

Similarity-Based Patent Selection using Natural Language Processing

Master's thesis in Computer Science: Algorithms Languages and Logic

Elmar Aliyev

MASTER'S THESIS 2021

Similarity-Based Patent Selection using Natural Language Processing

ELMAR ALIYEV



UNIVERSITY OF
GOTHENBURG



CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2021

Similarity-Based Patent Selection using Natural Language Processing
ELMAR ALIYEV

© ELMAR ALIYEV, 2021.

Supervisor: Carl-Johan Seger, Department of Computer Science and Engineering
Examiner: Magnus Myreen, Department of Computer Science and Engineering

Master's Thesis 2021
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Growth Curves for Carburetor and Fuel Injection technologies (see Section 5.2.2.2).

Typeset in L^AT_EX
Gothenburg, Sweden 2021

Abstract

Many companies spend a lot of resources and put significant effort into R&D activities to keep themselves informed of the latest advances in technology. As patent data is the world's largest technology repository, it is frequently utilized by technology managers for this purpose. The patent analysis of this kind usually involves much manual works, for example collecting patents to represent technology fields. It has been observed that creating such patent sets is the most critical part since poor patent selection would lead to biased results, no matter how well the analysis is performed. Manual nature, on the other hand, makes the quality of the patent selection process questionable.

This thesis studied the subject and proposed a novel method (called "SBPS") that assists users in building effective queries and, based on these queries, finds representative patents for technology fields.

The proposed method is divided into three main stages, namely query building, similarity calculation, and threshold finding. The essence of the first stage is offering synonyms to the user's query through the use of trained word embeddings. The second stage involves employing a keyword extraction algorithm for calculating document vectors and the cosine similarity measure for ranking documents based on similarity to the query. The third stage requires the adjustment of the similarity threshold between the range of 0 and 1. This manual step lets the users to define the degree of patent relatedness to the query.

To evaluate the method, four technology battles were studied from the development history viewpoint and compared to the histogram and growth curve graphs extracted for the corresponding technologies using the SBPS method. The results from the comparative analysis showed significant agreement between the historical events and the graphs and proved the potential of the proposed method.

Keywords: Patent, NLP, word2vec, similarity, technology, patent search, technology watch.

Acknowledgements

Foremost, I would like to express my warmest thank to my supervisor, Carl-Johan Seger, for his guidance, motivation, patience, immense knowledge, and invaluable thoughts which made the completion of the thesis possible.

My sincere thanks go to Erik Hjerpe and Volvo Cars Corporation, for giving access to the data and showing their support for the thesis project.

Finally, I would like to show my deepest gratitude to my family (especially my girlfriend) and my friends who have been the source of motivation for me throughout the thesis.

Elmar Aliyev, Gothenburg, June 2021

Contents

List of Figures	xi
List of Tables	xiii
1 Introduction	1
1.1 Context	1
1.2 Problem Formulation	2
1.3 Aim	2
1.4 Research Question	2
1.5 Related Work	3
2 Theory	5
2.1 Keyword Extraction Techniques	5
2.1.1 TF-IDF	5
2.1.2 TextRank	6
2.2 Text Vectorization Techniques	6
2.2.1 TF-IDF Vectorizer	6
2.2.2 Word2Vec	7
2.2.3 Doc2Vec	9
2.3 Dimensionality Reduction Techniques	10
2.3.1 Principal Component Analysis (PCA)	10
2.3.2 LargeVis	10
2.4 Clustering Algorithms	11
2.4.1 K-Means	11
2.4.2 DBSCAN	13
2.5 Evaluation Metrics	14
2.5.1 Silhouette Score	14
2.5.2 Calinski-Harabasz Index	15
3 Patent	17
3.1 Patent History	18
3.2 Use of Patent Information	18
3.2.1 Limiting Factors	19
3.3 Patent Data	19
4 Methods	21
4.1 Preprocessing	21

4.1.1	Tokenization	21
4.1.2	Stopword removal	22
4.1.3	Lemmatization	22
4.1.4	Frequency-based word removal	23
4.2	Clustering-Based Patent Selection	23
4.3	Similarity-Based Patent Selection (SBPS)	24
4.3.1	Query Building	24
4.3.2	Calculating Similarity Scores	24
4.3.3	Determining Similarity Threshold	25
5	Results	27
5.1	Verification via Single Technology Sector	27
5.2	Analysis on Multiple Technology Sectors	27
5.2.1	Failed Experiments	28
5.2.1.1	Parameter Tuning through Sanity Checking	28
5.2.1.2	Parameter Tuning through Cluster Evaluation	29
5.2.2	Successful Method: SBPS	29
5.2.2.1	SBPS as Command Line Application	29
5.2.2.2	Carburetor vs. Fuel Injection	30
5.2.2.3	Floppy Disk vs. Optical Disc vs. USB Flash Drive	31
5.2.2.4	Fluorescent vs. LED	33
5.2.2.5	SONAR vs. RADAR vs. LIDAR	35
6	Discussion	41
6.1	Future Work	42
A	Appendix 1	I

List of Figures

2.1	CBOW architecture of Word2Vec	8
2.2	PV-DM architecture of Doc2Vec	9
2.3	A typical pipeline of data visualization by first constructing a K-NNG and then projecting the graph into a low-dimensional space. [1] . . .	11
2.4	K-means at local optimum. The different shapes (circle, square, triangle) indicate different clusters and the centroids are denoted by '+' [2]	13
3.1	Front page of a sample patent document [3]	17
3.2	S-shaped Growth Curve for Technology Field	19
5.1	Example of Query Building Process	30
5.2	Growth Curve of Carburetor Technology	31
5.3	Growth Curve of Fuel Injection Technology	31
5.4	Growth Curve of Floppy Disk Technology	32
5.5	Growth Curve of Optical Disc Technology	33
5.6	Growth Curve of USB Flash Drive Technology	33
5.7	Growth Curve of Fluorescent Technology	34
5.8	Growth Curve of LED Technology	35
5.9	Histogram for Fluorescent Technology	35
5.10	Histogram for LED Technology	35
5.11	Cumulative number of patent applications [3]	37
5.12	Growth Curve of SONAR Technology	37
5.13	Growth Curve of RADAR Technology	37
5.14	Growth Curve of LIDAR Technology	38
5.15	Yearly number of patent applications [3]	38
5.16	Histogram for Sonar Technology	38
5.17	Histogram for Radar Technology	39
5.18	Histogram for Lidar Technology	39

List of Tables

3.1	Tables names in the PATSTAT database	20
4.1	Result of tokenization of raw abstract	21
4.2	Result of removing stopwords from tokenized abstract	22
4.3	Result of applying lemmatization on tokenized and stopword-free abstract	22
4.4	Result of the removal of frequent words from tokenized, stopword-free, and lemmatized abstract	23

1

Introduction

Fast technological changes bring increasingly difficult questions to incumbent companies. Actors have to continuously adjust their technological offerings to stay relevant. However, choosing the right technology to invest in or to scrap is a serious and difficult task on its own. For instance, will there be a standard for electric vehicle charging, and if so, will it be cordless? Which sensing technologies are likely to be most used in autonomous cars in ten years?

Companies usually use patent analytics to predict which technologies will succeed. Given the fast growth rate of some technologies in the patent landscapes, they can deduce that those technologies are developing swiftly and might eventually emerge as an industry standard. Even though this kind of analysis is performed frequently, companies lack a thorough understanding of what conclusions it actually supports.

The rate of advancements in the digital world has been increasing continuously over the decades and it opens countless doors to new opportunities in many areas. Patinformatics, “the science of analyzing patent information to discover relationships and trends” [4], is no exception. In fact, David et al. [3], through their careful examination of the related area, observed a noticeable increase in the use of patent information over the recent years as a way of assisting organizations in the business-critical technology positioning task. However, they concluded that patent data is not yet utilized effectively due to the research made in the field is fragmented. Therefore, a need for an overarching framework is prominent.

1.1 Context

This master thesis work is inspired by a research project [3] started in 2018 with the collaboration between Volvo Cars Corporation, Chalmers University of Technology, and Sahlgrenska School of Innovation and Entrepreneurship. The main aim of that research project was to provide technology managers with a framework to help them make more fail-safe decisions regarding positioning the companies in the technologically fast-changing environment. The main question of the research was whether the patent data is a relevant data source to be employed for extracting such insightful information to be used in decision-making. Therefore, they thoroughly studied three frameworks and incorporated them in an appropriate manner to construct a new, comprehensive patinformatics framework. They conducted a case study on three sensor technologies, namely lidar, radar, and sonar, in order to

test and evaluate the proposed framework. The reason for the choice of these sensor technologies was that the competition between them was exceptionally interesting from the aspect of enabling fully autonomous vehicles. It is still unknown which technology will lead the industry in deploying such vehicles. Despite the difficulties in explaining some metrics results without the use of expert knowledge, the authors were able to extract numerous valuable insights and reveal some hidden dynamics of the technology battle by applying the proposed framework on the cases.

The patent data sets are extremely large and contain well-structured information about technologies. These data sets are frequently involved in the R&D-related activities (usually called “technology watch”) of the most companies to get their technology offerings up-to-date.

European Patent Office provides a database containing information of about 100 million patent documents. The database is called PATSTAT and contains both technical and legal information. This well-structured, and tremendous amount of information about technology fields creates opportunities where many sophisticated Data Science methods can be used to extract patterns hidden to the naked eye.

1.2 Problem Formulation

The promising results obtained by David et al. [3] showed the potential of their framework; however, the effectiveness of the framework is highly dependent on how well the selected patent set represents the technology field. In their study, the authors retrieved the representative patent sets through a manual process. For this purpose, they utilized the sophisticated online tool known as CIPHER. This tool can be considered as a search engine for patents. The manual fashion of this kind makes the analysis tedious and prone to mistakes. Therefore, decreasing human interaction in the patent selection process is necessary for achieving reliable and unbiased results.

1.3 Aim

The main aim of this thesis is to investigate the problem discussed above (see Section 1.2) and develop a method to automate the representative data selection process for patent analysis by making use of a patent database such as PATSTAT.

1.4 Research Question

This thesis will answer the following main research question: “*How can unstructured textual information found in a patent database like PATSTAT be used to construct a patent selection system?*”. Since the scope of the main question is large, we have broken it down into two sub-questions:

1. *What unstructured textual information can be found in the PATSTAT database and which ones can be used for our purpose?*
2. *Which algorithms can be utilized to build a patent selection system? What value do they add?*

1.5 Related Work

In this section, we reviewed previous studies to find out what algorithms are employed by other researchers on patent data for similar purposes.

Query building in Information Retrieval systems is not a trivial task. Users are usually required to come up with an accurate query and it can be difficult depending on the user's knowledge of the domain. To assist users, there have been several studies conducted and several methods developed. Pseudo-relevance feedback method is one of them in which a query submitted by a user is expanded with the terms extracted from the top k documents that are assumed to be relevant to the initial query [5; 6]. While the authors concluded that the method improves the overall retrieval effectiveness, Magdy et al. [7] found the PRF method ineffective in patent search tasks. Another popular approach is to utilize dictionaries to expand the words in query with their synonyms. Using standard dictionaries like WordNet is found to be not helpful [7; 8]. On the other hand, Tannebaum et al. [8; 9] reported that building a domain-specific term dictionary out of the query logs generated by the patent examiner improves the results considerably. However, Zhang et al. [10] pointed out that such query logs still need to be updated manually; instead, they, as Verma et al. [11], proposed methods that make a patent search based on a generated query from the given patent document. Singh et al. [12] stated that by combining the Pseudo-relevance feedback method with the Word2Vec method the retrieval efficiency increases significantly.

Helmets et al. [13] proposed a method that finds the relevant patent set by calculating the similarity between documents. Their approach for building query requires a small set of manually selected patents which is later expanded with related patents found based on citation links. Thereafter, the query patents are transformed into feature vectors on which the cosine similarity measure gets calculated. For the feature vector transformation task, they tried several techniques, namely tf-idf based bag-of-words (BOW), Word2Vec, and Doc2Vec. They found out that Word2Vec and Doc2Vec generally perform better when a portion of text data, such as abstract, claims, etc. is considered; however, for the case of taking all text into account, BOW with tf-idf outperforms the other alternatives.

2

Theory

2.1 Keyword Extraction Techniques

Keywords are words in a text document that best describe the content in the briefest, yet concise manner. Keywords are widely used by information retrieval systems to index documents or to build search queries. Keywords play a vital role in large systems as a way to improve the performance of typical operations over the text documents. Since most documents do not get tagged with keywords, automatic keyword extraction techniques become the inevitable part of the studies that involve dealing with texts. In this study, we considered the following two methods for automatically extracting keywords out of the text documents: TF-IDF and TextRank.

2.1.1 TF-IDF

TF-IDF is the combination of two statistical measures, Term Frequency (TF) and Inverse Document Frequency (IDF). It elegantly incorporates these two measures to complement one another.

TF is a document-wide statistic that measures the usage frequency of terms in a given document. It is calculated by the following equation:

$$\text{tf}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

where $f_{t,d}$ is the number of occurrences of a term t in document d .

However, IDF is a corpus-wide statistic that measures the rareness of a term across all documents. It gets higher values for the rare words and lower values for the common words. It is calculated by the following equation given a set of documents D :

$$\text{idf}(t, D) = \log \frac{N}{df(t, D)}$$

where N is the number of documents in D and $df(t, D)$ is the number of documents in D that contain a term t .

Having tf and idf functions defined, TF-IDF score is computed by the following equation:

$$tfidf(t, d, D) = tf(t, d) * idf(t, D)$$

The *tfidf* function gets a high value when a term appears frequently in a document but rather rare in the entire corpus. It combines the advantages of *tf* and *idf* statistics by penalizing the score for very common terms in the corpus (e.g. articles, pronouns, prepositions) and very unique terms (e.g. typos).

With the right choice of threshold on scores, TF-IDF can be used as a keyword extraction technique.

2.1.2 TextRank

The TextRank algorithm [14] has emerged as an extension of a so-called PageRank algorithm in the Natural Language Processing domain. PageRank [15] was invented by researchers at Google and it is currently the algorithm running in the Google Search to evaluate the importance of websites. The main reasoning behind this algorithm is that the more important websites should get many references from other websites that are important too. The algorithm makes use of a directed graph, where the vertices are the websites and the edges represent links between them. The algorithm first starts with assigning arbitrary scores to the vertices. Then it updates the scores iteratively until no-change state is observed. The scores represent the importance of the vertices in the graph, translating to the importance of the websites (the higher the score, the higher the importance).

The TextRank algorithm is a special instance of the PageRank algorithm which considers words in a text as vertices and the neighborhood relationship between those words as edges. Running the algorithm on such a graph results assigning importance score for each word by taking into account the context that the word appears in. Then, these scores are used to rank the words and pick the top n words as keywords. The claim is that a word with the highest score should best describe the text given.

2.2 Text Vectorization Techniques

In natural language processing (NLP), text vectorization is an unavoidable step applied during the pre-processing phase. This is mainly because there yet exists no NLP and machine learning technique that can directly operate on the raw form of the NLP data. That is why a text must be transformed into its corresponding numerical representation for enabling the application of the available techniques. In this thesis, we reviewed three well-known techniques for text vectorization: TF-IDF vectorizer, Word2Vec, and Doc2Vec.

