Vrije Universiteit Amsterdam

Master Thesis Business Analytics

Application of Topic Modeling for Text document Management to facilitate the research phase of consultants

Author
Q.D.A Busch

Supervisors
MSc. S. Salmi
MSc. I.A. Wensing

Date
31st October, 2022

Application of Topic Modeling for Text document Management to facilitate the research phase of consultants

Author
Q.D.A Busch

First Supervisor
MSc. S. Salmi

External Supervisor
MSc. I.A. Wensing

Second reader
Prof. dr. G. Koole

Vrije Universiteit Amsterdam
Faculty of Science
Business Analytics
De Boelelaan 1081
1081 HV Amsterdam

Accenture Amsterdam
Applied Intelligence
ITO
Gustav Mahlerplein 90
1082 MA Amsterdam

Abstract

The exponential growth of published articles makes it increasingly difficult for researchers to find the appropriate articles. Queries in document databases, such as Google Scholar, provide millions of results, all of which may be most relevant to you. If you want to research a really specific topic or goal, a query can be modified or refined. Often, however, the quantity is not reduced to a manageable degree. Moreover, without such a specific topic or goal, how can you find what topics exist within your search area. In consulting, a distinctly busy world, where clients expect immediate answers to their questions, an efficient, accurate method of finding the right information is needed. The current study proposes a solution of using Topic modeling techniques to process this vast amount of unstructured text.

Topic modeling is considered a complex task within Text mining and Natural Language processing. Finding "hidden" topics in a collection of articles is already considered a complex task for any human. Let alone for machines that do not understand the context, meaning and interpretation of text. This thesis focuses on finding and testing different Topic modeling techniques to answer the research question: How can Topic modeling and clustering be useful in improving the research phase for consultants in Google Scholar articles?

This study investigates four different Topic modeling techniques to determine which one best fits the research problem of improving the research phase of consultants. The methods used are Latent Dirichlet Distribution (LDA), Term Frequency - Inverse Document Frequency (TF-IDF), Word2Vec and BERTopic. The techniques are evaluated based on their ability to accurately and efficiently create specific topic clusters. This ability is quantified using measures such as topic coherence, topic diversity and human interpretability.

The focus of the current study is to find the technique that provides the most value for the research phase of consultants. This study concludes that using the BERTopic model provides the most appropriate results for the research problem. The resulting topic clusters have the highest average topic coherence, the least diverse results and the highest human interpretability. The topic coherence measure Cv resulted in an average coherence of 0.7, meaning that the model is considered very good in the recognizing and clustering similar text documents. The human interpretability is substantiated by the clear topic labels created by the model. Supported by the results, it is argued that Topic modeling and clustering can be very useful in improving both the accuracy and efficiency of a consultant's research in Google Scholar articles.

Keywords— Topic modeling, unlabeled text clustering, LDA, TF-IDF, Word2Vec, BERTopic, K-means, Topic coherence Cv

Acknowledgements

This thesis is written as a graduate project to fulfill the requirements for the master's degree in Business Analytics at the Vrije Universiteit Amsterdam. The thesis is written during a 6-month graduate internship at Accenture in the Applied Intelligence team. First of all I would like to thank my supervisor Inez Wensing from Accenture for all her suppport and bi-weekly meetings to discuss my approach to the research problem during the internship. Furthermore, my supervisor Salim Salmi from Vrije Universiteit Amsterdam for all his feedback on my thesis and giving good suggestions during my thesis. Finally, I would like to thank all other colleagues at Accenture who helped me during my internship for offering to help during my research, their interest and enthousiasm and the nice work atmosphere.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In 2010 A. Jinha of the University of Ottawa completed a study that estimated that the 50th millionth science article was published in 2009.[17] The STM, Association of Scientific, Technical, and Medical Publishers, produces an annual overview of scientific and scholarly journal publishing. The 2015 report stated that 2.5 million new scientific papers are published annually. By 2020, this has already reached the number of 4.2 million records under the category "citable" documents.[91]

This exponential growth in published articles makes it increasingly hard for researchers to find the right articles. When querying document databases, such as Google Scholar, millions of results are provided, all of which may be the most relevant to you. In case you have a really specific topic or goal, a query can be customized or refined. Often, however, the quantity is not reduced to a manageable degree.

Within Consultancy, a specific type of research is required, namely that consultants need an overview of a search are or help in finding specific information about a particular problem for their project. Clients expect consultants to be experts in every relevant aspect of their project or business expertise. A client may demand this on short notice, which requires a quick, efficient, but thorough research method. Finding the right information in this vast bulk of articles can be seen as a needle in a haystack situation. To help consultants and other researchers navigate through this haystack and cope with the volume of literature, a combination of Text mining and Natural language Processing methods (NLP) is proposed as a solution.

This research proposes to answer the problem of finding the right information and guide consultants in doing research on their clients' expertise by using Text mining and NLP and more specifically Topic Modeling. Text mining is the process of deriving high-quality information by exploring and analyzing large amounts of unstructured text. This process is supported by NLP, which is the branch of artificial intelligence (AI) that is concerned with the ability of computers to understand text and speech similarly as how humans can. The capabilities of Natural Language Processing techniques range from simpler text analysis to text summarization, and in this research a combination of capabilities is explored. The main NLP technique for this research discussed in this paper is Topic Modeling. Topic Modeling includes all forms of topic creation, clustering, and classification of documents from a large number of text documents. Topic Modeling can be used to create the solutions requested by this study. The analysis of the texts based on Topic Modeling can lead to the insights needed to make the right decisions about which texts, what type of texts, and what methods should be used during the study. Topic modeling ensures that with appropriate preprocessed data and insights, the actual topics that appear in the document dataset are extracted, and the documents are clustered based on their internal topics. Based on these internal topics and topic cluster, the relevant documents and insights are

found in a search area. This is the basis of this study and therefore Topic Modeling is proposed as the solution to achieve the goal of helping consultants conduct accurate and efficient research.

In this paper, multiple Topic Modeling methods are examined with the main objective of classifying articles into topic clusters. A variety of Topic models exist, and also pre-trained language models, such as BERTopic are known to be used for similar goals. The Topic modeling methods in combination with other NLP techniques, like Keyword Extraction and Named Entity Recognition, will create an overview of the different components found in a group of documents. A representative Google Scholar search query is used for creating a text document database that will function as a test case to evaluate the Topic models. The text document database is created as the representation of the results of the search query that Google Scholar selected. This is done by creating a data set of documents that all include the same subject, which would represent the search query input. The research question that will be answered by this scientific paper is formulated concisely below. The question this paper attempts to answer is:

"How can Topic Modeling and clustering be useful in improving the research phase for consultants in Google Scholar articles?"

This research question includes multiple aspects, and a breakdown of the insights that are targeted is found in Chapter 2. The rest of the paper is constructed in the following order. First, all concepts and techniques used in this study are explained in Chapter 3. Subsequently, the Web scraping, Data exploration and preparation are discussed in Chapter 4. Next, the methods and their implementation are described in Chapter 5 and their results are displayed in Chapter 6. At last, the discussion and conclusion of the thesis are explicated in Chapters 7 and 8

Chapter 2

Research questions

This chapter includes all preliminary knowledge required to understand this study. First, the research angles and intended output are briefly explained. Then, the research question is elaborated to clarify the aspects addressed in this study.

2.1 Research angle

The purpose of this study is to gain insights and knowledge on how a consulting firm could improve the research phase of their consultants. The angle is from a business point of view to understand efficient techniques that are an improvement for finding the relevant documents quickly and accurately. The intended output is considered as exploring the possible techniques for this purpose and creating dense human interpretable topic clusters of text documents from a document database. A high density of the clusters is desirable, as the documents in the same cluster should be similar. Human interpretability is crucial from the perspective that the technique should be easy to use and thus will actually be used.

