterative Patent Search
STUDENT Daniel Jogstad
HANDLEDARE Pierre Nugues (LTH), Fredrik Edman (LTH), Henrik Benckert (Mindified)
EXAMINATOR Jacek Malec (LTH)

Interaktiv Iterativ Patentsökning

POPULÄRVETENSKAPLIG SAMMANFATTNING **Daniel Jogstad**

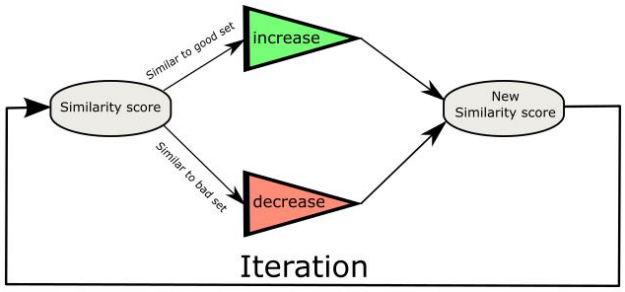
Vid sökning efter matchande dokument i stora patentdatabaser används metoder för att effektivisera sökningen. Detta arbete har undersökt möjligheten att förbättra resultat från en sökning med hjälp av Sentence embeddings i en för användaren interaktiv, iterativ process.

De senaste åren har användandet av neurala nätverk blivit allt vanligare för att hitta semantiska likheter mellan texter. Dessa nätverk blir tränade på en stor mängd data för att skapa numeriska vektorrepresentationer, så kallade embeddings, av ord och meningar. Vektorer kan enkelt jämföras matematiskt för att se hur lika varandra de är. Därmed reducerar man den, för en maskin, komplicerade processen att säga hur lika två texter är varandra, till ett enklare matematiskt problem.

Eftersom det inte alltid finns ett exakt svar på hur lika olika texter är varandra, är resultatet av en jämförelse väldigt subjektiv. Detta avspeglar sig i att det uppstår skillnader i vad ett program tycker är den mest lika träffen till en text och vad användaren tycker.

I detta examensarbete har jag använt mig av en databas med titlar och abstrakt från 310 024 patent. Varje patent har representerats av en sammanslagen titel och abstrakt översatt till embedding-format av ett SBERT-nätverk. Sökningar kan då ske, där man jämför en embedding från ett input-patent med alla andra embeddings i databasen. Resultatet från en sökning är en sorterad lista med de, för input-patentet, matematiskt mest lika träffarna.

Examensarbetet har haft som syfte att testa om det går att ranka om resultaten från en sådan sökning efter vad användaren tycker om de högst rankade patenten. Genom att interaktivt markera för programmet vilka träffar som var bra och dåliga, får man ett underlag för en ny sökning. Sökningen kan därmed göras om från början, men med skillnaden att man nu har en större mängd patent som man vill att de nya träffarna skall bli lika, och även en mängd patent som man vill att de inte skall bli lika. Genom att göra detta i flera steg blir processen iterativ.



Jag har testat 2 enkla matematiska metoder för att ranka om träffar. En testgrupp har sedan testat och utvärderat resultat från metoderna i 3 iterationer för 5 olika input-patent. Resultaten visar att testgruppens betyg för de bästa träffarna växer med stigande antal iterationer och att modellerna därmed fungerar.