2.2.1 TF-IDF Vectorizer

The TF-IDF vectorizer is based on the *tf-idf* score described in section 2.1. It is essentially a data representation technique incorporating *tf-idf* method. The output from applying this technique is a two-dimensional matrix. Its rows and columns

represent document and term vectors respectively. The values in the matrix cells are the tf-idf scores of the terms and the documents at the intersection. Here is the illustration of such a matrix given $\{d_1, \dots, d_n\}$ documents, $\{t_1, \dots, t_m\}$ unique terms and tf-idf score (weight) of i^{th} document and j^{th} term as $w_{i,j}$:

	t_1	t_2	...	t_{m-1}	t_m
d_1	$w_{1,1}$	$w_{1,2}$...	$w_{1,m-1}$	$w_{1,m}$
d_2	$w_{2,1}$	$w_{2,2}$...	$w_{2,m-1}$	$w_{2,m}$
\vdots					
d_{n-1}	$w_{n-1,1}$	$w_{n-1,2}$...	$w_{n-1,m-1}$	$w_{n-1,m}$
d_n	$w_{n,1}$	$w_{n,2}$...	$w_{n,m-1}$	$w_{n,m}$

As it can be seen from the table above, the output matrix is a sparse matrix since every document comprises of only a small subset of all the terms in the vocabulary and, therefore, matrix should contain lots of zeros.

2.2.2 Word2Vec

The Word2Vec model was proposed by researchers at Google to produce high-quality word representations [16]. It is based on an artificial neural network. It has a simple architecture with only one hidden layer which makes it lightweight enough to be applied on very large data sets. According to Tomas et al. [16], it does not only provide efficiency from the time perspective, but produces word vectors that capture the semantic meaning and keep the subtle relationships between the corresponding words. They used the basic algebraic properties of the vectors to justify the argument and show the usefulness of the word embeddings acquired from Word2Vec through the following simple but yet interesting example [17]:

$$\text{vector}(\text{"King"}) - \text{vector}(\text{"Man"}) + \text{vector}(\text{"Woman"}) \simeq \text{vector}(\text{"Queen"})$$

The pairwise similarity between the query vector and word vectors is calculated with the cosine similarity measure. The score range for this measure is between -1 and 1; the higher the similarity score, the more related the word is. In the example given above, the word “Queen” had the highest similarity score compared to the other words in the vocabulary.

Word2Vec was proposed in two different architectures: Continuous Bag-of-Words (CBOW) and skip-gram.

In the CBOW architecture, the model is trained in a way to predict the target words given their contexts [16]. The context is the surrounding words of the target word in a pre-defined window.

... models such as word2vec and doc2vec are NLP tools ...

w_{t-2} w_{t-1} w_t w_{t+1} w_{t+2}
 window with radius 2

As shown in the above illustration, the Word2Vec algorithm aligns a window in a way to centralize the target word (w_t) and pick the surrounding words from both

sides according to the window size. Then it feeds the neural network model (NNM) with the context words as one-hot vectors¹ and gets a vector in the output layer containing the scores for each word in the vocabulary and then, applies softmax function to convert them to probabilities. By comparing this resulting vector with the one-hot vector for the target word, it calculates the error and updates the weights accordingly using back-propagation method like a normal feed-forward NNM [2]. This process is repeated for every word in the corpus as the window shifts over them one by one. The following figure shows the CBOW architecture on the high level when the window radius is 2:

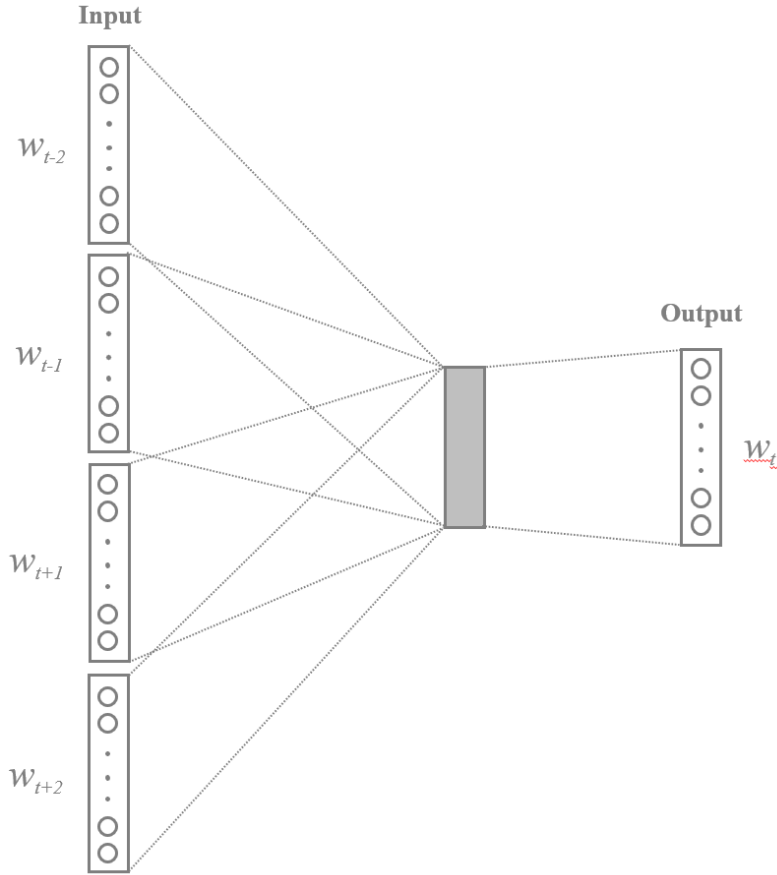


Figure 2.1: CBOW architecture of Word2Vec

The skip-gram architecture is the exact opposite of CBOW. It takes one-hot vector of a single word and tries to predict the context words.

Thomas et al. [16] showed in their studies that a model trained with skip-gram outperforms the one trained with CBOW in capturing the semantic meanings of the words. They also found that increasing the vector size (number of neurons in the hidden layer) usually improves the quality of the word embeddings, but training high dimensional word vectors requires a large amount of data [16]. One of their arguments was also that it should be possible to find the semantically correct answers by applying simple vector algebra if the word embeddings are trained well [16].

¹A boolean vector with a size of vocabulary having 0s in all elements except for the element representing the word in question

2.2.3 Doc2Vec

The Doc2Vec algorithm is based on “Paragraph Vector” which is an unsupervised algorithm proposed by researchers at Google [18]. It is seen as a natural extension of Word2Vec as it is also a neural network model based and their architectures are similar. Unlike Word2Vec, this model is capable of producing fine-tuned document embeddings. Le and Thomas [18] inspired by the working principle of the Word2Vec algorithm when developing Doc2Vec. They observed that the Word2Vec model learns word vectors as a result of predicting the target word given the context words. This observation made them to realize that including document vectors in the same prediction process results in training these vectors to a degree which the semantic meaning of the documents is captured.

The following figure shows the architecture of Doc2Vec at the highest level.

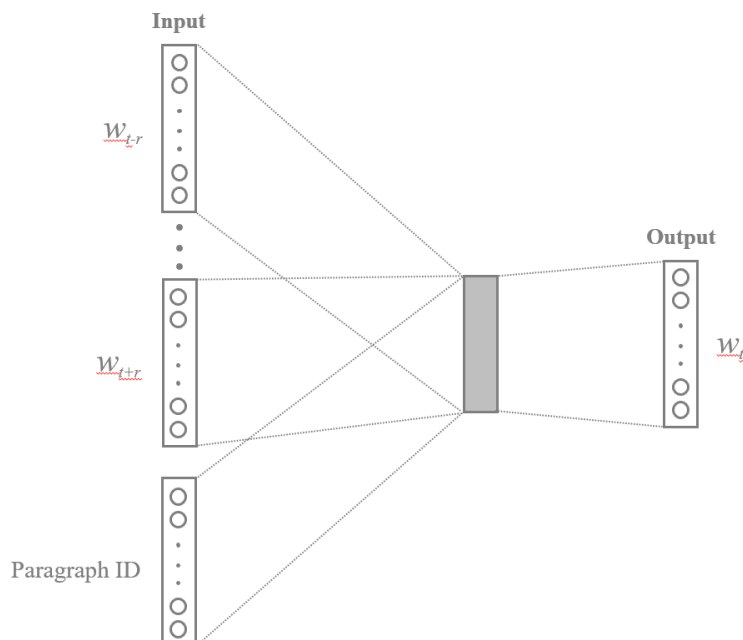


Figure 2.2: PV-DM architecture of Doc2Vec

Similar to the Word2Vec model, Doc2Vec has two types: distributed memory model of paragraph vectors (PV-DM) and distributed bag-of-words (DBOW).

PV-DM architecture is shown in the figure above. The additional paragraph id plays a role of an extra context word in the model which, as authors stated, “acts as a memory to remember the missing information from the current context” [18]. This exact behavioural property gave rise to the name of this architecture. On the contrary, in DBOW architecture the context words do not get passed as inputs. The only input is the paragraph id while context words are treated as the output words to be predicted. This architecture is equivalent to skip-gram architecture in Word2Vec. The main difference of DBOW compared to the PV-DM is that the word ordering is not taken into consideration since the words are sampled from the text window. Ignoring the order of words generally improves the performance of the algorithm,

but it also means losing information which leads to poor results for the tasks where word orders matters. Le and Tomas [18] observed in their experiments that PV-PM model produces sufficiently good paragraph embeddings, but the combination of the document/paragraph vectors acquired from both variants is more consistent and improves the results for most tasks.

2.3 Dimensionality Reduction Techniques

A vast amount of data is produced every minute, most of which is very complex in structure. While this volume of data creates opportunities, it also challenges the users to get hold of the data efficiently and effectively. Much work is being carried out by researchers, organizations, etc. to develop new techniques to transform the complexity and vastness in the data to a form that is comprehensible by a normal human brain. The dimensionality reduction techniques are from this family. They are developed to reduce the complexity in terms of dimensions while preserving structural integrity.

Considering the size of the patent database and expected high-dimensional feature vectors from the text vectorization techniques, reducing dimensionality is the inescapable step to take before both the clustering and data visualization steps. Therefore, we considered two methods: PCA and LargeVis.

2.3.1 Principal Component Analysis (PCA)

PCA is one of the most popular statistical techniques that is used to explore high-dimensional data for extracting important information (i.e. principal components). PCA is a linear approach to the problem that performs projection of the data with high dimensions into a linear lower-dimensional subspace where the variance in the data is maximal [19]. Maaten et al. [19] described it as an optimization problem of finding a linear mapping M formed by d principal eigenvectors of the covariance matrix of the data X that maximizes the following cost function:

$$\lambda = M' cov(X) M$$

The popularity of PCA comes mainly from its versatility. While it has successful applications in many different domains, PCA has several major drawbacks. Maaten et al. [19] argue that the application of PCA might be a costly operation since the size of the covariance matrix is proportional to the dimensionality of the data. They also claim that keeping the small pairwise distance between the data points is more important, but PCA mainly focuses on the opposite. On the other hand, Tang et al. [1] describe the performance of linear techniques such as PCA as insufficient by arguing that the most high-dimensional data usually does not lie on a low-dimensional linear manifold.

2.3.2 LargeVis

Unlike PCA, LargeVis is a non-linear technique for reducing dimensions. It is inspired by the t-SNE algorithm [1; 20]. In this study, we did not focus on t-SNE

due to the known issues such as high computational cost and parameter sensitivity on different datasets [1]. LargeVis is developed to address these issues. Its working principle can be broken down into two main stages: computation of a similarity structure and projection into a subspace. That being said, LargeVis first constructs an approximate K-nearest neighbor graph (K-NNG) and then uses a probabilistic model for projecting the similarity graph on a low-dimensional space by preserving the intrinsic structure [1]. The following figure illustrates the process at a high level.

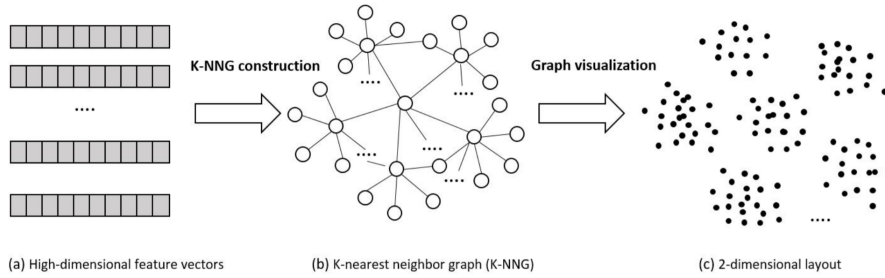


Figure 2.3: A typical pipeline of data visualization by first constructing a K-NNG and then projecting the graph into a low-dimensional space. [1]

Tang et al. [1] claim that the construction of the similarity graph with the approximate K-nearest neighbor method outperforms the state-of-art methods and the efficiency of the algorithm increases proportionally to the size of the data. They observed that parameters of LargeVis are stable across the different datasets. In spite of the advantages, LargeVis produces non-deterministic results and this can be thought of as a weakness compared to PCA.

2.4 Clustering Algorithms

Data categorization is often required in problems where data is grouped based on similarities between the observations to help revealing hidden, yet valuable information such as patterns, trends and so on. There are two sets of algorithms serving similar purposes: classification and clustering algorithms. While classification algorithms belong to the supervised learning techniques where the category labels are known a priori, the clustering algorithms are unsupervised which means no information is known about the categories.

In this thesis, we considered clustering algorithms since the technology groups that patents belong to are unknown and the aim is to discover the new technologies. Clustering algorithm can be divided into two main categories: distance-based and density-based [21]. We employed two well-known algorithms, one from each category, K-means and DBSCAN respectively.

2.4.1 K-Means

K-Means is a popular and widely used algorithm because of its simplicity and computational efficiency [22]. It requires only one parameter to be passed prior to the

execution which is the number of clusters. So, given the number of clusters, the algorithm assigns each data point to a single cluster based on the proximity to the cluster centroids¹ in the euclidean space.

The K-Means algorithm starts with initializing the centroids randomly. Then it iteratively performs two main actions, namely assignment and update in order until convergence. Assignment is the step where the data points are assigned to their nearest centroids. Once all the data points are visited, the algorithm updates the centroids to the new center points of the clusters. This process is repeated until no change is detected in the update step, which is called convergence.