2.2 Research sub-questions

Answering the research question involves highlighting multiple aspects that substantiate the research question. First, the methods of scraping the necessary data are tested, and what kind of text both includes enough information and is easily accessible. Second, the data analysis and data preparation steps required for the various Topic Modeling methods are examined. Third, the Topic models are used to compare their suitability for the present problem, while also examining their advantages and disadvantages. Finally, the results are calculated and compared, and conclusions are drawn about the research question and sub-problems.

To evaluate Topic models, multiple metrics are used, and it is approximated which best describes the desired results. Topic Models are known to be hard to validate as there is a large subjective factor regarding whether topics include the right documents, are interpretable, and even whether humans would have chosen similar topic clusters. However, combining multiple metrics creates a good comparison that this research seeks to quantify the results and argue which methods perform best. Established methods, like Latent Dirichlet Allocation (LDA), are compared to methods that use feature creation and text clustering, such as the Term frequency-inverse document frequency (TF-IDF) measure or word embeddings created by the Word2Vec algorithm. Then, a language model based-method, BERTopic, which is fairly new in this area of research, is reviewed. These methods are examined in terms of how dense clusters can be formed, how distinct these clusters are, and how easy the clusters can be labeled. Cluster density combined with distinctiveness looks at the balance between intrinsic and extrinsic measures. After labeling, it is interesting to see how interpretable the topics are to us humans.

Chapter 3

Literature Review

This chapter discusses the various aspects that emerged in this study and explains important knowledge needed to understand the research. First, the methods of data collection are explained, including some background on text scraping from the Internet. In addition, both the text mining and natural language processing (NLP) components of the study are discussed. The NLP methods used in the later stages of the study are explained.

The difference between text mining and NLP is the purpose for which the techniques are used. Text mining is primarily used to extract information from both unstructured and structured content, with the research focusing more on the structure than the meaning of the content. The main use of NLP is to understand human language by analyzing text, speech or grammatical syntax as a human would understand it. In this research, the main purpose would be the analysis of text, but text is needed for testing different analysis techniques. Therefore, text scraping will be used to retrieve the text that fits the research problem and can be used for analysis.

3.1 Web scraping

Data collection is the process of finding and analyzing relevant data from various sources to arrive at insights and solutions to research problems. The text mining framework built for this research is also largely dependent on data collection. In addition to correctly processing the data to fit the model, the data must first be collected. In our case, it is necessary to collect a list of articles resulting from a search query and retrieve the text data of those articles. Collecting text documents and other forms of data from Web sources and structuring the data is called Web scraping. [54] A Web scraper is a specialized tool designed for the accurate and fast extraction of data from any type of Web source. Each Web source or website requires a different design and complexity. Therefore, a different scraper is required to be built for each publisher. This can increase the difficulty of text scraping, but for text mining problems, it is essential that the data is gathered correctly and no inconsistencies occur.

Web scraping became popular shortly after the introduction of the World Wide Web (WWW) in 1989. Very basic Web scraping dates back to when one Tim Berners-Lee, a British scientist, created the World Wide Web. The World Wide Web was created as a platform where scientists from around the world could share information and articles. As the World Wide Web became globally accessible, the technique of web scraping was born: Collecting all relevant text information from the Web. Web scraping requires three main features of the WWW. [76]

1. All specific websites are identified by a designated URL, uniform resource locator, a unique Web address that distinguishes not only websites but each individual Web page.

2. Navigation through these Web pages is done using hyperlinks, which cause one to move from one hypertext document to another Web page or text document. In the WWW, hyperlinks are activated by clicking a highlighted word or image.

3. Every Web page consists of text data, images, audio, videos etc. These different types of data are attached to the Web page and contain information bound to the page.

In June 1993, the first World Wide Web Wanderer is created for the purpose of measuring the size of the Web. However, the potential of Web scraping is also created by this 'first Web robot'. [76] About 10 years later, a new Web scraper is created, BeautifulSoup. BeautifulSoup is considered an HTML or XML document parser and is a Python-based library. A document parser like BeautifulSoup creates a parse tree for a text document. Such a parse tree is used to represent the syntactic structure of a text according to its context. A parse tree representation allows BeautifulSoup to understand the structure of the text and with this understanding, the algorithm is able to find and extract relevant data, Web scraping. Especially with the growing amount of data, the required information becomes too large to copy manually. So, Web scraping with BeautifulSoup provided a faster way of obtaining information needed for an increasing number of problems. [12] Still, not all websites prohibit the ability to download their content.

In more current times, the ability of Web scraping is recognized to build and power revolutionary business applications. It is used for a broad perspective of business improvement, from informing executive decisions to monitoring customer service. It is even applied to competitor price tracking in the e-commerce, as it provides an accurate and fast display of sometimes highly dynamic prices. [40] This allows companies to stay ahead and adjust their prices accordingly, which is considered a valuable goal for nearly every industry. In research, Web scraping is still mostly used to extract the most relevant information. In this article, articles resulting from the search query are scraped as HTML scripts using BeautifulSoup.

3.2 Text mining

Text mining is considered an artificial intelligence technology that examines large collections of documents to discover new information and gain insights. Using natural language processing (NLP), text in documents is converted into normalized structured data suitable for analysis or driving machine learning algorithms. In order to test NLP techniques, a Text mining framework is needed. Within this Text mining framework, the methods can be deferred while the rest of the framework is as similar as possible to compare the effects of the specific NLP techniques. A text mining process generally involves five distinct phases. [31]

1. According to Chakraborty et al. the first phase involves collecting the text data, in our case the article text data. This data is likely to be unstructured and collected from the various articles from scientific publishers. An important step for any Text mining framework and explained in Section 3.1

2. The next phase concerns parsing the text. The unstructured text data is preprocessed into data that can be understood by any model applied at a later stage. This data is used for model learning to improve results.

   Parsing text includes extracting, cleaning, and creating a dictionary. This research, which is based on article texts, focuses on sentence identification, removal of stop words and word stemming. This implies removing irrelevant words and parsing the extracted words to identify entities, as well as reducing words to their root form. An additional activity could be changing words to the most common synonym. The exact steps taken are illustrated in Section 4.3.

3. Step three concerns filtering out the terms that are not relevant in distinguishing or summarizing the documents. These can be prepositions and common articles or similar less relevant words, as well as words that appear in all documents, such as "edge computing" in our example. During this filtering process, numbers and punctuation are also removed. This step is usually performed manually, as it requires a fair amount of domain experience. After all irrelevant terms are filtered out of the term-document matrix and a new weighted matrix is created using various term-weighting techniques.

4. The transformation step, after the text is parsed and filtered, involves the numerical representation of the data. For the transformation of the text into a data format needed for commonly used methods, such as Latent Dirichlet Allocation (LDA). The result of these methods is a term-document matrix that represents the frequencies of the words or other features for each document. The number of documents and the different terms that occur in the documents determine the dimension of the resulting matrix. The purpose of this step is to create the required input for the last phase, which is the actual Text mining part. The required input depends on the methods used in conjunction with the research problem.

5. The last phase is the text mining phase, which retrieves the information that will solve the problem at hand. This is done by applying traditional data mining algorithms, such as clustering, classification, association analysis, and link analysis. In this particular research, one needs clusters of documents as results, so a clustering algorithm will be applied.