Given the data set $\{x_1, x_2, \dots, x_n\}$ and the centroids $\{\mu_1, \mu_2, \dots, \mu_k\}$, the k-means algorithm produces clusters $\{c_1, c_2, \dots, c_k\}$ by minimizing the objective function known as sum of squared errors (SSE):

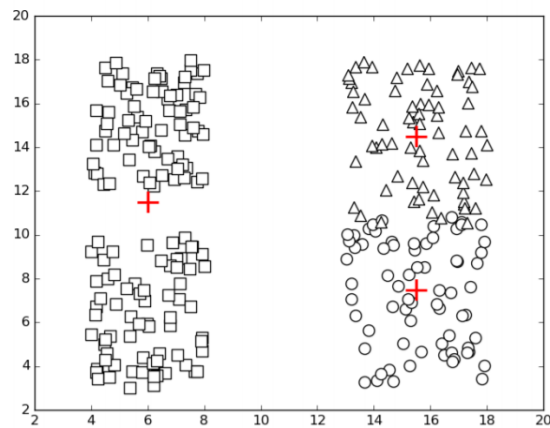
$$SSE = \sum_{i=1}^n \sum_{j=1}^k w_{ik} ||x_i - \mu_j||^2$$

where

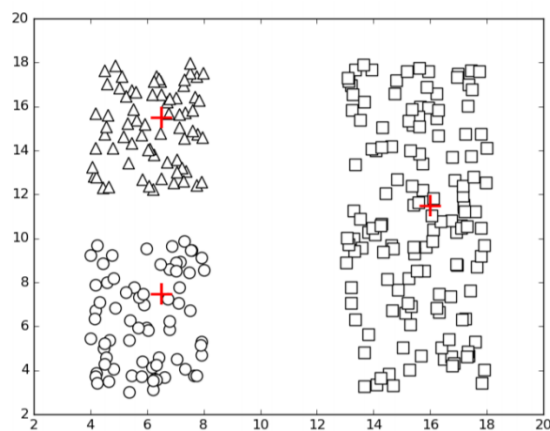
$$w_{ik} = \begin{cases} 1 & x_i \in c_i \\ 0 & otherwise \end{cases}$$

Celebi et al. [22] argue in their study that the quality of the clusters is highly sensitive to the initial placement of the centroids because of the gradient descent nature of the algorithm. They have evaluated many initialization methods having linear complexity such as forgy, macqueen, maximim, k-means++ and so on. They found out that k-means++ performs better in most cases compared to the others. The Figure 2.4 shows the difference of the outputs of k-means with both poor and good initializations of the center points. Dash et al. [21] and Ester et al. [23] pointed out that besides the advantages, the k-means algorithm fails in finding arbitrary shaped clusters and identifying outliers.

¹Central point of a cluster



(a) K-means at wrong local optimum



(b) K-means at correct local optimum

Figure 2.4: K-means at local optimum. The different shapes (circle, square, triangle) indicate different clusters and the centroids are denoted by '+' [2]

2.4.2 DBSCAN

DBSCAN is a density-based algorithm that relies on the idea that clusters have dense structure compared to their surroundings [23]. Unlike K-Means, it does not require a prior knowledge about the number of clusters which means it automatically finds all the clusters that fulfil density requirements. It also addresses the shortcoming of K-Means to find the clusters with arbitrary shapes and detect noise [21].

The DBSCAN algorithm requires two parameters before it can be applied on the data set such as neighbourhood radius (ϵ) and the minimum number of points in the neighborhood ($MinPts$). Having these parameters initialized, the algorithm iterates over the points in the dataset and for each point, it finds the number of neighbours within its ϵ -radius. If the number of neighbour points is less than the minimum threshold $MinPts$ and it is not a member of any cluster, then the point is classified as noise and the algorithm continues with the next point. Otherwise, the point is marked as a cluster point and the search is expanded to its neighbours.

The algorithm repeats the same process until all points are visited. The pseudocode of the algorithm [21] is as follows:

Algorithm 1 Density-based Clustering

Input: Data $\{X_1, \dots, X_N\}, \epsilon, MinPts$

Output: Dense Clusters

```

1: ClusterId = 1
2: for  $i = 1$  to  $N$  do
3:   if  $X_i$  is UNVISITED then
4:     if  $X_i$  is not NOISE then
5:       ExpandCluster( $X_i$ ) /*finds all cluster points connected via  $X_i^*$ */
6:       ClusterId = ClusterId + 1
7:     end if
8:   end if
9: end for

```

Dash et al. [21] stated in their study that DBSCAN algorithm has a couple of drawbacks such as facing difficulty in setting the right density threshold and being slow compared to K-Means.

2.5 Evaluation Metrics

The metrics we considered are for evaluating the quality of the clusters. The cluster evaluation metrics are divided into two categories, extrinsic and intrinsic, depending on the availability of the labeled data. Since there exists no information about the true labels of the patents and it is impossible to create such a dataset without the help from the experts in the domain, we are constrained to focus on only the intrinsic evaluation metrics. Therefore, we have reviewed two of the most popular metrics such as Silhouette score and Calinski-Harabasz index.

2.5.1 Silhouette Score

Silhouette score was proposed by Rousseeuw in 1987 [24] as a tool to assess the quality of the clusters from both the compactness and separation points of view. In other words, silhouette score is at its maximum if the data points in the clusters are close to each other while they positioned far from the neighboring clusters. In the mathematical terms, given k clusters ($k > 1$), let A be the cluster to which the data point i is assigned and B be the closest cluster to A , the silhouette score of i is calculated as

$$s(i) = \frac{d_i(A) - d_i(B)}{\max\{d_i(A), d_i(B)\}}$$

where

$$d_i(X) = \frac{1}{|X|} \sum_{x \in X} \|x - i\|$$

It can be seen from its equation, silhouette score is defined between the range $[-1, 1]$. Rousseeuw [24] gave clear definitions of these values: the score of 1 means the data point is well-clustered; -1 means it is assigned to a wrong cluster; and 0 means that it is unclear whether the data point should have been assigned to A and B since it is positioned in between.

As $s(i)$ communicates for one data point, the average score of all data points indicates the overall quality of the clusters. In fact, Lovmar et al. [25] found silhouette score very useful for assessing the quality of clusters and Shahapure et al. [26] observed through several experiments that the silhouette score is a powerful method in finding the near-optimal number of clusters for the algorithms such as k-means.

2.5.2 Calinski-Harabasz Index

Calinski-Harabasz index [27] is another intrinsic method for evaluating the assignments from the clustering algorithms. It is similar to the silhouette score in the sense that it is also based on within-cluster and between-clusters relationship but from the variance point of view. The higher values in this index correspond to better clustering results which translate to lower within-cluster variance and higher between-clusters variance [28]. However, since there is no upper bound on its value range, this method does not give the same assurance as the silhouette score does.

3

Patent

The patent is a type of intellectual property that grants the ownership of an invention or a technology to a patent holder. It enables the owners to have a monopoly of the inventions in countries or regions by restraining the competitors from making use of the invention or technology. In case of infringement, patent holders have the right to sue the other parties to enforce them for complying with the claims of the patents. Besides the protection, it is also an effective way of officially disclosing the invention to the public which contributes to the development of future technologies. As an example, the following figure shows the front page of a sample patent document:

US006664529B2

Patent No. **US 6,664,529 B2**

Date of Patent: **Dec. 16, 2003**

(12) United States Patent
Pack et al.

(35) Title: 3D MULTISPECTRAL LIDAR

(75) Inventors: Robert Taylor Pack, Logan, UT (US);
Frederick Brent Pack, Waipahu, HI (US)

(73) Assignee: Utah State University, North Logan, UT (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 27 days.

(21) Appl. No.: 10/052,825

(22) Filed: Jan. 15, 2002

(65) Prior Publication Data
US 2002/0060784 A1 May 23, 2002

(63) Continuation of application No. 09/909,165, filed on Jul. 19, 2001.

(60) Provisional application No. 60/219,725, filed on Jul. 19, 2000.

(51) Int. Cl.: H01L 27/00

(52) U.S. Cl.: 250/208.1; 356/4.01

(58) Field of Search: 250/208.1, 206/1, 250/203.2, 559.38; 244/3.16, 3.17, 5.18; 356/73, 3.01, 4.01, 5.01

(56) References Cited
U.S. PATENT DOCUMENTS
5,006,721 A * 4/1991 Cameron et al. 250/559.38
5,101,108 A * 3/1992 Galema et al. 250/559.38
5,177,556 A * 1/1993 Rioux 356/73
5,231,401 A * 7/1993 Kaman et al. 356/5.04
5,270,780 A * 12/1993 Moran et al. 356/5.04
5,446,529 A 8/1995 Stettner et al.
5,696,577 A 12/1997 Stettner et al.
5,796,471 A 8/1998 Wilkerson et al.
5,822,047 A 10/1998 Contarino et al.
5,835,204 A 11/1998 Urbach
5,870,179 A 2/1999 Cathey, Jr. et al.

FOREIGN PATENT DOCUMENTS
CA 2105501 9/1993

OTHER PUBLICATIONS
Paul Clapis, Mark Cullen, Duncan Harris, David Jenkins, "Real-time Image Processing and Data Fusion of a Two-Channel Imaging Laser Radar Sensor" SPIE vol. 1633 Laser Radar VII (1992) pp. 281, 283, 285, 287.
www.lasermap.com/laserM/english/p3.asp Lasermap Image Plus. At least as early as Jun. 27, 2001.
Mosaic Mapping Systems Inc. LIDAR Remote Sensing Services. At least as early as Jun. 27, 2001.
www.saii-lla.com/photoorthoimage.html. Orthoimage Production. At least as early as Jun. 25, 2001.

Examiner(s)
Primary Examiner—Diego Gutierrez
Assistant Examiner—Madeline Gonzalez
(74) Attorney, Agent, or Firm—Clayton, Howarth & Eamon, P.C.

Abstract
(57) A 3D MultiSpectral Lidar. The system comprises a laser transmitter light source, a laser detection assembly, optics that couple the outgoing and incoming laser signals, a digital camera assembly for collecting passive light, a position and orientation system, and processing hardware. The system provides real-time georectified three dimensional images and topography using an airborne platform. The system collects time-synchronous lidar range and image data in an optical receiver. The individual images are then mosaiced and orthorectified in real-time. The lidar range data and image data are then coupled to a position and orientation system to transform the three dimensional range images to a single geographically referenced multispectral three dimensional image.

Drawings
83 Claims, 8 Drawing Sheets

referring to a set of criteria that an invention must satisfy for being accepted as eligible. That is to say, an invention must fulfill the following requirements: novelty, inventive step, and industrial applicability [3; 29]. Novelty demands the invention must be new i.e. not been known before. The “inventive step” criterion requires the invention not to be obvious to anyone knowledgeable in the field. Industrial applicability demands the invention can be used or reproduced in some industries.

3.1 Patent History

The earliest version of patents emerged in 1474 in the Venetian Republic, where people were granted exclusive rights for inventing or just bringing in the already existing technologies from outside. As it was created to encourage economic growth, it gradually became popular among other European rulers [30]. However, according to Khan et al. [31], its power was frequently abused by the monarchs by allowing monopolies on even common goods (e.g. salt) to increase revenue. Therefore, more restrictions had to be brought in to prevent those practices (e.g. limiting the scope of patent usage on only new inventions). In fact, the Statute of Monopolies Act was introduced in Britain in 1623 in which patent granting rights were taken away from the crown and given to the Parliament [30; 31].

The first article of the constitution of the United States involves a clause (Article I, Section 8, Clause 8¹) about patent rights. Khan et al. [31] argued that this clause set up the grounds for the very first modern patent system in which the main aim is “to promote learning, technology and commercial development as well as create a repository of information on prior art”.

3.2 Use of Patent Information

The patent data sets are extremely large and contain well-structured information about technologies. Information contained in patents in a technological domain is strongly correlated with the rate of technological progress in that domain [32; 33]. The rate of progress may indicate whether the technology will be successful or not in the future. Thus, analysis of patent information usually plays a major role in the R&D-related activities of most companies [34; 33]. Jürgens et al. [33] called this activity “technology watch” or “technology monitoring”. It is a branch of so-called “competitive intelligence” which refers to a set of research activities carried out by companies to keep themselves informed about the changes in their industries for adjusting their offerings and making knowledgeable decisions [33].

According to Daim et al. [34], the total number of patents over time is useful statistic since patent increase rate usually drops down when technology enters a maturity phase. They divided the patent growth life-cycle into three stages: early, fast-growing, and maturity. An early stage is where the number of patents is limited. This stage is followed by a fast-growing stage where the rate of patent increase is

¹To promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries

at its highest. The last stage is where the number of incoming patents levels out. These stages are illustrated on the following graph:

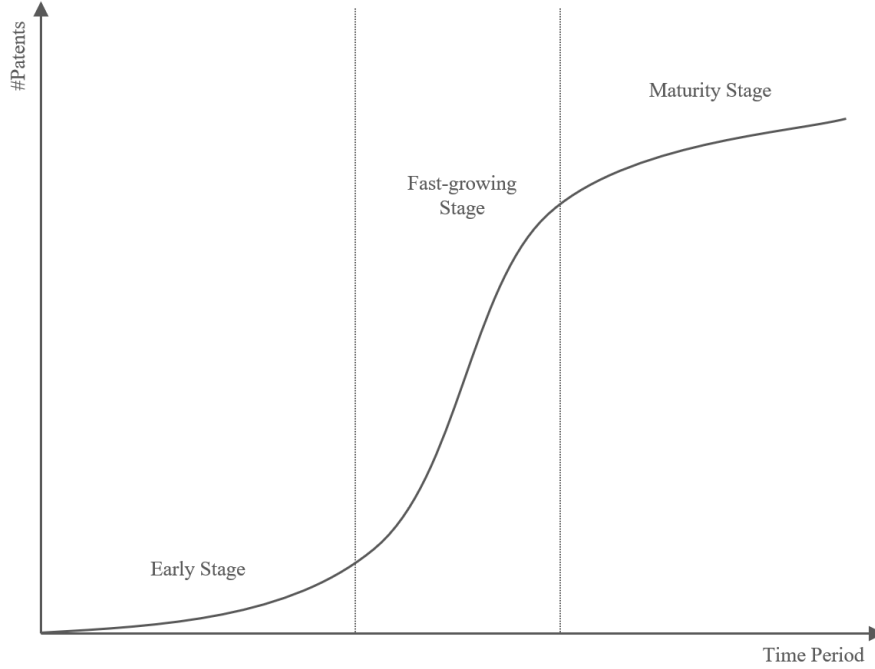


Figure 3.2: S-shaped Growth Curve for Technology Field

3.2.1 Limiting Factors

David et al. [3] mentioned that the protection granted by the patent organization is limited, usually 20 years. They have also written about a well-known phenomenon which is the existence of at least 18-month execution time required by the most patent offices to examine and publish the patent document. Petra Moser emphasized in her study [30] that some companies may decide not to apply for a patent for their valuable innovations due to several internal factors including a secrecy policy. This factor especially contributes to a risk of missing important inventions during the “technology watch” activities.

Jürgens et al. [33] mentioned that patenting activity significantly differs from one industry to another.

3.3 Patent Data

The Patent data used in this study was purchased from European Patent Office in late March 2018. Data came in the form of a relational database, namely PATSTAT, consisting of 29 tables. The following list shows all the tables available in the database [35]:

Table Name	Description
tls201_appln	<i>Patent applications</i>
tls202_appln_title	<i>Application titles</i>
tls203_appln_abstr	<i>Application abstracts</i>
tls204_appln_prior	<i>Paris Convention Priority</i>
tls205_tech_rel	<i>Technical Relation: 'priority-like' relation</i>
tls206_person	<i>Information about applicants and inventors</i>
tls207_pers_appln	<i>Mapping between person and application</i>
tls209_appln_ipc	<i>International Patent Classification</i>
tls210_appl_n_cls	<i>National classification</i>
tls211_pat_publn	<i>Patent publication</i>
tls212_citation	<i>Citations to publications and applications</i>
tls214_npl_publn	<i>Non patent literature publication</i>
tls215_citn_categ	<i>Citation category</i>
tls216_appln_contn	<i>Application continuation</i>
tls222_appln_jp_class	<i>Japanese classification</i>
tls223_appln_docus	<i>US classification</i>
tls224_appln_cpc	<i>Cooperative Patent Classification</i>
tls226_person_orig	<i>Unmodified person data</i>
tls227_pers_publn	<i>Mapping between person and publication</i>
tls228_docdb_fam_citn	<i>Citation between DOCDB families</i>
tls229_appln_nace2	<i>NACE2 industry classification</i>
tls230_appln_tech_n_field	<i>Classification by technical field</i>
tls231_inpadoc_legal_event	<i>Legal events related to application</i>
tls801_country	<i>Reference table of country codes</i>
tls803_legal_event_code	<i>Reference table of legal event codes</i>
tls901_tech_n_field_ipc	<i>Mapping between technology fields and IPC codes</i>
tls902_ipc_nace2	<i>Mapping between IPC codes and industry sectors</i>
tls904_nuts	<i>NUTS regional codes</i>

Table 3.1: Tables names in the PATSTAT database

The PATSTAT database is a combination of two databases maintained by European Patent Office: DOCDB worldwide bibliographic database and INPADOC worldwide legal event database. It contains data for approximately 92 million patent documents some dating back to 1782 which covers over 90 countries [35].

Despite the massive amount of information the PATSTAT database offers, we mainly focused on using textual information such as titles and abstracts. However, we discovered that around 42 million patent documents were missing abstract information, approximately 5 million abstracts are non-English, and a small subset of English abstracts are incomplete (e.g. “The invention relates to a display device comprising a flat display screen and to a guiding devi”). Thus, we applied a minimum threshold on abstract length of 150 characters (mean length is 884 characters) to eliminate as many incomplete and short abstracts as possible. This further cut about 250,000 abstracts. As a result, approximately about 45 million patents are found to be usable in this study.

4

Methods

At the beginning of this thesis, the intention was to use clustering algorithms in order to get representative data sets. However, as we were making progress with the practical work and not getting desired results, the method was shaped into a form which we named Similarity-Based Patent Selection (SBPS).

This chapter consists of three sections. The first section will describe the preprocessing techniques applied to the patent data. In the second section, the initially intended method will be briefly described. Finally, the last section will explain the proposed method, SBPS, in detail.

4.1 Preprocessing

4.1.1 Tokenization

Tokenization is the very first step to take when dealing with text data. It splits longer text into smaller pieces (e.g. sentences into words). It usually comprises a set of basic actions such as lowercasing, removing accents, punctuation marks, parentheses, etc. In this thesis, we employed the “simple_preprocess” function of the “Gensim” library to tokenize the data. This specific implementation allows defining minimum and maximum length for the tokens, which we set 2 and 50 respectively. It also gets rid of digits which is favorable in our case. The following example shows the result of applying the “simple_preprocess” function to a small part of a sample patent abstract:

Abstract (Raw)	After Tokenization
A rectangular-parallelepiped hinged lid type package for filter cigarettes comprising an inner pack (36) for cigarette pack, and a casing formed from a sheet of blank (50) and housing the inner pack (36).	rectangular parallelepiped hinged lid type package for filter cigarettes comprising an inner pack for cigarette pack and casing formed from sheet of blank and housing the inner pack

Table 4.1: Result of tokenization of raw abstract

4.1.2 Stopword removal

Stopwords are words that do not add much value to the general meaning of a text (e.g. a, an, the, for, etc.). Removing those words helps to reduce the dimensionality of documents, hence improves the efficiency of algorithms.

In this study, we used a vocabulary of stopwords in English language downloaded from a python library called “nltk”. Additionally, we implemented a logic to remove roman letters as well. The following demonstrates the result of the application of this technique:

Abstract (Tokenized)	After Stopword Removal
rectangular parallelepiped hinged lid type package <u>for</u> filter cigarettes comprising <u>an</u> inner pack <u>for</u> cigarette pack <u>and</u> casing formed <u>from</u> sheet of blank <u>and</u> housing <u>the</u> inner pack	rectangular parallelepiped hinged lid type package filter cigarettes comprising inner pack cigarette pack casing formed sheet blank housing inner pack

Table 4.2: Result of removing stopwords from tokenized abstract

4.1.3 Lemmatization

Lemmatization is the process of reducing inflected words back to their original form. For instance, the words “comprising” and “comprised” are the inflected forms of the word “comprise”.

Lemmatization is a useful technique to decrease the number of unique words in a vocabulary which is directly proportional to the efficiency and effectiveness of almost all of the algorithms in NLP.

In this thesis project, we utilized the “en_core_web_sm” model from a python library called “spacy” for lemmatization purposes. The result of applying this technique can be seen in the following table:

Abstract (Tokenized and Stopwords removed)	After Lemmatization
rectangular parallelepiped <u>hinged</u> lid type package filter <u>cigarettes</u> <u>comprising</u> inner pack cigarette pack <u>casing</u> <u>formed</u> sheet blank <u>housing</u> inner pack	rectangular parallelepiped hinge lid type package filter cigarette comprise inner pack cigarette pack case form sheet blank house inner pack

Table 4.3: Result of applying lemmatization on tokenized and stopwords-free abstract

4.1.4 Frequency-based word removal

Similar to stopwords, frequent words in the corpora usually do not add value to the meaning of individual documents. Besides increasing the dimensionality of documents, they usually hinder algorithms to catch the individual characteristics of documents by adding unnecessary complexity. Therefore, we implemented logic to get rid of the frequent words used in the entire corpora. Based on experimental observations, we found that a word used in at least 5% of all the documents should be regarded as a frequent word.

We also observed that the patent data set contains unignorable number of typographical errors which do not help the algorithms to work efficiently and effectively. To remove such words, we set a limiting requirement on words, to be present in at least 5 documents. We stick to the same example used in the above-mentioned preprocessing techniques in order to better illustrate the effect of removing words based on frequency. The result can be observed in the following table:

Abstract (Tokenized, Stop-words removed, Lemmatized)	After Frequent Word Removal
rectangular parallelepiped hinge lid type package filter cigarette <u>comprise</u> <u>inner</u> pack cigarette pack <u>case</u> <u>form</u> sheet blank house <u>inner</u> pack	rectangular parallelepiped hinge lid package filter cigarette pack cigarette pack sheet blank house pack

Table 4.4: Result of the removal of frequent words from tokenized, stopword-free, and lemmatized abstract

4.2 Clustering-Based Patent Selection

As mentioned above, the initial aim was to get the patent documents clustered into technology fields and then, look for the representative data sets among the clusters. To achieve it, the plan was divided into three major steps:

- The first step was to convert documents into vectors. We used the Doc2Vec model for this purpose since document embeddings obtained with TF-IDF vectorizer required more memory resources in the computer than there were available.
- The second step was to get the document embeddings clustered. According to the plan, K-Means was the first algorithm to try due to its simplicity and efficiency. Depending on acquiring promising results assessed with Silhouette and Calinski-Harabasz evaluation metrics, DBSCAN was yet another clustering algorithm to try.
- The final step was to run TF-IDF or TextRank keyword extracting algorithms on each cluster to find a technology field it represents. However, the execution of this step was highly dependent on the success in the second step.

As discussed in Section 5.2.1, the clusters obtained in the second step were inseparable from one another. This conclusion was suggested by the “Silhouette Score” metric (see Section 2.5.1) as the calculated average value was negative, close to 0. The potential cause leading to the overlapping clusters was the underperformance of the Doc2Vec model employed in the first step of this method.

4.3 Similarity-Based Patent Selection (SBPS)

SBPS is a method proposed in this study that incorporates several Natural Language Processing techniques in order to automate the selection of representative patent set based on an input query from a user. It is inspired by the way Word2Vec and Doc2Vec models from the “Gensim” library finds similar words and documents. The working principle of SBPS can be broken down into three main stage: interactive query building, calculating similarity scores in the patent database, and determining similarity threshold.

4.3.1 Query Building

Query to SBPS is a single vector representing the description of a technology field. It is provided by a user and the process of building this query relies on the algebraic properties of the word vectors described in Section 2.2.2. Thus, the essential part of this stage is to train the Word2Vec model in order to obtain vectors for the words used in the corpora.

There exist two word lists, namely *positive* and *negative*. Users are required to provide keywords in these list: words that they want the results being closely related to in the positive list; words that they want the results being unrelated to in the negative list. The query is a resulting vector obtained by adding word vectors in the positive list and subtracting the ones in the negative list.

The query building process is assisted with the utilization of “most_similar” function of “Gensim” Word2Vec. As soon as a word is added to either of the lists, SBPS prepares the current query and presents top n most similar words in the corpora. This feedback feature of the system hints about the possible results and informs the users of the associations that the query makes. Then, the users either remove the unwanted associations by adding those words to the negative list or keep extending positive list until they confirm the query.

4.3.2 Calculating Similarity Scores

Once the query is confirmed, SBPS, similar to “Gensim” Doc2Vec, iterates over all the documents in the data set to calculate cosine similarity scores. Unlike Doc2Vec, document vectors are calculated the same way the query vector gets calculated, that is, a document vector is a total sum of the word vectors for n keywords in the corresponding document.

In this study, the TextRank algorithm is utilized for extracting keywords out of the documents. The choice of keyword extraction algorithm is based on experimental

observations as the TextRank algorithm is found to be better in summarizing patent documents than the TF-IDF algorithm.

4.3.3 Determining Similarity Threshold

Determining the similarity threshold is a manual stage that requires user's interaction since all documents are assigned with a similarity score in the second stage. The manual fashion is mainly because the query building stage is open-ended, that is, the user defines the scope of the query or the desired technology field; the narrower the scope it gets, the lower the threshold is required.

5

Results

5.1 Verification via Single Technology Sector

In order to quickly check if Doc2Vec algorithm produces good results, we limited the number of patents by focusing on a single technology sector. We selected all patent abstracts categorized under the chemistry technology sector since it was the largest sector in the database. This accounted for about 10.5 million abstracts after the application of the initial filters for excluding short, incomplete, and non-English patent abstracts. We divided the data set into training and test subsets comprising of 80% and 20% of the whole data respectively. Both datasets were subjected to the tokenization and stop-word removal processes before feeding the training dataset in to the Doc2Vec model with the default settings. Since there was not any labeled test data, we carried out a manual evaluation test on the acquired document embeddings based on our intuition. For this task, we randomly picked a small subset of patent abstracts out of the test data set and asked the trained Doc2Vec model to bring back the most similar 20 abstracts for each selected abstract. We analyzed the results and found the traces of similarities between the queried and returned abstracts. However, we could not draw a concrete conclusion about the quality of document embeddings since this test required a substantial knowledge in chemistry.

Besides the document embeddings, the Doc2Vec model produced word embeddings on which a similar test is applied. We randomly selected 10 words from the vocabulary and asked the model to return 10 most similar words for each of them. The result showed that the model trained the word vectors well as it was able to capture the semantic meaning of words (e.g. for the word 'aluminium' the returned results are mostly related to metals such as 'metallurgy', 'aurum', 'argentum', 'fer-rum', 'natrium' etc.). This result yielded that the Doc2Vec algorithm is capable of producing quality document embeddings out of the patent abstracts. Thus, we expanded the scope of the experiment by considering patents from all technology sectors available.

5.2 Analysis on Multiple Technology Sectors

The data set considered in this section contains information for all the relevant patents (about 44.75 million) in the database. Unlike the data used in Section 5.1, we used both patent titles and abstracts merged together.

We stuck to the same rule (80% training and 20% test) for dividing the data set. Therefore, we were left with 36 million data points to train the model on. Due to the limitations on the resources of the used computer, a single run of the Doc2Vec model was taking about 8 to 9 hours. Spending this amount of time on a single run was not feasible by taking into account the number of parameters to tune. This is why, we decreased the amount of training dataset by randomly sampling about 5 million data points.

As mentioned in Section 5.1, the main obstacle was having no labeled data set to be used for performance evaluation of the trained models and it was not feasible to manually create one from both time and domain knowledge perspectives. The reason for it is that patents are usually prepared by owners in a way to touch as many different fields as possible, which makes the manual categorization task significantly complicated.

5.2.1 Failed Experiments

In this subsection, we will present the results of the experiments that did not work.

5.2.1.1 Parameter Tuning through Sanity Checking

We decided to use “sanity check” described in Section 4.2 as a method to do a sort of model evaluation and tune hyper-parameters of Doc2Vec. For this purpose, we randomly selected 10,000 data points from the training data set. We trained numerous models with different parameter settings. The following table shows the considered values for each parameter (all possible combinations):

Vector size	Min. Count	Traning Algorithm	Window	Epochs
100	10	"dm"	10	30
—	20	"dbow"	20	45
—	—	—	—	60
—	—	—	—	100

We carried out “sanity check” on all trained models several times and averaged the results. Here are our findings:

- The 'DBOW' algorithm outperforms the 'PV-DM' algorithm as the accuracy was around 99% and 82% respectively.
- The accuracies for the models trained with 30 epochs were under the 95% threshold, averaged around 92%.
- Inconsistent accuracy results for the rest of the parameter values.

Overall the evaluation attempt through “sanity check” did not contribute to any information gain as the results were unclear and inconsistent with the parameter values. This confirmed that “sanity check” should not be used as a technique to measure the accuracy of models.

5.2.1.2 Parameter Tuning through Cluster Evaluation

Since “sanity check” did not help to assess the quality of the document embeddings, we came up with an idea of evaluating document embeddings and clusters altogether using intrinsic cluster evaluation techniques. The assumption was that quality clusters can only be produced if document embeddings are well-trained. We chose K-Means as a clustering algorithm since it is powerful enough and requires only one parameter which is the number of clusters. Since that parameter is unknown, we ran K-Means several times on the same set of document embeddings with different choices of K value, ranging from 75 to 175 with 25 difference between the runs. As an evaluation method, we decided to employ “Silhouette Score” since its value range is well defined and it will enable us to easily compare models to each other. The aim to find near-optimal parameter settings for both Doc2Vec and KMeans through maximizing the silhouette score. The observed silhouette scores were all negative close to 0 and it did not improve with any choice of K value. According to the Section 2.5.1, a negative silhouette score suggests that clusters are not separable and they have significant overlaps. In our case, these results translate to the document vectors being closely positioned to each other in the subspace. It may be due to having many common words in patent title and abstracts which in return, contributing to the document vectors being more alike. In order to decrease similarities and put more emphasis on individual differences of the documents, we applied an extra preprocessing technique to get rid of the frequent words across the patent titles and abstracts. We trained the models with the same parameter settings and clustered the acquired document vectors with the same choices of K . However, we did not observe any significant improvement in the silhouette scores compared to those in the previous experiment.

5.2.2 Successful Method: SBPS

This section will first give brief information about how the SBPS method has been implemented and utilized in this study. Then, it will continue to present the graphs (e.g. growth curve described in the Section 3.2) extracted for several competing technologies using the SBPS tool. Each technology battle will be discussed in a separate subsection where the results will be analyzed in comparison to the history of the corresponding technologies.

5.2.2.1 SBPS as Command Line Application

In this study, the SBPS method was implemented in the Python programming language as a command-line application.