3.3 NLP Topic Modeling

Natural Language processing comprises the field of artificial intelligence in which computers understand, analyze and derive meaning from human language. This includes all research and application of texts, speech, and syntax. [64] NLP techniques focus on gathering and organizing knowledge about how people understand language so that desired tasks can be performed. Sentiment Analysis, Named Entity Recognition, Summarization, Topic Modeling, Text Classification, Keyword Extraction and Lemmatization and stemming are considered 7 common NLP techniques, with Topic Modeling being the main technique applied in this research. An attempt is made to achieve the research goal by using different Topic modeling techniques.

Topic Modeling is considered the root of this research because it is known for unsupervised clustering of documents based on internal topics. Based on different types of models, topics that appear in a corpus, a collection of documents, are discovered. The technique operates as follows: a set of documents is scanned, word and phrase patterns are detected within the texts, and the word groups and similar expressions that best characterize a set of documents are used for the clustering of the documents. [59]

There are many different methods for Topic Modeling, all optimized for different structures and text formats. The four methods included here are Latent Dirichlet Allocation (LDA), Term Frequency-inverse document frequency (TF-IDF), Word2Vec, and BERTopic, which give a good view of what aspects are important for clustering document across topics. There are numerous other methods available and applicable for this study, but these four methods are a good representation of the types of models available. [86]

The study begins by applying more established methods, such as LDA and Non-Negative matrix factorization (NMF). The performance of LDA was higher, as was the potential of the technique, leaving NMF out of further study. Next, two Topic modeling techniques with text clustering are explained. Both techniques are based on newer measures and ideas in Topic Modeling that seem

promising, the TF-IDF measure and the Word2Vec algorithm. Research shows that methods based on the TF-IDF measure can achieve high performance for certain problems, especially for information retrieval. [22, 55] The Word2Vec algorithm, based on word embeddings, would understand the full idea of the documents better than previous methods. [39] In 2018, Li, Changzhou, et all, specifically worked on a similar application of Word2Vec for abstracts of academic texts. [50] Last, Topic modeling is applied using a language model BERT and its application BERTopic. This method is relatively new in the field, but is already yielding good results with small data sets. In particular, because the language model is pre-trained on millions of texts and can understand texts more deeply and does not require preprocessing. [87]

3.3.1 Basic Topic modeling techniques

In many studies, the first model can be considered a benchmark model to validate the results. In topic modeling, Latent Dirichlet Allocation (LDA) was long considered the best technique, but numerous alternatives have been proposed in recent years. It strongly depends on the dataset and the research goal which method is preferred. In this study, several Topic modeling techniques are tested and compared, and it is interesting to compare a more established Topic modeling technique such as LDA with newer models. Below, LDA is proposed as the first Topic modeling technique, and this model is expected to give a good indication of task difficulty.

LDA

Latent Dirichlet Allocation, in short LDA, is an unsupervised clustering technique that was seen as the most popular in this field for a long time and was introduced by D. Blei in 2003. [8] The way this technique extracts 'hidden topics' from data is conveyed by the term latent, which in a literal sense means concealed or hidden. Dirichlet Allocation as the technique is based on the Dirichlet distribution and process. A Dirichlet process entails a distribution over distributions, as every draw from a Dirichlet process itself is also a distribution.[56]

An LDA topic model calculates the probabilities of certain distributions, such as whether words occur together, are expected to occur or are in similar contexts. Based on those probabilities, text is classified and grouped by topic, meaning that the documents contain the distributions of words and topics. This leads to some key assumptions when using LDA:

1. The first is the so-called distributional hypothesis, which means that words that often appear together are assumed to be close in meaning;
2. The second is that documents include a mixture of topics;
3. the The last is that topics include a mixture of words (or tokens).

Therefore, documents are known as the probability density of topics, and topics are known as the probability density of words.[67]

An LDA model is called a generative model because it tries to discover the underlying mechanism that generates the topics and articles. Thus, the model tries to mimic a mechanism based on the output, rather than mimicking the output itself. The output is the document-word, or term-document, matrix, a matrix with the documents on one axis and the words on the other. It is a binary matrix where a one in the matrix indicates that a word appears in that document. An example of such a matrix is Figure 3.1, where documents are represented by D_1, ..., D_5 and the words by W_1, ..., W_8. [82]

Document Word Matrix

	W1	W2	W3	W4	W5	W6	W7	W8
D1	0	1	1	0	1	1	0	1
D2	1	1	1	1	0	1	1	0
D3	1	0	0	0	1	0	0	1
D4	1	1	0	1	0	0	1	0
D5	0	1	0	1	0	0	1	0

Figure 3.1: Document-Word Matrix

The model uses the document-word matrix to create two new matrices, a document-topic matrix, and a topic-word matrix. A document-topic matrix contains the relationship between a document and a topic, where multiple topics can appear in a single document. In the topic-word matrix, the relationship between topics and words can be found, where one word can be connected to multiple topics, as well as topics containing many words. Therefore, these matrices can become very large. How the two topic matrices are formed is shown in Figure 3.2, where the topics are represented by K_1, ..., K_6. [82]

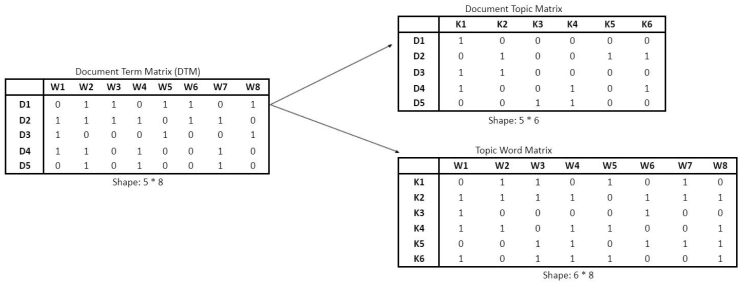


Figure 3.2: Creation Topic Matrices

This creates the distribution over distributions part of the Latent Dirichlet allocation, which is the combination of the document-topic and topic-word distribution based on α and β, respectively. The α is called the Dirichlet prior of the topics and the sparsity or density of the topics is indicated by this prior. A low Dirichlet prior, $\alpha < 1$, means that the topics are fairly separated and less likely to be evenly distributed across the documents. A high Dirichlet prior, $\alpha > 1$, indicates a greater likelihood of an even mix of topics within the documents or more similar topics. The Dirichlet prior of the words is represented by the β and denotes the density or sparsity of the words. A high β indicates a greater similarity between the words and their associated topics, and a low β suggests greater sparsity. These two hyperparameters are used by the LDA model to try to mimic the output of the document-word matrix. Graphically, LDA can be represented as Figure 3.3.[82]

As mentioned, the Dirichlet process is viewed as a distribution over distributions and the graphical model above shows that the two distributions are combined in the LDA model. The yellow area of the model in Figure 3.3 refers to all m documents in the corpus M. The words in a document are covered by the pink area in Figure 3.3 where N is the total number of words in a document and w is one word in the document. Every observed word w is associated with a latent topic, denoted by z. The assignment of z to a topic word forms the topic word distribution θ in the corpus.

The optimization of the distributions of both Topic Matrices is based on LDA's principle that documents are a mixture of topics, while topics are a mixture of words. Therefore, LDA iteratively optimizes the backtracking process from the document level to determine which topics are

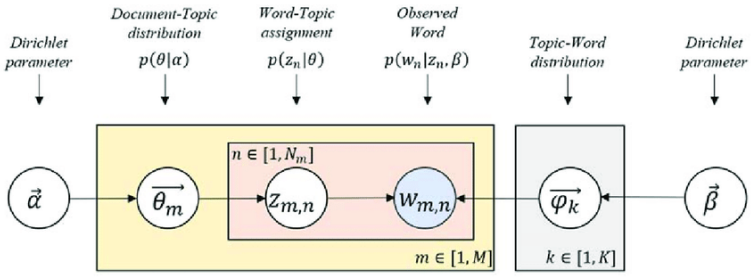


Figure 3.3: Graphical model of LDA

found and include similar documents. In the first iteration, the topics are randomly assigned to each word in the document. Then LDA provides both initial topic matrices and, by iteration across all documents and words, calculates two probabilities for each topic. The first probability, p_1, captures the proportion of words per document D already assigned to the topic. The second probability, p_2, entails the proportion of assignments, i.e., in which of these documents the same word w is also assigned to the topic. LDA creates a product probability of these two probabilities, which is the identifier for the topic to which a word belongs. At each iteration, the model converges to a steady state, which is the most optimized representation of the document-word matrix and topic-word matrix. The formulas of both separate probabilities are seen below:

$$p_1 = \text{proportion } \left(\frac{\text{topic } k}{\text{document } D}\right) \qquad \text{and} \qquad p_2 = \text{proportion } \left(\frac{\text{word } w}{\text{topic } k}\right)$$

3.3.2 Topic Modeling with Text clustering

In this study, the comparison of the different methods is based on the ability to create clusters. For the topic modeling techniques, LDA and BERTopic, the results generated are the topics into which the documents can be clustered. However, the TF-IDF and Word2Vec models only create an interpretation of the documents in the form of features. TF-IDF, or term frequency inverse document frequency, is a statistical measure to interpret the importance of words in a document relative to the entire corpus. The second method is an algorithm called Word2Vec, which interprets a document as a combination of word embeddings. Since both methods create a representation for each word in the document, a document consists of a set of features. These sets can be examined to determine which topics appear in the corpus.

Text clustering methods are considered unsupervised algorithms that can help summarize information from large amounts of text data by producing distinct text clusters. The goal is to understand the meaning and idea of the dataset. However, the main reason why text clustering is needed in this study is to group the text documents into separate categories, because then the text clustering capability of the topic model can be tested. An important method for text clustering is K-means because it can handle the high dimensionality of texts. [89] Several other clustering techniques are also investigated for this research. However, previous research has shown that for these specific techniques, K-means produces the best results. [65] Therefore, it was decided to cluster both techniques using K-means clustering.

First, the clustering technique is explained and then both feature creation methods are elaborated.

K-means

The K-means clustering algorithm is proposed as the most popular clustering formulation. [81] The original K-means principle was proposed by Forgy as early as 1965 [1] and by MacQueen in 1967.[2] The purpose of their articles was to introduce the technique behind K-means. A technique that appears to give reasonably efficient partitions based on within-cluster variance. Over time, the application has been modified, but the general idea is still similar. [29] A well-known optimization heuristic was proposed using Lloyd's algorithm in 1982.

In Lloyd's paper a good representation of the method is given. It is proposed that the algorithm starts with a random partition. This random partition is created using a number of k centroids, where the center of each cluster is represented by a centroid. The next part of the method consists of two steps, alternating iteratively. The first step, in which each item is assigned to the cluster represented by the nearest centroid. The second step, in which the centroids of the clusters are updated based on the updated partition from the previous step. For every cluster, a new centroid is calculated based on all data points that are currently part of the cluster. Then the process of the two steps is repeated. It is proven that this algorithm converges after a finite number of iterations. The algorithm is complete if the position of the clusters remains the same or if the change of the centers does not exceed a predetermined threshold. [3]

The exact method differs for the different applications of the algorithm, but the main idea remains the same. [10] The algorithm is a method of vector quantization that iteratively divides n data points into k disjoint clusters. The principle of the algorithm is based on finding the cluster closest to the object. The clusters are grouped based on minimizing these distances to the centroid of the cluster. Some techniques update this centroid, while others use the mean of the cluster, and even methods fix the centroid. The centroid or mean serve as the prototype of the cluster. The goal of k-means is achieved when the expected similarity between data points and the center point of the corresponding cluster is maximized. The quality of the result of the K-means algorithm is usually quantified by the variances of the data points within the cluster.

The mathematical representation of the document clustering algorithm is as follows. [6] The algorithms documents are composed using the vector space model, as in this paper the document vectors are clustered by K-means. In such a vector space model, every document d is considered as a vector $\boldsymbol{d}$ of all terms inside the document. For example, as the term frequency vector represented by the Equation 3.1 with tf_i the term frequency of term i in the document.

$$\boldsymbol{d}_{tf} = (tf_1, tf_2, \ldots, tf_n) \tag{3.1}$$

The vectors of the documents have different lengths because the documents do not have the same number of words. Therefore, the documents are normalized for comparison. Similarity measures, such as the cosine function are used to compare two documents. This cosine function is defined in Equation 3.2. The dot product of the vector is denoted by the $\bullet$ and the length of the vector $\boldsymbol{d}$ by $\|\boldsymbol{d}\|$.

$$\text{cosine}\,(\boldsymbol{d}_1, \boldsymbol{d}_2) = (\boldsymbol{d}_l \bullet \boldsymbol{d}_2) \,/\, \|\boldsymbol{d}_I\| \, \|\boldsymbol{d}_2\| \tag{3.2}$$

For a collection of documents, S, that form a cluster, the centroid vector is calculated using the Equation 3.3. The average of all terms is determined by dividing the summed weights of the terms within the document vectors by the total weights of the terms in the document set.

$$c = \frac{1}{|S|} \sum_{d \in S} \boldsymbol{d} \tag{3.3}$$

The similarity between a document and the centroid and also between centroids themselves is determined by the same cosine formula, see Equation 3.4.

$$\begin{aligned} \operatorname{cosin} e(d, c) &= (d \bullet c)/\|d\|\|c\| = (d \bullet c)/\|c\| \\ \operatorname{cosin} e\left(c_1, c_2\right) &= \left(c_l \bullet c_2\right) / \left\|c_l\right\| \left\|c_2\right\| \end{aligned} \tag{3.4}$$

This principle is required during K-means clustering, since the nearest centroid to the document is found by the dot product of a document and a cluster centroid. This is demonstrated in Equation 3.5.

$$\boldsymbol{d}_1 \bullet c = \frac{1}{|S|} \sum_{d \in S} \boldsymbol{d}_1 \bullet \boldsymbol{d} = \frac{1}{|S|} \sum_{d \in S} \operatorname{cosin}\left(\boldsymbol{d}_1, \boldsymbol{d}\right) \tag{3.5}$$

To evaluate the final partition, the square of the length of the centroid vector is represented as the average pairwise similarity between all documents within that cluster. In mathematical form, this is quantified as the Equation 3.6.

$$\frac{1}{|S|^2} \sum_{\substack{d \in S \\ d \in S}} \operatorname{cosine}\left(\boldsymbol{d}', \boldsymbol{d}\right) = \frac{1}{|S|} \sum_{d \in S} \boldsymbol{d} \bullet \frac{1}{|S|} \sum_{\boldsymbol{d} \in S} \boldsymbol{d} = \boldsymbol{c} \bullet \boldsymbol{c} = \|\boldsymbol{c}\|^2 \tag{3.6}$$