All the preprocessing techniques mentioned in Section 4.1 were applied to the data set (comprising of about 45 million patent abstracts and titles) and a Word2Vec model with 500 vector size was trained. The model was then used in the query building stage to assist users with the 10 most similar words as they update the “positive” and “negative” lists. The following figure shows an example of building a query for radar technology:

```

> python main.py --name example
COMMANDS> p (+|-) <word>: {alter positive list} |      n (+|-) <word>: {alter negative list} |      f: {finish query}

-----
positive=[], negative=[]

CMD> p+radar
-----
positive=['radar'], negative=[]

Most similar words: ['sonar', 'doppler', 'clutter', 'lidar', 'azimuth', 'echo', 'rader', 'sar', 'satellite', 'monopulse']

CMD> p+radiowave
-----
positive=['radar', 'radiowave'], negative=[]

Most similar words: ['sonar', 'doppler', 'rader', 'radio', 'azimuth', 'antenna', 'satellite', 'monopulse', 'obstacle', 'radiolocation']

CMD> n+ultrasound
-----
positive=['radar', 'radiowave'], negative=['ultrasound']

Most similar words: ['rader', 'radiolocation', 'radio', 'azimuth', 'satellite', 'antenna', 'obstacle', 'clutter', 'monopulse', 'roadside']

```

Figure 5.1: Example of Query Building Process

As can be seen in the figure, the tool found the words “radar”, “sonar” and “lidar” related. It is indeed factually accurate since they all are sensor technologies. It has also been shown in the example that the introduction of “radiowave” in the positive list and “ultrasound” in the negative list respectively made the words “lidar” and “sonar” less related. These results are expected since LIDAR and SONAR are the technologies based on light and sound respectively while the RADAR technology makes use of radio waves.

As described in Section 4.3, the TextRank algorithm was employed to extract at most 15 keywords from the documents and save them in a csv¹ formatted file. The tool was developed in a way to take the keywords file as an input and, then obtain word vectors for the keywords by utilizing the trained Word2Vec model and finally, add them together to form the document vectors.

Once the query is passed by the user, the tool utilizes the “cosine_similarity” function from the well-known python library called “Sklearn” to calculate the similarity scores between the query vector and the document vectors and saves the result in a file.

In sum, the SBPS tool used in this thesis produces a file containing similarity scores mapped to patent applications, the histogram or growth curve graphs for technologies.

5.2.2.2 Carburetor vs. Fuel Injection

Carburetor and Fuel Injection are two competing technologies used in vehicles that serve the same purpose, delivering fuel to engines. These technologies are responsible for mixing air with fuel in an appropriate ratio before the delivery. The way they achieve this aim is what makes them different. The carburetor is simpler in design compared to fuel injection systems. It has two mechanical valves that control air and fuel flow into the combustion engine while most of the fuel injection systems are electronically controlled which means the computer decides the amount of fuel delivery based on external factors.

¹Comma Separated Values

Both the carburetor and fuel injection have been around since the 19th century, but the carburetor is the older technology. Fuel injection was being mostly used in diesel engines until the early 1950s. Its usage gradually increased after the first introduction for petrol engines for passenger cars and, finally, it became the preferred technology over carburetors to be used in the automotive industry due to its efficiency and effectiveness. The shift from carburetors to fuel injection systems took place in the late 1980s. This event can be clearly observed in the figures produced by the SBPS tool. The Figures 5.2 and 5.3 show that the growth of the carburetor technology started to plateau in the mid-1980s while the fuel injection was on the rise. Figure 5.3 also shows that there was a slight increase in the number of filed patent applications in the fuel injection technology around the 1950s, which is in correspondence with the historical events.

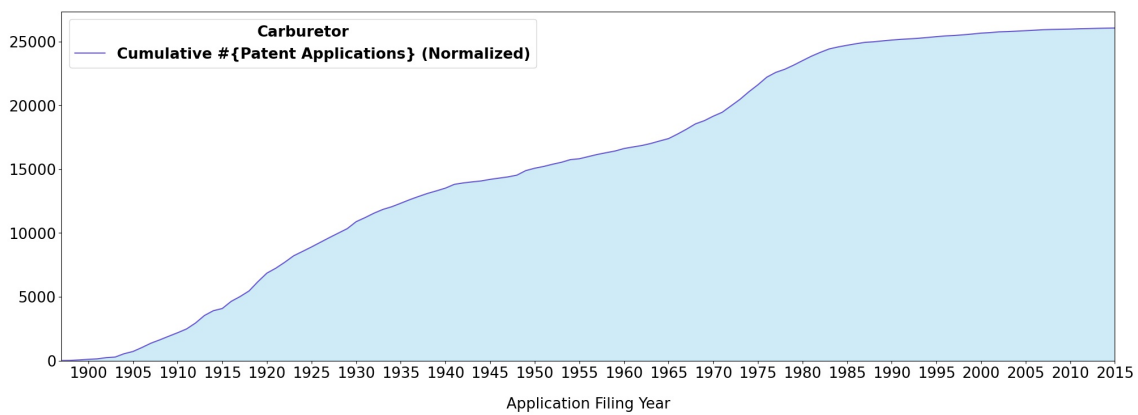


Figure 5.2: Growth Curve of Carburetor Technology

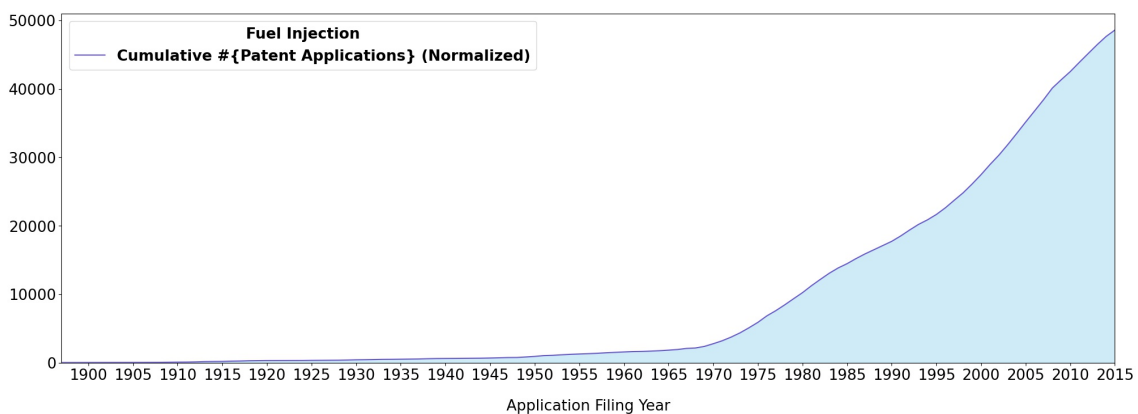


Figure 5.3: Growth Curve of Fuel Injection Technology

5.2.2.3 Floppy Disk vs. Optical Disc vs. USB Flash Drive

Floppy disk and USB flash drive belong to the family of data storage technologies. The first floppy disk was introduced in the early 1970s by IBM. Due to its well-known issues including large dimensions (8-inch) and little capacity (about 80KB), its new versions were introduced in 1976 (5,25-inch) and 1982 (3,5-inch) and became

more popular. Floppy disk remained the most popular storage technology until it was slowly replaced by optical disc technology (e.g. CD/DVD). The optical disc was invented in the 1960s however the earliest version, namely “LaserDisc”, was introduced around the early 1970s and become commercially available by the end of the decade. However, it did not gain popularity as it was not fully digital and its retail price was high. This technology attracted more attention with the introduction of CD and DVD in 1985 and 1995 respectively and became mainstream technology to be used in computers. Between 2000 and 2005, this technology became even more popular with the introduction of Blu-ray discs. However, it was made obsolete by the advancements in flash memory technology (e.g. USB Flash Drive). The foundation of flash memory technology was laid down around 1980 by Toshiba. The increase in reliability and decrease in the cost per capacity ratio of this technology over time led to the rise in its popularity. In 1999, the first patent was filed for the USB Flash Drive technology. Towards the end of the 2000s, it became the preferred technology over its competitors.

The following graphs (see Figure 5.4, Figure 5.5, and Figure 5.6) have been extracted with the SBPS tool. The historical events discussed in the paragraph above can be clearly observed in these graphs. What may come as a bit of a surprise in Figure 5.6 is that the USB Flash Drive technology seems to be entering its maturity stage. In this case, the relevant question might be if the recent progress in cloud technology puts an end to the life of the USB Flash Drive technology.

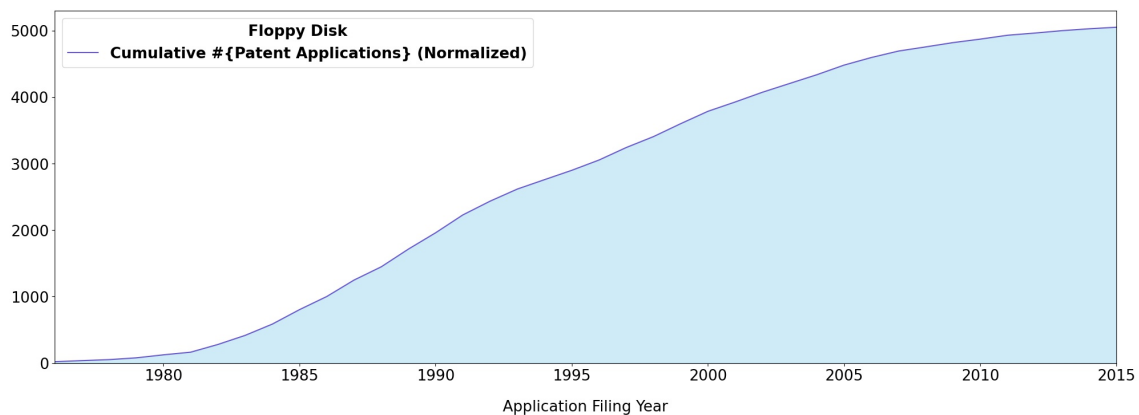


Figure 5.4: Growth Curve of Floppy Disk Technology

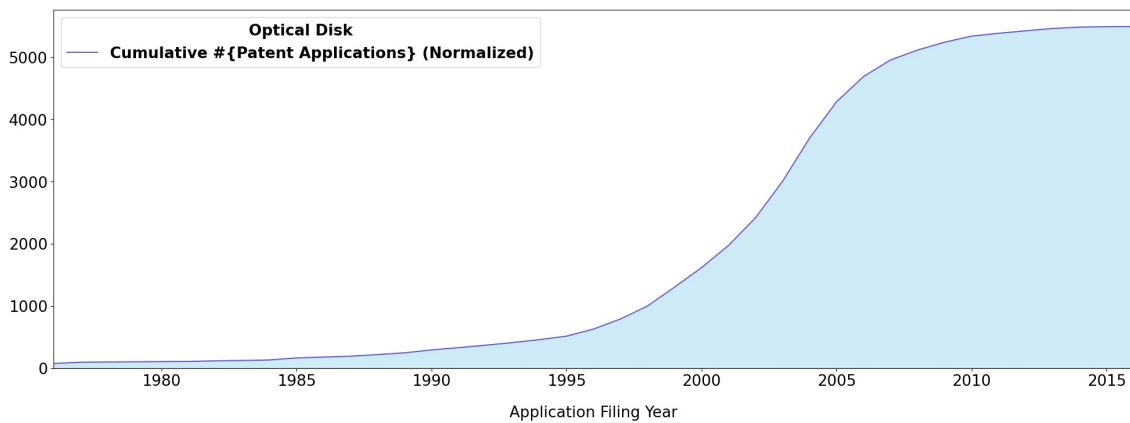


Figure 5.5: Growth Curve of Optical Disc Technology

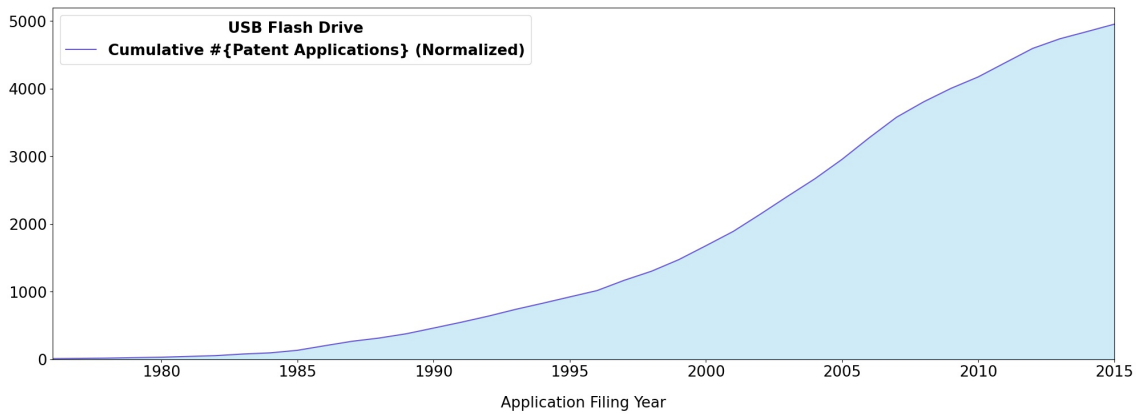


Figure 5.6: Growth Curve of USB Flash Drive Technology

5.2.2.4 Fluorescent vs. LED

Fluorescent and LED are two well-known lighting technologies. Despite the former has a much longer development history than the latter, the battle between these two technologies is especially interesting since the advancements in the LED technology have led to a decrease in the interest in fluorescent technology. Fluorescent technology dates back to 1857. It is when the French physicist Alexandre E. Becquerel investigated the topic for the first time and formed a theory of the fluorescent tube [36]. There were several experiments on building similar lamps and some patenting activities took place between the end of the 19th century and 1930 but the invented lamps were not practical enough from many technical aspects in order to gain popularity [36]. According to Bellis [36], the first practical fluorescent lamp was invented in 1934 and sold in 1938. In 1980, Philips brought the first compact screw-in fluorescent lamp to the market which offered several benefits over its competitors [37]. According to Bell [37], this invention led to further advancements in the technology. On the other hand, LED has a shorter history which dates only back to the 1960s. The first light-emitting diode (red light) was invented in 1962 by Nick Holonyak [38; 39]. Holonyak et al. [39] called this invention an industry revolutionizing invention since it was the basis for the modern LED lamps. The first commercially

LED was introduced in 1968 [40]. After that, there were several LEDs developed producing monochromatic light between 1980 and early 1990s [41]. According to Cho et al. [41], these inventions created the grounds for the invention of white LED. In fact, the first white LED lamp was announced two years after the invention of high-brightness blue LED in 1994 [42; 41]. Thereafter, this technology has seen great attention by the companies in the lighting industry and made them shift from investing in other lighting technologies to the LED technology. As a matter of fact, Philips announced to cut off their investments in Fluorescent bulbs in favor of the research of the LED technology in 2008 [43].

Figure 5.7 shows that there was a slight increase in the patenting activity for fluorescent technology between 1930 and 1935. This increase is expected since the first practical fluorescent lamp was built during that period. The growth curve for the fluorescent technology also shows that the fast-growing stage began after 1975 which can be explained by the potential development period of the first compact fluorescent bulb introduced by Philips in 1980. In the same figure, inclination after 2011 is observed which may indicate the beginning of the maturity stage (see also the histogram graph in Figure 5.9). Entering the maturity stage is expected, given the success of the LED technology during the period. As can be seen in Figure 5.8, the SBPS tool managed to get the beginning of the technology somewhat right and capture the rise in interest to the LED technology after 2000; however, we could not find an answer to the decline in the patenting activity in the LED technology observed after 2012 (see Figure 5.10). As we were having a hard time explaining all the trends shown on the figures, an expert interpretation might be valuable in this technology battle.