TF-IDF

Term Frequency-Inverse Document Frequency, or TF-IDF for short, is a statistical measure that evaluates how descriptive a word is for one document in a corpus. As the name of the measure indicates, it is a combination of two metrics multiplied. In more mathematical terms, the TF-IDF score for a word t in a document d from a set of documents D is calculated with the following formula:

$$tf - idf(t, d, D) = tf(t, d) \cdot idf(t, D)$$

The first metric, term frequency (TF), measures how often a word appears in a document. The simplest calculation involves counting the raw number of times the word appears. However, this value can be easily altered when shorter or longer texts are used. Therefore, the log frequency weight of the term is often the solution over the raw word frequency, because the relevance does not increase proportionally. The frequency portion of the formula is then adjusted. The simplest form of the TF formula can be seen as:

$$tf_{(t,d)} = \log(1 + \operatorname{freq}(t, d))$$

The second metric, inverse document frequency (IDF), indicates whether a term occurs frequently or rarely in the entire document corpus. The measure is obtained by dividing the total number of documents by the number of documents that contain the word in the corpus.

$$idf_{(t,D)} = \log\left(\frac{N}{\operatorname{count}(d \in D : t \in d)}\right)$$

By multiplying these two metrics, the TF-IDF measure is calculated. This number approaches 0 for words that are very common and appear in many documents, such as articles in a text or in our case the overarching topic of "edge" and "computing". For less common and especially words that appear only in one document, the measure will approach 1. This emphasizes the relevance of a word in that specific document in comparison to the other documents. [75]

TF-IDF is used in almost all search engine algorithms because it fits well with automated text analysis and specifically scores words in machine learning algorithms for natural language processing. The biggest hurdle of natural language processing techniques is the transformation of text into a numerical representation, known as text vectorization. The method used for text vectorization changes the result dramatically, so one must pick the method carefully.

A TF-IDF score is an example that can be used for algorithms such as Naive Bayes and Support Vector Machines, and is much better than the results of more basic scores computed by methods such as word counts. The best explanation for this improvement is that the documents with similar, relevant words will have similar vectors that can be recognized by the machine learning algorithm. Therefore, the topics in which different articles best fit can be better recognized.

Determining the relevance of words to a document, or TF-IDF algorithms, are usually used for Information Retrieval and Keyword Extraction functionalities. Therefore, it is interesting how good a Topic model based on this measure can perform. When search engines are used, the results are ranked based on the order of relevance to your query. The keyword extraction can also help to create insights into why the information is divided into article clusters.

Word2Vec

In 2013, the first article based on word vector representations was presented by Tomas Mikolov et al. working for Google. The aim of their article was to introduce techniques that can be used for learning high-quality word vectors from a large corpus of text. [28] Word2Vec is a two-layer neural network, that means, two hidden layers, which processes text corpus into a set of vectors or also called word embeddings. Word embeddings or vectors are the numerical representation of the word. Each unique word is mapped to a vector, which captures the characteristics of the word from the perspective of the overall text. This can include the definition, context, or even semantics of the word. Other research has shown that Topic modeling with Word2Vec is especially strong in semantic analysis. By creating context from the documents with the embeddings. [35, 70] Therefore, it is interesting to explore the application of this word embedding model for this study.

Word2Vec's hypothesis is that words that occur in similar contexts have similar meanings. Two words may look different to a computer from a single word perspective, such as 'happy' or 'glad'. However, their definition, the context in which the words occur and even the semantics are similar. By numerically representing these two terms, a word embedding, the computer can understand that these words should be considered very similar. Word2Vec's word embeddings are particularly strong in the part that the meaning of a word can be inferred based on its occurrence in the corpus, and the association between words and context. [75] An example to best illustrate this association is the "King - Man + Woman = Queen" equation, where the differences between vectors King-Man and Queen-Woman appear to be the same as King-Queen and Man-Woman, see Figure 3.4. [38]

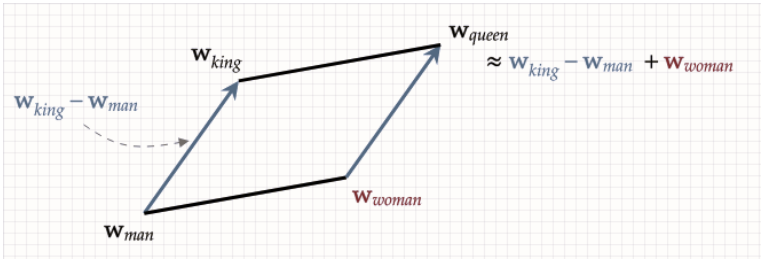


Figure 3.4: "King - Man + Woman = Queen" example

The Word2Vec algorithm is based on two methods, the Continuous Bag of Words and the continuous Skip-Gram model. [84] The Continuous Bag of words (CBOW) architecture is similar to that of a feed-forward neural network and predicts a target word from a list of context words surrounding the word in the text. Because it uses the continuously distributed representation of

context, it differs from a standard bag-of-words model. The architecture is similar to that of a bag-of-words model, since the order of words does not affect the projection. The words in the sliding context window of the CBOW method are all equally related to the target word because the distance from the target word i not considered. [53] Figure 3.5 presents the architecture of the CBOW as proposed in the original paper. [28]The continuous Skip-gram model is also a neural network with only one hidden layer. The model can be seen as the opposite of CBOW because instead of predicting the word from the context, this model tries to predict the context from a target word. So the words before and after the word in the text are given a probability of occurring in the context window. In Figure 3.5 on the right also shows the architecture of the Skip-gram model. [75]

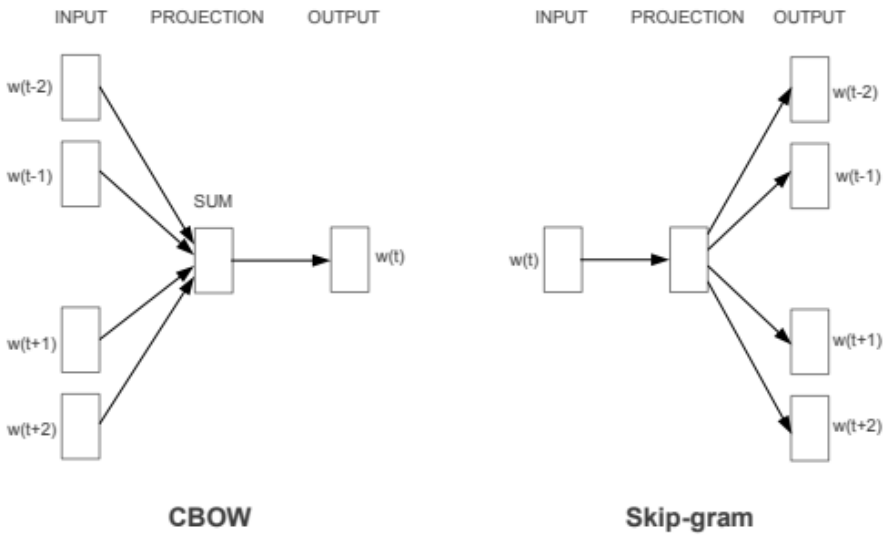


Figure 3.5: CBOW and Skip-gram

The choice of the model is based on the problem at hand. The CBOW method is trained much faster than the Skip-gram model, which might be preferred in cases where there is a lot of training data. Moreover, this model requires little memory, thus it does not require large amounts of RAM. Finally, this method is slightly more accurate for frequent words. However, the Skip-gram model performs well even when there is little training data. Moreover, the context prediction-based method may work better with data sets that contain words and phrases that are rarer. [48, 84]

Both techniques are used to create word embeddings for all tokenized words that occur in the corpus. The word embeddings are combined into document embeddings, which contain all the word embeddings that appear in the document. In Topic Modeling, the document embeddings are used for text comparison and topics are derived from the documents. These document embeddings are then clustered using K-means clustering to have the actual clustering that can be compared to the other methods. [77] It is expected that similar document embeddings lead to similar topics, which is an extension of the Word2Vec model hypothesis mentioned earlier. The hypothesis implies that words occurring in similar context, whose word embeddings are numerical representation, have similar meanings. This would imply that documents that occur in similar contexts whose document embeddings are numerical representation also have similar meanings. This implies that the meaning of the text is captured by the topic of the text.

3.3.3 Topic Modeling with Language models

Topic Modeling with language models differs from the latter two approaches to Topic modeling because the language model is assumed to already understand the language. A language model uses statistical and probabilistic engineering to establish a particular sequence of surrounding words. The corpus of a text is analyzed and it is predicted which words are likely to appear in the text. The goal of a language model is to understand a text on the same level as a human can, by understanding the meaning and perspective of a text. This research uses the language model BERT, and the Topic modeling application that uses BERT, BERTopic. [87]

BERTopic

In recent years, very interesting results have been obtained with BERTopic in various NLP tasks. [66] BERTopic is the topic modeling application of the language model BERT. BERT embeddings are leveraged against class-based TF-IDF to create dense clusters that allow interpretable topics and topic descriptions. [79] To better understand the Topic Modeling application of BERT that will be applied in this research, one must first understand some aspects of the BERT language model. Therefore, the base model of BERT is first explained. Then, the architecture of BERTopic is presented and the four techniques that make up the Topic model.

BERT base model The language model BERT was introduced by Google in 2018 as an open-source machine learning framework for natural language processing (NLP). It is designed for a better understanding of ambiguous language in texts by using surrounding words for establishing context. BERT stands for Bidirectional Encoder Representation from Transformers as it is based on the deep learning model Transformers. [46] Breaking down the name of BERT, bidirectional implies the strength that both the context words before and after the word are scanned and recognized simultaneously. BERT creates deep bidirectional representations, in which the information is learned both left to right and right to left. Models that learn bidirectional are considered more powerful than unidirectional models.[69] An encoder is explained as a process that can convert data from one format to another. In BERT, the encoder allows the model to encode certain information into the embedding. This information can be the semantics and syntax. As an encoder, the model uses the encoder layer of the Transformer model.

The Transformer model is a neural network consisting of six layers of encoders and decoders. Decoders are viewed as the opposite of an encoder. A decoder decodes the word embedding and information retrieved by the encoder back to a vector that can be represented again as text. Each encoder in the Transformer model consists of two sublayers, a multi-head attention layer, and a feed-forward neural network, while each decoder consists of three sub-layers, a multi-head attention layer, a second multi-head attention layer for understanding the encoder input and also a feed-forward neural network. [78]

The mechanism that allows the encoder and decoder to understand the text in context is called attention. Attention to a word is based on the softmax function of the query vector Q and key vector K and multiplication by the value vector V. This leads to Equation 3.7.

The "multi-headed" attention layer could be seen as calculating attention from different angles. The angles are interpreted as how a text can be interpreted. This allows BERT to understand the context in both directions, before and after reading a word, bidirectionally. The difference in architecture between the Transformer model and BERT is that in BERT, only the encoder layers are used. As explained above, through the encoder step, BERT understands the text in its context, which can be used for all different natural language processing tasks, such as Topic Modeling.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{3.7}$$

The multi-head Attention is calculated by using the formula 8 times with different matrices. For clarification, the module for attention runs through the mechanism multiple times in parallel. These independent outputs for attention are concatenated and transformed into a linearly expected dimension. The formulas derived from this process are shown as Equation 3.8.

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(head_1, ..., head_h)W^O \\ head_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \\ &\text{Where } W_i^Q \in \text{R}^{d_{model}} \times d_k \end{aligned} \tag{3.8}$$

The language model BERT has been pre-trained on a vast corpus of unlabeled text. This includes the entire Wikipedia, which is 2,500 million words, and Book Corpus, which are over 800 million words long. Half of BERT's success can be attributed to this pre-training phase. That is because, as the model is trained on a big text corpus, it begins to pick up on the more subtle and personal details of how the language works. This information can be applied to a wide variety of NLP tasks, such as Topic modeling in our research. [62]

Architecture The architecture of BERTopic is a combination of four methods. First, the embeddings are created with the use of SBERT, sentence-BERT. Second, the embeddings dimensionality is reduced with UMAP, Uniform Manifold Approximation and Projection. Third, the clustering of the document embeddings is done using HDBSCAN, hierarchical density-based spatial clustering of applications with noise. At last, the topic representation and calculation of the most important terms per cluster is done with c-TF-IDF, a class-based TF-IDF measure. [87]

Sentence-BERT SBERT, short for Sentence-BERT is first presented in 2019 by Reimers and Gurevych. It is a modification of the pretrained BERT model and is used for deriving semantically meaningful sentence embeddings that can be compared using cosine-similarity. [60] In BERTopic, the documents are embedded for semantically comparing their representations in vector space. Documents that are semantically similar are expected to contain the same topic.[87] The performance of SBERT is tested on various sentence embedding tasks and the results are promising. [72] SBERT is especially considered applicable on tasks that computationally are not feasible with BERT. Where BERT takes around 65 hours to cluster 10,000 sentences with hierarchical clustering, because 50 Million sentence combinations have to be computed. SBERT is able to reduce this to about 5 second without losing accuracy. [60]

UMAP The embeddings created with the Sentence-BERT method are very high dimensionality. When the dimensionality increases, the distance to the nearest embedding approaches the distance of the farthest embedding. [5, 7] This means that in high dimensional space, using distance measures for comparing is harder. Therefore, a solution is to reduce the dimensionality of the text embeddings. BERTopic uses UMAP, Uniform Manifold Approximation and Projection, for this dimensionality reduction. UMAP is chosen over other dimensionality reduction methods, like PCA and t-SNE as it should preserve more features when high-dimensional data is projected in lower dimensions. [51] UMAP is able to achieve this by three significant steps. First, local manifold approximations are calculated. A manifold is signified by a topological space that is locally Euclidean, meaning that it resembles Euclidean space near each point. Second, a fuzzy simplicial set representation of the local manifold approximations can be constructed. A simplicial set representation is a higher-dimensional generalization of elements, such as a catagory,

directed graph and partially ordered set. Third, the layout of the representation is low dimensional space is optimized by minimizing the cross-entropy C between the fuzzy topological representations. A fuzzy representation is given by set A and a strength function $\mu : A \to [0, 1]$. The formula for the cross-entropy C of two fuzzy sets (A, μ) and (A, ν) is shown in Equation 3.9. [51]

$$C((A, \mu), (A, \nu)) \triangleq \sum_{a \in A} \left(\mu(a) \log \left(\frac{\mu(a)}{\nu(a)} \right) + (1 - \mu(a)) \log \left(\frac{1 - \mu(a)}{1 - \nu(a)} \right) \right) \tag{3.9}$$

HDBSCAN The technique used for clustering of these reduced embeddings is called HDBSCAN, hierarchical density-based spatial clustering of applications with noise. [24, 33] HDBSCAN find a clustering by integrating the results of the DBSCAN over varying values of epsilon and checks for the best stability over epsilon. [43] The goal of the algorithm is to find dense clusters, the representation of the algoirthm is the following. The algorithm attempts to find so-called islands of higher density amid a sea of sparser noise. Here, the assumption of noise is important because real data can be messy and has outliers, corrupt data, and noise.