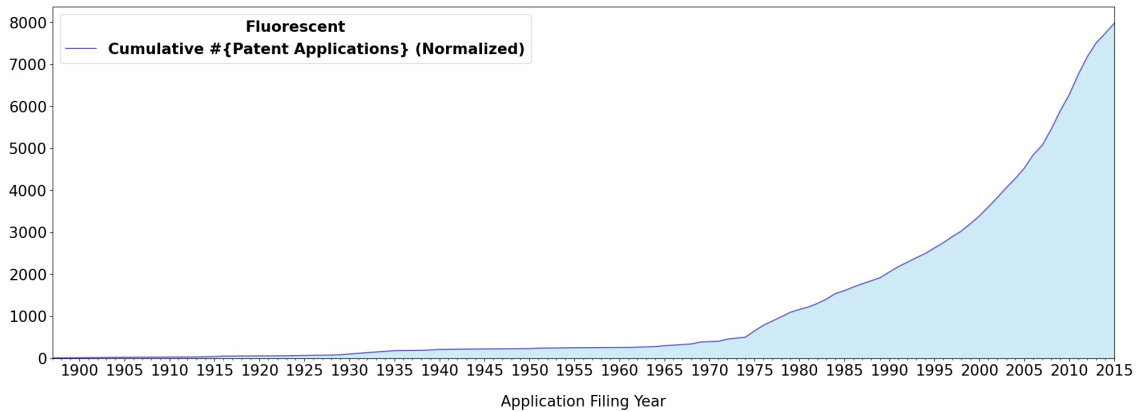


Figure 5.7: Growth Curve of Fluorescent Technology

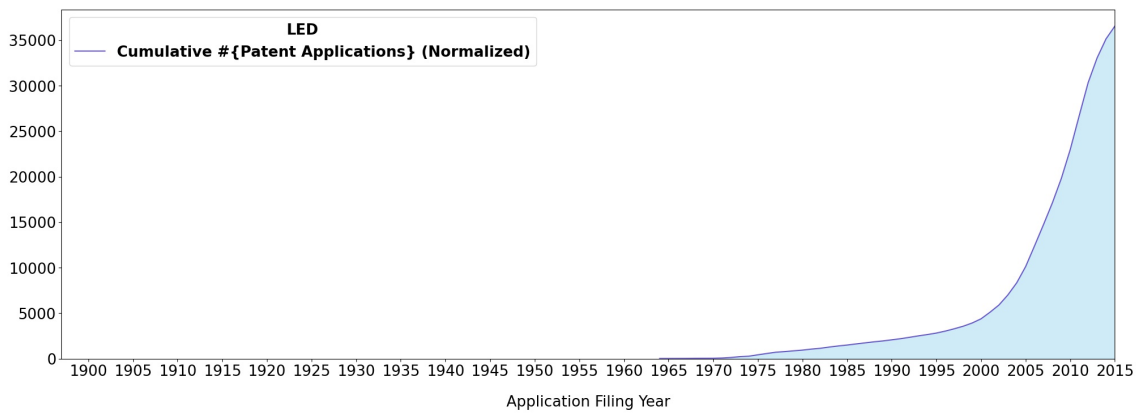


Figure 5.8: Growth Curve of LED Technology

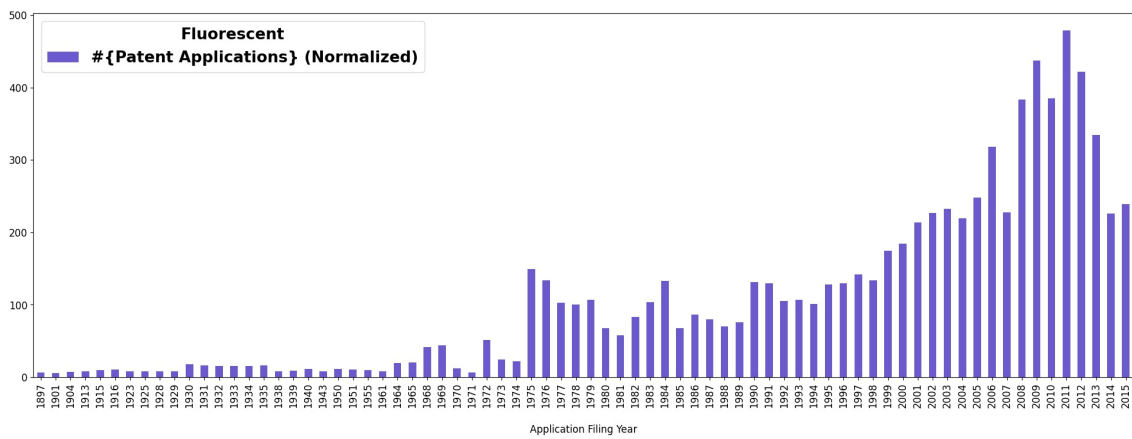


Figure 5.9: Histogram for Fluorescent Technology

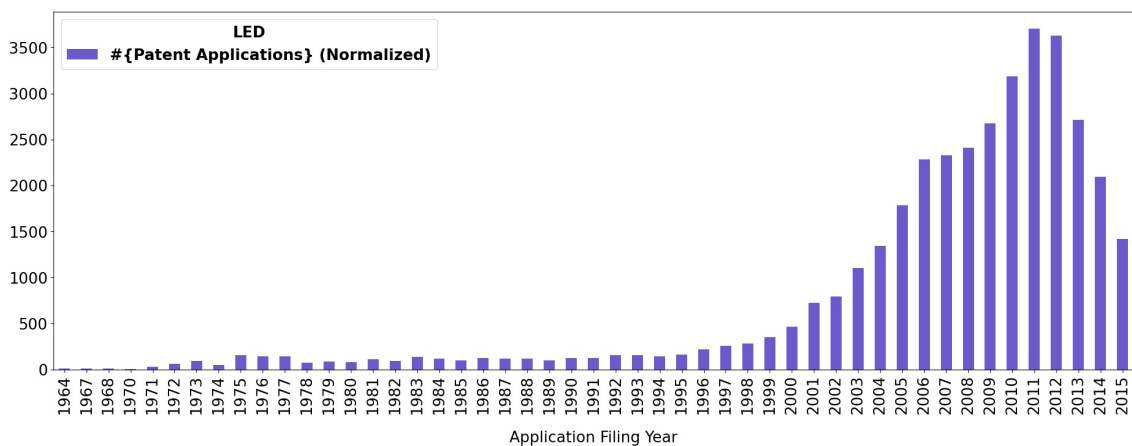


Figure 5.10: Histogram for LED Technology

5.2.2.5 SONAR vs. RADAR vs. LIDAR

Sonar, radar, and lidar are similar technologies that rely on the reflection of sound, radio, and light waves respectively. These technologies are developed to locate distant objects. The oldest one of these technologies is sonar, invented in 1906 for

detecting icebergs [3; 44]. According to Ainslie [45], the world's first patent for an echolocation system similar to sonar was filed in 1912. The sonar technology was further developed during World War I in order to detect submarines [3; 44]. In fact, the first functional sonar was built in 1918 [45]. The technology was even investigated and developed further during World War II [45]. David et al. [3] stated that sonar technology has been widely used in the automotive industry due to its accuracy in short distances and reliability in different conditions. According to Ainslie [45], advancements in digital computers created new possibilities and stimulated the development of sonar technology. However, David et al. [3] have stated that sonar is reaching to the end of its development life cycle. Radar, on the other hand, emerged just before the beginning of World War II and developed during the war for being used in military applications [3; 46]. Besides the usage in the military, it has been successfully utilized for numerous civilian applications such as air traffic control, speed measurement, planetary observations, and so on. According to David et al. [3], radar has grown to become one of the key technologies in the automotive industry and it is predicted to grow even further due to its potential for enabling autonomous driving. Finally, lidar is the youngest member of this group which was developed shortly after the invention of laser in 1960 [47; 48]. However, the term, lidar, was used for the first time in 1953 for a similar measurement technique according to Wandinger [47]. The research in lidar technology continued to rise, but the invention of GPS technology, happened in the 1970s [49], made the technology less popular [47]. At the beginning of the 21st century, interest in lidar technology has re-arisen due to its great potential for enabling fully autonomous driving [3].

By looking at the growth curves produced by the SBPS tool (see Figure 5.12, Figure 5.13, and Figure 5.14), it can be clearly seen that the SBPS method could be able to capture most of the main events for all these three technologies (e.g. approximate invention time, fast-growing period). However, the graphs for Sonar and Radar technologies do not feature much patenting activity during WWI and WWII respectively. The reason probably is due to the fact that the PATSTAT database misses abstract information for most patents filed during the first half of the 20th century (more than 85 percent).

Interesting results came out when we compared the graphs from the SBPS tool with the graphs that David et al. [3] acquired for their manually selected data sets. The authors concluded that all these three technologies are in their fast-growing stage (see Figure 5.11) which agrees with what our graphs (see Figure 5.12, Figure 5.13, and Figure 5.14) display. The relative rises and falls in Figure 5.11 reflect the relative fluctuations in the figures acquired by the SBPS tool. These comparative results confirm even further the strength of the employed method.

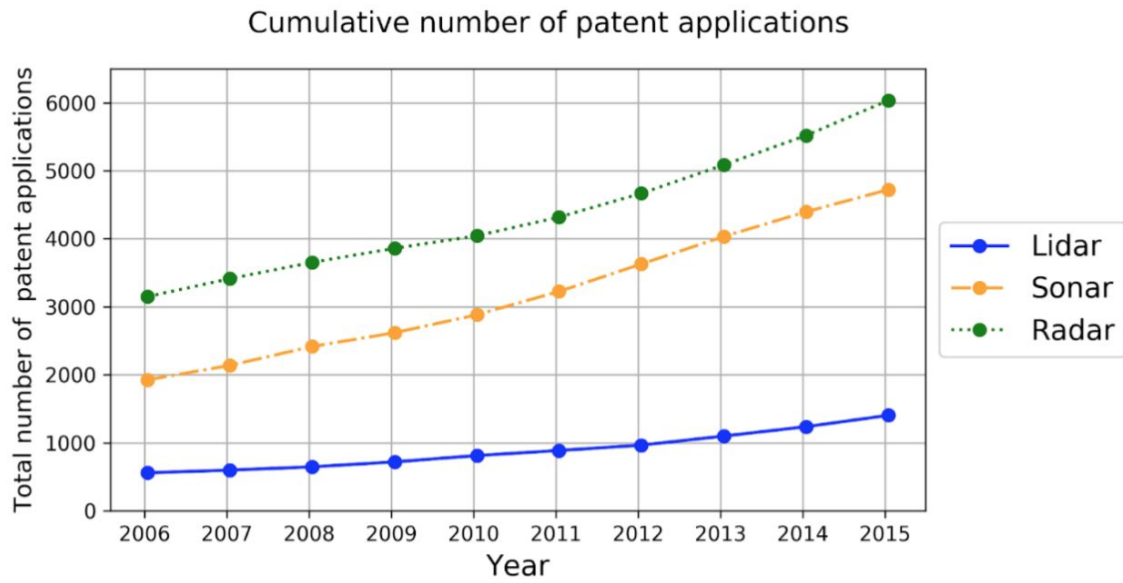


Figure 5.11: Cumulative number of patent applications [3]

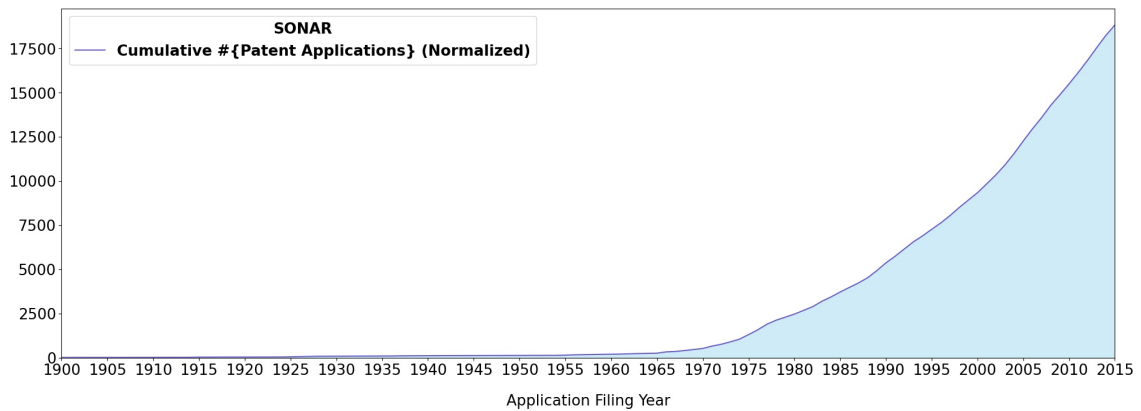


Figure 5.12: Growth Curve of SONAR Technology

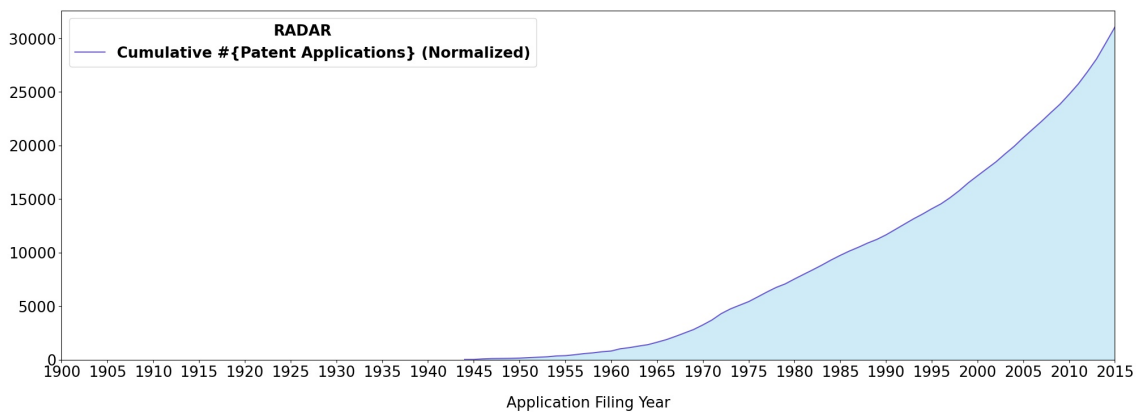


Figure 5.13: Growth Curve of RADAR Technology

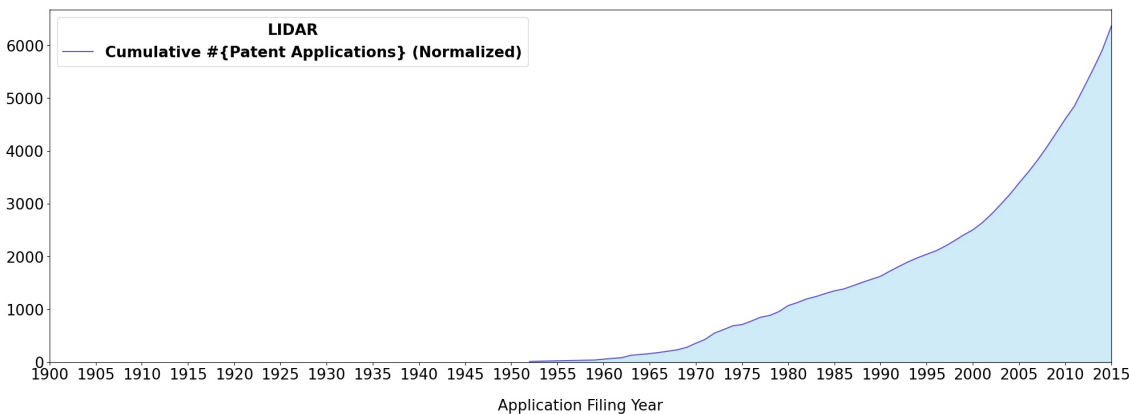


Figure 5.14: Growth Curve of LIDAR Technology

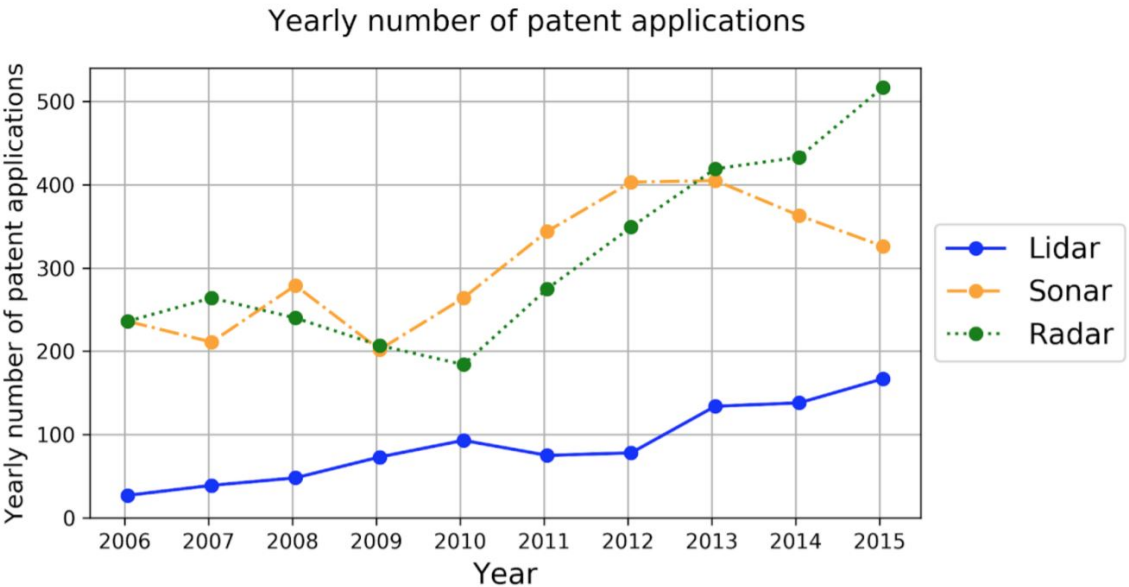


Figure 5.15: Yearly number of patent applications [3]