The HDBSCAN algorithm deals with this principle in the following steps. [43] First, the space is transformed, and for all data points the core distance is defined as $\text{core}_k(x)$. This is the core distance of data point x to reach k other data points. The core distance is considered the radius of a circle, and the size is linked to how big the core of the circle must be to include k other data points. This measure calculates the density of a data point as if the core is small, a lot of data points surround data point x and it has a high density. The mutual reachability distance is calculated to ensure that data points with a high core distance are pushed away to be at least their core distance away from any other point. The mutual reachability formula is shown below as Equation 3.10 with $d(a, b)$ being the distance metric. [43] In the case that the core distance of a data point is larger than the original distance metric, this is perceived as the new distance between data points. It is shown that the mutual reachability distance improves single linkage clustering in better approximating hierarchy level sets of the true density distribution of which our points are sampled. [34]

$$d_{\text{mreach } -k}(a, b) = \max \{\text{core}_k(a), \text{core}_k(b), d(a, b)\} \tag{3.10}$$

Second, the minimum spanning tree is constructed. The data is considered as a weighted graph with all data points as vertices and the weight of the edges between two points being the mutual reachability distance between them. The tree is built one edge at the time, starting with the edge that has the lowest weight to connect the current tree with a new vertex.

Third, the minimum spanning tree is converted into the cluster hierarchy. The edges of the minimum spanning tree are sorted increasingly by distance and by iterating over the edges a new merged cluster is created. The difficulty lays in identifying the edge that connects two clusters, which is done by a union-find data structure. A union-find data structure implies a structure that observes the set of elements that are partitioned into a number of disjoint subsets.

Fourth, the hierarchical representation of all vertices is condensed into a smaller tree with more data attached to each node as each node represent a set of vertices. To create this condensed, smaller tree, each split is analyzed. As the model considers the hyperparameter of minimum cluster size, for each cluster split it is considered whether both parts after the split are above this minimum. A split between two sets of vertices is mainly based on two ways. The first being one or two points splitting off from the cluster, which are considered 'losing points'. The second, called a true cluster split, if a split creates two valid separate clusters above the minimum cluster size. For the entire hierarchical representation, the splits are examined and the all points that are 'lost' in the process are added to the noise cluster with the distance metric mentioned.

The last step of the process is the extraction of the actual clusters. A persistence measure λ is calculated for each cluster, $\lambda_{cluster} = \frac{1}{\text{distance}}$. For each cluster or set of vertices, λ_{birth} can be computed for the moment that the cluster is split off and became a cluster or a λ_{death} when it split into smaller clusters. Besides, for each data point p in a cluster, λ_p can be computed at which it got 'lost'. This λ_p is in range [λ_{birth},λ_{death}] as it happened in the lifetime of the cluster or at the end when the cluster is split into two new clusters. Now, the stability of the cluster is calculated by the Equation 3.11.

$$\sum_{p \in \text{cluster}} (\lambda_p - \lambda_{\text{birth}}) \tag{3.11}$$

After calculation of the stability for each cluster, start from the leaf nodes and check the clusters stability. When the sum of stabilities of the separate clusters is greater than the stability of the merged cluster, the set the cluster stability to be the sum of the separate stabilities. However, if the merged cluster stability is greater than the sum of separate stabilities, the merged cluster is considered a new cluster and the separate clusters are unselected. After going through the entire tree, the final clustering is presented of all remaining clusters.

c-TF-IDF A class-based procedure of the TF-IDF formula is adopted for multiple classes by joining all documents per class. Instead of using the frequency f of words t, which is per document, the frequency is taken per class $tf_{x,c}$. Each class is converted to a single document instead of a set of documents. The formula becomes the following:

$$W_{x,c} = tf_{x,c} \times \log(1 + \frac{A}{f_x}) \tag{3.12}$$

with $tf_{x,c}$ = frequency of word x in class c
f_x = frequency of word x across all classes
A = average number of words per class

The difference with the TF-IDF formula is that all documents in one class have the same 'class vector'. This means that it needs to be looked at from the class-based point of view instead of from individual documents.

3.3.4 Validating the Topic Models

Validating unsupervised machine learning techniques is not only about measuring performance, but also about understanding the model at a deeper level. [57] Therefore, the evaluation metrics chosen play a fundamental role in the research and the purpose of the study. In unsupervised learning, the problem that validation encounters is that there is no ground truth. In the absence of labels, it is impossible to calculate which documents are correctly clustered. However, there are two classes of statistical techniques that are used for validation of the results of clustering: Internal and External validation.[68, 88]

The first, internal validation, includes all metrics that take into account cluster density or cohesion within clusters. These metrics are based on all documents appearing in the same cluster, and true labels or a ground truth are not required. The second class, external validation, is based on documents clustered in the true clusters. This kind of metric requires ground truth, so this kind of metric is not applicable in this study. The internal metrics of interest for comparison and on which the study is based are metrics based on cohesion and separation. Cohesion is calculated by summing the similarity between all pairs of documents within a single cluster. Separation is calculated by summing the distances between pairs of records that are both in a different cluster. Logically, cohesion is minimized because it tries to be highly coherent, while separation is maximized to ensure clusters are distinct.

$$\text{Cohesion } (C_k) = \sum_{x \in C_k; y \in C_k} \text{similarity(x, y)}, \qquad \text{Separation } (C_j, C_k) = \sum_{x \in C_j; y \in C_k} \text{distance(x, y)}$$

The cohesion of the clusters is calculated with topic coherence measures. Several types exist, but the Cv coherence, the u_{mass}, the c_{uci}, and the c_{npmi} are considered the most relevant for this study. [85]

The coherence score Cv is the most commonly used score to measure coherence. Cv refers to the Context vectors created by using the co-occurrences of words. [45] Context vectors are used to calculate cosine similarity and are created through NPMI, which stands for normalized pointwise mutual information. The context vectors are created by linking the vectors of both words to other words in the sliding window. The pairing is calculated via the NPMI, see Equation 3.13. NPMI is calculated by dividing the probability of both words appearing together by the product of the appearance of one word and the appearance of the other word. This number is equal to one if the words always appear together. Normalization of the formula is done by dividing the whole fraction by the logarithm of both words together plus a small ϵ, which indicates whether the logarithm is zero. The γ is added to increase the weight on higher values for NPMI. The formula of NPMI is shown in Equation 3.14. The context vectors of both words are combined in the confirmation measure θ by calculating the cosine vector similarity, see Equation 3.15. [45]

$$\vec{v}\left(W'\right) = \left\{ \sum_{w_i \in W'} NPMI\left(w_i, w_j\right)^{\gamma} \right\}_{j=1,\ldots,|W|} \tag{3.13}$$

$$NPMI\left(w_i, w_j\right)^{\gamma} = \left(\frac{\log \frac{P(w_i, w_j)+\epsilon}{P(w_i)-P(w_j)}}{-\log\left(P\left(w_i, w_j\right)+\epsilon\right)} \right)^{\gamma} \tag{3.14}$$

$$\phi_{S_i}(\vec{u}, \vec{w}) = \frac{\sum_{i=1}^{|W|} u_i \cdot w_i}{\|\vec{u}\|_2 \cdot \|\vec{w}\|_2} \tag{3.15}$$

The u_{mass} coherence score is determined by calculating the number of times two words appear together in the texts. The formula is as follows:

$$C_{Umass}(w_i, w_j) = \log \frac{P(w_i, w_j) + \epsilon}{P(w_i)} \tag{3.16}$$

The formula shows the fraction of how often the two words occur together in the documents $D(w_i, w_j)$ divided by how often one of the two words occurs alone $D(w_i)$. The metric is not symmetric because it is divided by the number of times one of the two words appears. Another score is created for the other word of the two. The coherence of an entire topic is determined as the average pairwise coherence of the top k words with the highest probability of describing the topic.