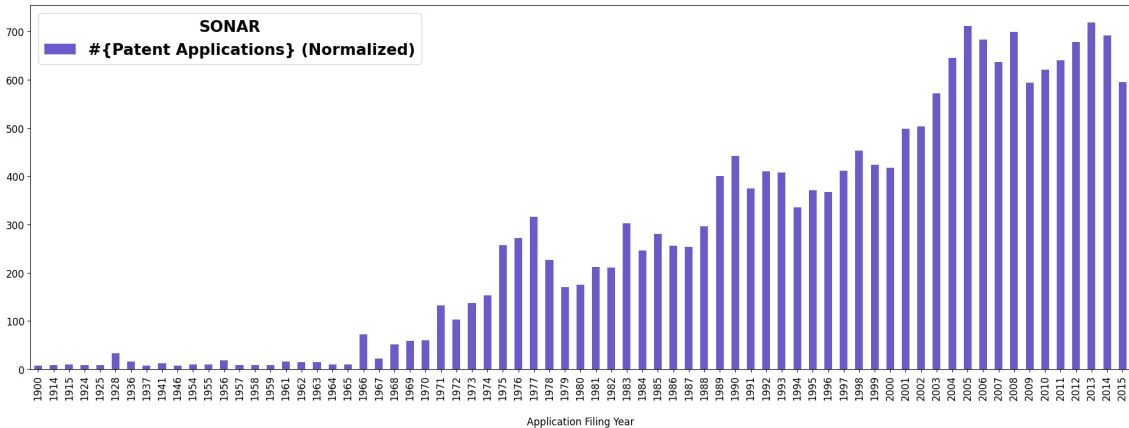


Figure 5.16: Histogram for Sonar Technology

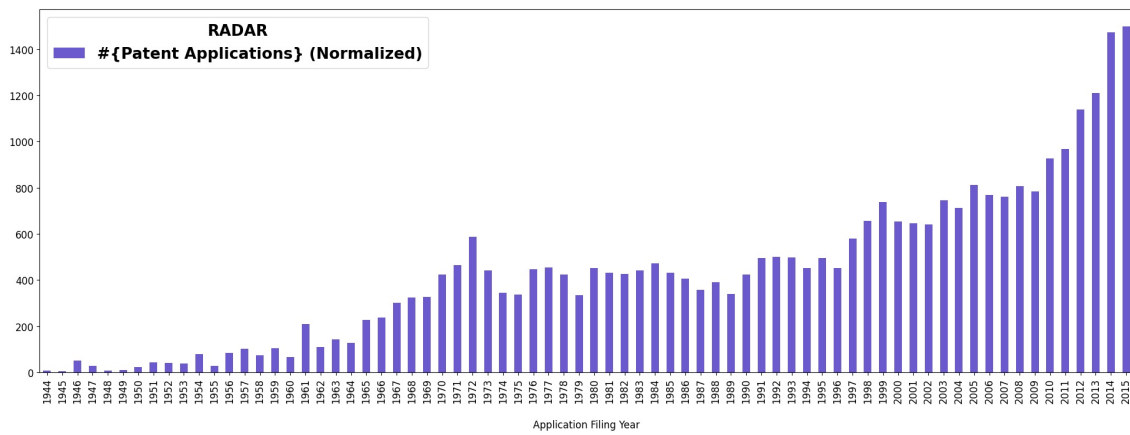


Figure 5.17: Histogram for Radar Technology

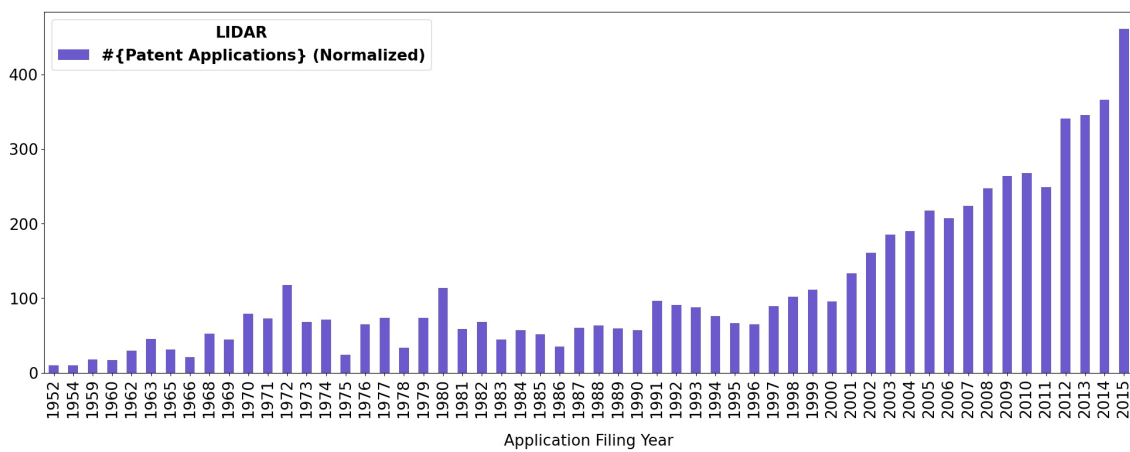


Figure 5.18: Histogram for Lidar Technology

6

Discussion

The primary aim of this thesis project was to develop a method to automate the most tedious but yet important task of the patent analysis process which is retrieving representative data sets. Therefore, the following main research question was set: “*How can unstructured textual information found in a patent database like PATSTAT be used to construct a patent selection system?*”. To answer this question, two sub-questions were asked:

1. *What unstructured textual information can be found in the PATSTAT database and which ones can be used for our purpose?*
2. *Which algorithms can be utilized to build a patent selection system? What value do they add?*

A patent document is similar in structure to an academic paper, consisting of title, abstract, description, and inventors. Additionally, it contains claims, assignee, and examiners information. Among these information pieces, only four were found to be useful in directly revealing information about the technology sector that a patent belongs to. We could access only the title and abstract since claims and descriptions were not available in the database. It was also found that the PATSTAT database provides abstracts for about 56% of all patents and 16% of the ones filed before 1980. It means we may miss most of the historical patenting activities in the results.

The nature of the source data restricted us to choose from Natural Language Processing algorithms. It included several sets of techniques to be considered: pre-processing, text vectorization, and keyword extraction. From the first set, “lemmatization” is found to be the most important technique since words in abstracts are often used in their different inflected forms. From the second set, we found word embeddings to be more flexible and useful than document embeddings. It may be affected by not having a sophisticated method to assess the quality of techniques providing document embeddings. Overall, the use of keyword extraction techniques and the cosine similarity measure in the SBPS method were the key factors in reaching positive results, given in Section 5.2.2.

The SBPS method proposed in this study is not limited to be used only on the patent data. It has the potential to be effective on any text data comprised of the data points with at least a few descriptive sentences such as academic papers, traffic accident descriptions [2], encyclopedia articles (e.g. Wikipedia), and so on.

6.1 Future Work

Recommendations for future scholars who would like to conduct research on this topic:

- Utilizing a database that offers more textual data for the patent documents than PATSTAT.
- Having access to a powerful computer that has at least 160GB RAM, 12 physical CPUs, GPU enabled, and at least 2TB SSD storage.
- Preferably using C programming language since most of the Python libraries suffer from an infamous limitation known as “Global Interpreter Lock”
- Keywords in the abstracts tend to be misspelled frequently and this affects some word vectors being under-trained. Using a method (such as “TextBlob” in Python) to fix typographical errors may increase the accuracy noticeably.
- The third stage of the SBPS method requires rather manual work and there is room for automation. It has been observed that there is an edge in the similarity range where the values above it account for only selecting relevant patents. For the values below that edge, the patent count increases dramatically.

Bibliography

- [1] J. Tang, J. Liu, M. Zhang, and Q. Mei, “Visualizing large-scale and high-dimensional data,” in *WWW '16: Proceedings of the 25th International Conference on World Wide Web*, pp. 287—297, April 2016.
- [2] S. Abdullayev and A. Mekonnen, “Topic modeling and clustering for analysis of road traffic accidents,” Chalmers University of Technology / Department of Applied Mechanics, 2017.
- [3] G. David and Y. Lili, “Using patent data to position firms in technology-intensive environments,” Chalmers University of Technology / Department of Technology Management and Economics, 2018.
- [4] A. Trippe, “Patinformatics: Tasks to tools,” vol. 25, pp. 211–221, 2003.
- [5] S. Bashir and A. Rauber, “Improving retrievability of patents in prior-art search,” in *Advances in Information Retrieval*.
- [6] P. Mahdabi and F. Crestani, “Learning-based pseudo-relevance feedback for patent retrieval,” in *Multidisciplinary Information Retrieval*, pp. 1–11, Springer Berlin Heidelberg, 2012.
- [7] W. Magdy and G. J. F. Jones, “A study on query expansion methods for patent retrieval,” in *In Proceedings of the 4th workshop on Patent information retrieval*, pp. 19–24, 2011.
- [8] W. Tannebaum and A. Rauber, “Using query logs of uspto patent examiners for automatic query expansion in patent searching,” vol. 17, pp. 451–470, 2014.
- [9] W. Tannebaum and A. Rauber, “Patnet: A lexical database for the patent domain,” in *Advances in Information Retrieval*.
- [10] L. Zhang, Z. Liu, L. Li, C. Shen, and T. Li, “Patsearch: an integrated framework for patentability retrieval,” vol. 57, pp. 135–158, 2018.
- [11] M. Verma and V. Varma, “Applying key phrase extraction to aid invalidity search,” in *Proceedings of the 13th International Conference on Artificial Intelligence and Law*, p. 249–255, Association for Computing Machinery, 2011.
- [12] J. Singh and A. Sharan, “Relevance feedback-based query expansion model using ranks combining and word2vec approach,” vol. 62, pp. 591–604, Taylor Francis, 2016.

- [13] L. Helmers, F. Horn, F. Biegler, T. Oppermann, and K.-R. Müller, “Automating the search for a patent’s prior art with a full text similarity search,” vol. 14, pp. 1–17, Public Library of Science, 2019.
- [14] R. Mihalcea and P. Tarau, “Textrank: Bringing order into texts,” in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, 2004.
- [15] S. Brin and L. Page, “The anatomy of a large-scale hypertextual web search engine,” in *Proceedings of the Seventh International World Wide Web Conference*, vol. 30, pp. 107–117, 1998.
- [16] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” arXiv:1301.3781, 2013.
- [17] T. Mikolov, W. tau Yih, and G. Zweig, “Linguistic regularities in continuous space word representations,” NAACL HLT, 2013.
- [18] Q. Le and T. Mikolov, “Distributed representations of sentences and documents,” in *Proceedings of the 31st International Conference on Machine Learning*, vol. 32, pp. 1188–1196, PMLR, 22–24 Jun 2014.
- [19] L. van der Maaten, E. Postma, and J. van den Herik, “Dimensionality reduction: A comparative review,” Tilburg University, 2009. Online print.
- [20] G. Linderman and S. Stefan, “Clustering with t-sne, provably,” vol. 1, pp. 313–332, 2019.
- [21] M. Dash, H. Liu, and X. Xu, “‘1+1>2’: merging distance and density based clustering,” in *Proceedings Seventh International Conference on Database Systems for Advanced Applications. DASFAA 2001*, pp. 32–39, 2001.
- [22] M. E. Celebi, H. A. Kingravi, and P. A. Vela, “A comparative study of efficient initialization methods for the k-means clustering algorithm,” vol. 40, pp. 200–210, 2013.
- [23] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *In Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD’96)*, 1996.
- [24] P. J. Rousseeuw, “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis,” vol. 20, pp. 53–65, 1987.
- [25] L. Lovmar, A. Ahlford, M. Jonsson, and A.-C. Syvänen, “Silhouette scores for assessment of snp genotype clusters,” in *BMC Genomics*, vol. 6, 2005.
- [26] K. R. Shahapure and C. Nicholas, “Cluster quality analysis using silhouette score,” in *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 747–748, 2020.
- [27] T. Calinski and J. Harabasz, “A dendrite method for cluster analysis,” vol. 3, pp. 1–27, 1974.

-
- [28] X. Wang and Y. Xu, “An improved index for clustering validation based on silhouette index and calinski-harabasz index,” vol. 569, p. 052024, IOP Publishing, aug 2019.
- [29] S. I. P. Office, “Conditions for a patent,” 2020. Available at: <https://www.prv.se/en/patents/applying-for-a-patent/before-the-application/conditions-for-a-patent> [Accessed April 3, 2021].
- [30] P. Moser, “Patents and innovation: evidence from economic history,” vol. 27, pp. 23–44, 2013.
- [31] B. Z. Khan and K. L. Sokoloff, “History lessons: The early development of intellectual property institutions in the united states,” vol. 15, pp. 233–246, 2001.
- [32] C. L. Benson and C. L. Magee, “Quantitative determination of technological improvement from patent data,” PLoS ONE 10(4), 2015. e0121635.
- [33] B. Jürgens and V. Herrero-Solana, “Patent bibliometrics and its use for technology watch,” vol. 7, pp. 17–26, 2017.
- [34] T. U. Daim, G. Rueda, H. Martin, and P. Gerdtsri, “Forecasting emerging technologies: Use of bibliometrics and patent analysis,” vol. 73, pp. 981–1012, 2006. Tech Mining: Exploiting Science and Technology Information Resources.
- [35] E. P. Office, “Data catalog – patstat global – 2020 autumn edition,” 2020. Available at: [http://documents.epo.org/projects/babylon/eponot.nsf/0/D3AECDEF45825DC7C1258614004FC6D9/\\$File/DataCatalog_Global_v5.16.pdf](http://documents.epo.org/projects/babylon/eponot.nsf/0/D3AECDEF45825DC7C1258614004FC6D9/$File/DataCatalog_Global_v5.16.pdf) [Accessed April 8, 2021].
- [36] M. Bellis, “The history of fluorescent lights,” 2019. Available at: <https://www.thoughtco.com/history-of-fluorescent-lights-4072017> [Accessed May 27, 2021].
- [37] J. Bell, “The art and craft of fluorescent lamps,” vol. 97, pp. 718–720, 1983.
- [38] N. Holonyak and M. Feng, “Coherent (visible) light emission from ga(as1xpx) junctions,” vol. 1, pp. 82–83, 1962.
- [39] N. Holonyak and M. Feng, “The transistor laser,” vol. 43, pp. 50–55, 2006.
- [40] D. L. Andrews, “Photonics, volume 3: Photonics technology and instrumentation,” John Wiley Sons, 2015.
- [41] J. Cho, J. H. Park, J. K. Kim, and E. F. Schubert, “White light-emitting diodes: History, progress, and future,” vol. 11, p. 1600147, 2017.
- [42] S. Nakamura, T. Mukai, and M. Senoh, “Candela-class high-brightness in-gan/algan double-heterostructure blue-light-emitting diodes,” vol. 64, pp. 1687–1689, 1994.
- [43] NYT, “Fans of l.e.d.’s say this bulb’s time has come,” 2008. Available at: <https://www.nytimes.com/2008/07/28/technology/28led.html> [Accessed May 27, 2021].

- [44] M. Bellis, “The history of sonar,” 2020. Available at: <https://www.thoughtco.com/the-history-of-sonar-1992436> [Accessed May 28, 2021].
- [45] M. A. Ainslie, “A brief history of sonar,” in *Principles of Sonar Performance Modelling*.
- [46] M. I. Skolnik, “Radar,” Encyclopedia Britannica, 2020. Available at: <https://www.britannica.com/technology/radar> [Accessed May 28, 2021].
- [47] U. Wandering, “Introduction to lidar,” in *Lidar: Range-Resolved Optical Remote Sensing of the Atmosphere*, pp. 1–18, Springer New York, 2005.
- [48] P. F. McManamon, “History of lidar,” in *LiDAR Technologies and Systems*, pp. 29–87, SPIE, 2019.
- [49] T. Mai, “Global positioning system history,” National Aeronautics and Space Administration, 2017.

A

Appendix 1

Assuming the input files are “Word2Vec model”, “document keywords” (csv file with 2 columns: *patent application id* and *keywords*), “application filing years” (csv file with 2 columns: *patent application id* and *patent filing year*), and “normalization factors” (csv file with 2 columns: *patent filing year* and *normalization factor*), the source code for the command line application of the SPBS method used in this thesis is as follows:

```
"""
Usage:
    main.py
        [(-n <output_file_name>|--name <output_file_name>)]
        [(-s <min_similarity>|--similarity <min_similarity>)]
        [--mean]
        [--norm]
        [--cumsum]
        [-t <tech_title>|--title <tech_title>]
        [--start <start_year>]
        [--end <end_year>]

Options:
    -n <output_file_name> --name <output_file_name>      Name of the file to save the similarity scores
                                                           in [default: output]
    -s <min_similarity> --similarity <min_similarity>    Minimum similarity score as a basis to filter
                                                           out the patents [default: -1]
    --mean                                                Instead of summing word vectors for query, it
                                                           enables calculating mean vectors for the words
                                                           in both positive and negative list
    --norm                                                Normalize number of patents before plotting
    --cumsum                                              Calculate cumulative sum by years before plotting
    -t <tech_title> --title <tech_title>                 Title of the technology. Appears as a legend title
                                                           on the figure
    --start <start_year>                                Start year for plotting [default: 0]
    --end <end_year>                                    End year for plotting [default: 2015]
"""
import os, sys, logging, re, pickle
from docopt import docopt
from gensim.models.word2vec import Word2Vec
import pandas as pd, numpy as np
import dask, dask.dataframe as dask_DataFrame
from dask.diagnostics import ProgressBar, ResourceProfiler
from sklearn.metrics.pairwise import cosine_similarity
import matplotlib.ticker as ticker
import matplotlib.pyplot as plt

logging.basicConfig(level=logging.WARNING,
    format="%(asctime)s [%(threadName)s] [%(levelname)s] --- %(message)s",
    datefmt="%H:%M:%S",
    stream=sys.stdout)
logger = logging.getLogger("SBPS")
logger.setLevel(logging.INFO)

dask.config.set(scheduler='processes')
```

A. Appendix 1

```
CURRENT_DIR = os.path.dirname(__file__)
KEYWORDS_PATH = os.path.join(CURRENT_DIR, "<relative_path_keyword>")
MODEL_PATH = os.path.join(CURRENT_DIR, "<relative_path_word2vec_model>")
FILING_YEAR_PATH = os.path.join(CURRENT_DIR, "<relative_path_filing_year>")
NORM_FACTORS_PATH = os.path.join(CURRENT_DIR, "<relative_path_norm_factors>")

# Column names used in the CSV files
COL_APPLN_ID = "appln_id"
COL_FILING_YEAR = "appln_filing_year"
COL_NORM_FACTOR = "norm_factor"
COL_KEYWORDS = "keywords"
COL_SIMILARITY = "similarity"

# Global variables
TECH_TITLE = None
MEAN = False
NORM = False
CUMSUM = False

def main(output_file_name, min_sim, start_year, end_year):
    output_file_name = f"{output_file_name}_mean" if MEAN else output_file_name
    output_file_path = os.path.join(CURRENT_DIR, "data", "cos_sim", f"{output_file_name}.csv")

    # Check if similarity score file exists in the disk otherwise compute
    if os.path.exists(output_file_path):
        logger.info(f"Similarity score file for '{output_file_name}' is found. Reading...")
        sim_df = pd.read_csv(output_file_path, usecols=[COL_APPLN_ID, COL_SIMILARITY])
    else:
        sim_df = compute_similarity_scores(output_file_path)

    # Filter out patents and plot histogram if min_sim is given
    if min_sim > 0:
        df = pd.read_csv(FILING_YEAR_PATH, usecols=[COL_APPLN_ID, COL_FILING_YEAR])
        df = df[(df[COL_FILING_YEAR] >= start_year) & (df[COL_FILING_YEAR] <= end_year)]
        df = df.join(sim_df.set_index(COL_APPLN_ID), on=COL_APPLN_ID)
        df = df[df[COL_SIMILARITY] >= min_sim]
        if df.empty:
            logger.warning(f"There is no patent found that has similarity score above {min_sim}")
            return

        df = df.groupby([COL_FILING_YEAR]).agg(['count'])
        df.columns = ["_".join(x) for x in df.columns.ravel()]
        df = df.sort_values(by=COL_FILING_YEAR)

    # Column name to use in y-axis
    y_column_name = f"norm_{COL_APPLN_ID}_count" if NORM else f"{COL_APPLN_ID}_count"

    # Normalize data: increase the magnitude of the yearly values proportional
    # to how much PATSTAT database miss abstract info in the corresponding year
    if NORM:
        norm_factors_df = pd.read_csv(NORM_FACTORS_PATH, usecols=[COL_FILING_YEAR, COL_NORM_FACTOR])
        df = df.join(norm_factors_df.set_index(COL_FILING_YEAR), how="left")
        df[y_column_name] = df.apply(lambda row: row[f"{COL_APPLN_ID}_count"]*row[COL_NORM_FACTOR],
                                   axis=1)

    if CUMSUM:
        df[y_column_name] = df[y_column_name].cumsum()

    df.reset_index(inplace=True)

    logger.info("Plotting figure")
    figure_file_name = _generate_figure_file_name(output_file_name, min_sim)

    # Fixing the font size of axis ticks and labels on the figure
    plt.rcParams.update({"font.size": 19 if CUMSUM or df.shape[0] < 60 else 12})

    plot_kind = "line" if CUMSUM else "bar"
    df.plot(kind=plot_kind, x=COL_FILING_YEAR, y=y_column_name, color="Slateblue", figsize=(24, 8))
```

```

if CUMSUM:
    # SPECIFIC PARAMETERS FOR LINE GRAPH
    # Filling area below the line on the figure
    plt.fill_between(x=df[COL_FILING_YEAR], y1=df[y_column_name], color="skyblue", alpha=0.4)

    min_year = start_year if start_year else df[COL_FILING_YEAR].min()
    max_year = df[COL_FILING_YEAR].max()

    # Set value ranges on axes
    plt.xlim(min_year, max_year)
    plt.ylim(bottom=0)

    ## Set major ticks: starts from the first value divisible by 5, frequency is 5 years
    min_year_div_5 = min_year + (5 - min_year%5 if min_year%5 != 0 else 0)
    plt.xticks(np.arange(min_year_div_5, max_year+1, 5))

    # Setting label for x-axis
    plt.xlabel("Application Filing Year", labelpad=20)

    # Fixing the legend
    plt.rcParams.update({"font.size": 19, "font.weight": "bold"})
    plt.legend(title=TECH_TITLE, labels=[_generate_figure_legend("#{Patent Applications}")])

    figure_file = os.path.join(CURRENT_DIR, "figure", figure_file_name)
    _make_dirs(figure_file)
    plt.savefig(figure_file, bbox_inches='tight')
    logger.info(f"Figure is saved at: {figure_file}")

def compute_similarity_scores(output_file_path):
    """
    A pipeline that triggers the stages of the SBPS method in order
    """
    logger.debug(f"Processing: {KEYWORDS_PATH}")
    logger.debug(f"Will be saved at: {output_file_path}")

    # Interactively build query vector
    query_vector = build_query_vector()

    # Read all data into memory as pandas dataframe and convert to partitioned dask dataframe
    # to enable parallelism
    df = pd.read_csv(KEYWORDS_PATH, usecols=[COL_APPLN_ID, COL_KEYWORDS])
    ddf = dask_DataFrame.from_pandas(df, npartitions=24)
    del df

    with ProgressBar(), ResourceProfiler() as rprof:
        ddf = ddf.map_partitions(run_partitions, query_vector).compute()

    # Save dataframe into disk
    ddf = ddf[[COL_APPLN_ID, COL_SIMILARITY]]
    _make_dirs(output_file_path)
    ddf.to_csv(output_file_path, mode='w', index=False, header=True)
    return ddf

def calculate_doc_vector(text, w2v_model):
    """
    Helper function to build a document vector given document keywords and word2vec model
    """
    vector = np.sum([w2v_model.wv[word] for word in str(text).split() if word in w2v_model.wv],
                    axis=0)
    is_valid_vector = isinstance(vector, np.ndarray) and len(vector) == w2v_model.vector_size
    return vector if is_valid_vector else np.array([0]*w2v_model.vector_size)

def run_partitions(df, query_vector):
    """
    This function is expected to be called by dask dataframes to enable parallelism.
    """
    # Since this function will be run simultaneously in different processes, loading separate
    # word2vec model in each would make the models work in a safe and expected manner.
    w2v_model = _load_w2v_model()
    query_vector = np.array([list(query_vector)])

```

A. Appendix 1

```
df[COL_SIMILARITY] = df[COL_KEYWORDS].apply(calculate_doc_vector, args=(w2v_model, ))
cos_sim_values = cosine_similarity(list(df[COL_SIMILARITY]), query_vector)
df[COL_SIMILARITY] = [val[0] for val in cos_sim_values]
return df

def build_query_vector():
    """
    Facilitates the query building stage
    of the SBPS method
    """
    w2v_model = _load_w2v_model()
    positive = []
    negative = []

    print("COMMANDS> p (+|-) <word>: {alter positive list}\t|\t" \
          "n (+|-) <word>: {alter negative list}\t|\tf: {finish query}\n")
    while True:
        print("-"*100)
        print(f"positive={positive}, negative={negative}\n")
        if len(positive) or len(negative):
            most_similar_words = w2v_model.wv.most_similar(positive=positive, negative=negative)
            print("Most similar words: {}".format([(pair[0], round(pair[1], 2)) for pair in
                                                  most_similar_words]))
        user_input = input("CMD> ").lower().strip()
        matches = re.match(r"^(p|n)\s*(+|-)\s*([a-zA-Z]*)|(f)$", user_input)
        if not matches:
            print("Entered command is not correct. Please follow the guide below:")
            print("COMMANDS> p (+|-) <word>: {alter positive list}\t|\t" \
                  "n (+|-) <word>: {alter negative list}\t|\tf: {finish query}\n")
            continue

        g_list = matches.group(1)
        g_action = matches.group(2)
        g_word = matches.group(3)
        g_finish = matches.group(4)
        if g_finish:
            break
        else:
            if g_list == "p":
                if g_action == "+":
                    if g_word not in w2v_model.wv:
                        print("The word doesn't exist in the vocabulary")
                    else:
                        if g_word not in positive:
                            positive.append(g_word)
                else:
                    if g_word in positive:
                        positive.remove(g_word)
            else:
                if g_action == "+":
                    if g_word not in w2v_model.wv:
                        print("The word doesn't exist in the vocabulary")
                    else:
                        if g_word not in negative:
                            negative.append(g_word)
                else:
                    if g_word in negative:
                        negative.remove(g_word)

    # Now that positive and negative lists are ready,
    # we can compute a vector out of the words given
    p_vector = np.array(0)
    for p_word in positive:
        if p_word in w2v_model.wv:
            p_vector = np.add(p_vector, w2v_model.wv[p_word])

    n_vector = np.array(0)
    for n_word in negative:
        if n_word in w2v_model.wv:
            n_vector = np.add(n_vector, w2v_model.wv[n_word])
```

```

if MEAN:
    if positive: p_vector /= len(positive)
    if negative: n_vector /= len(negative)

return np.add(p_vector, -n_vector)

def _load_w2v_model():
    """
    Helper function for loading
    Word2Vec model
    """
    model = Word2Vec.load(MODEL_PATH)
    model.wv.most_similar("network")
    return model

def _generate_figure_file_name(output_file_name, min_sim):
    """
    Helper function to generates figure file name based on global
    variable values
    """
    figure_name = f"{output_file_name}_{min_sim}"
    figure_name = f"{figure_name}_norm" if NORM else figure_name
    figure_name = f"{figure_name}_cumulative" if CUMSUM else figure_name
    return f"{figure_name}.jpg"

def _generate_figure_legend(legend):
    """
    Helper function to generates a legend for a figure
    based on global variable values
    """
    legend = f"{legend} (Normalized)" if NORM else legend
    legend = f"Cumulative {legend}" if CUMSUM else legend
    return legend

def _make_dirs(file_path):
    """
    Helper function to create directories in the file
    system given the file path
    """
    dir_path = os.path.dirname(file_path)
    if not os.path.exists(dir_path):
        os.makedirs(dir_path)

if __name__ == "__main__":
    args = docopt(__doc__)

    # Setting global variables
    MEAN = args["--mean"]
    NORM = args["--norm"]
    CUMSUM = args["--cumsum"]
    TECH_TITLE = args["--title"]

    main(args["--name"], float(args["--similarity"]),
        int(args["--start"]), int(args["--end"]))

```