The c_{uci} score is determined by the co-occurrence within sliding windows and the pointwise mutual information for all word pairs appearing in the top k words. The co-occurrence of two words in the text depends on whether they also appear in the sliding window. Word pairs that appear in the same text, but not in the sliding window are not interpreted as co-occurring. The formula is shown below:

$$C_{UCI}\left(w_i, w_j\right) = \log \frac{P\left(w_i, w_j\right) + \epsilon}{P\left(w_i\right) \cdot P\left(w_j\right)} \tag{3.17}$$

The probability of two words occurring in each other's sliding window $P(w_i, w_j)$ is divided by the product of both words appearing in the sliding window. The topic coherence is again determined by averaging the pairwise coherences of the top k words per topic.

According to the creator of these coherence measures Röder, the greatest correlation to human topic coherence assessment is obtained when using the C_{NPMI}. [36] For this measure, the elements of the vectors are defined via NPMI, which is also used inside the first discussed coherence metric Cv. If two words are observed together, the probability of observing one is equal to the probability of observing the other, which is equal to the probability of seeing them together. The logarithm of the probability of both words is chosen for normalization, because it normalizes both the upper and lower bounds. The difference with the coherence metric Cv is that now only the NPMI between both words is calculated and not within a context vector, for which the cosine similarity is calculated. However, the formula for NPMI is the same, but without the weight of γ and can be though of as Equation 3.18. [14]

$$C_{NPMI}\left(w_i, w_j\right) = \frac{\log \frac{P(w_i, w_j)+\epsilon}{P(w_i)P(w_j)}}{-\log\left(P\left(w_i, w_j\right)+\epsilon\right)} \tag{3.18}$$

Chapter 4

Data

This chapter explains all the steps taken that relate to the data collection, exploration and model preparation. First, the data collection steps are discussed in the Text scraping section, since all data used for this study are scraped from the Web. Second, the data exploration section outlines all the steps taken to explore the potential of the dataset obtained. In addition, a simple data analysis is performed on the dataset and it is discussed why this dataset is a good representation of the real problem this thesis attempts to solve. Finally, the data preparation steps needed for topic models are discussed in data preprocessing section and examples of pieces of text are provided. For this study, an open access dataset was extracted from IEEEDataPort, which contains 1700 articles on the topic of "Edge Computing". The dataset contains a list of articles indexed by Google Scholar from 2003 until 2019 and can be found under the name of "Dataset on Edge Computing EdgeComputingClean.csv". Therefore, this list of articles is considered very similar to an actual search on Google Scholar on the topic "Edge Computing". It fits the description for a test case for our problem. The data in the dataset is used to scrape the article texts needed to apply the topic models, and the texts are added to the dataset used for the rest of the study.

4.1 Text scraping

For text scraping, a common method is scraping via the use of a URL. A python library built to extract data from HTML and XML files is BeautifulSoup. It works using a parser to provide idiomatic ways to navigate, search, and modify the parse tree, resulting in an HTML file. HTML files are known to have a clear structure, allowing the desired text to be extracted from this file.

4.1.1 Web scraping

The web scraping is done in python and in the following steps. First, an empty definition is created that takes the URL of an article and outputs the full text of the URL web page. Second, the 'request' library is imported and used in the definition to extract the HTML source code of the page. Third, the BeautifulSoup algorithm is executed to use the HTML parser and understand the structure of the HTML scripts. Fourth, the output of BeautifulSoup contains all the information about the web page structured in the HTML script. The output of the BeautifulSoup includes the full text of the web page, which functions as the output of our definition. As it is now in this BeautifulSoup format, there are several functions that can extract specific parts of the script. The functions 'find' and 'find_all' are easy to use for selecting classes or groups in the HTML script. However, functions such as 'body' and 'get_text', are built in for extracting the entire body or all found text in the HTML script.

Since the dataset provided the URL of 1631 articles, these can be used to extract the appropriate information using BeautifulSoup. However, many publishers do not allow you to randomly scrape text from the articles they publish. These texts can be accessed, but different rights and rules apply to text scraping. Some publishers do not allow scraping of text without requesting an API, some allow no kind of text scraping at all, some allow only small portions of the text, and some allow full-text scraping. For this research and the use of Topic Modeling techniques, small amounts of text per article are sufficient to apply Topic Modeling. Because of all these different permissions, rules and structures, it is necessary to build a specific script for each publisher. A custom scraper is created only for those publishers who published more than 5 articles. This amounted to 19 publishers.

Full-text scraping is only allowed for 2 of the 19 publishers because of paid subscription that many publishers use. However, because the publishers want to show the articles that can be found on their platform, 12 out of 19 publishers have open access to the abstract. Therefore, it is decided to base this study on the abstracts of these 12 publishers. These abstracts are scraped in python with different scrapers using the BeautifulSoup algorithm for extracting the HTML scripts, which yielded a number of 1228 abstracts. The number of abstracts per publisher for the seven major publishers in this research can be seen in Table 4.1. In addition, abstracts can be considered the most important text for Topic Modeling because it contains the most important words of a text and should contain the essence of the full article. [45] The topic models and methods used in the rest of this research are applied to the text of the abstracts. The abstracts consist of different lengths, and it should be kept in mind that text scraping from each publisher requires a different script.

Publisher	**# articles in dataset**
ieeexplore.ieee.org	850
Springer	144
dl.acm.org	91
Elsevier	80
mdpi.com	41
arxiv.org	41
Google patents	29

Table 4.1: Major publishers in dataset

4.1.2 Abstract scraping

The problem with text scraping is that the format of the HTML script is different for each publisher. This means that each publisher requires a specific script to extract the correct data from the web page. As noted in the previous section, the abstracts for 1266 articles are open access. Therefore, the output of the created definition that extracts the HTML script of the web page is used to find and extract only the text of the abstracts of the articles. [45] After finding the abstract in the HTML script, the retrieval steps are done. The search for the abstract class or abstract text in the HTML script is done manually in this study, because the specific name or tag that the abstract has is different for each publisher. The abstract appears in many different places in the parse tree that is created by BeautifulSoup. In some cases, the abstract is placed in the metadata and sometimes in classes grouped by $< div >$ elements. In the latter case, the abstract class is easily accessible because one only needs to use the 'find_all' function with the specific class name to locate and extract the abstract class. If the abstract is located in the metadata, the last and first part of the structure where the abstract is located must be found

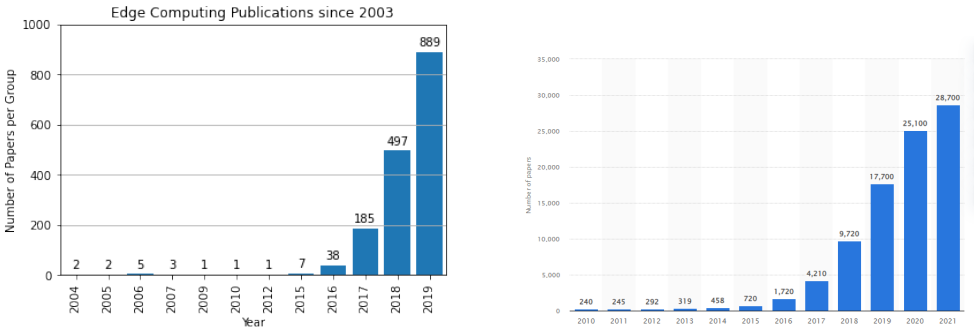
and only the text of the abstract must be isolated. After the abstract class or text is extracted, if present, parentheses and other unimportant terms are removed from the text. In HTML scripts, numerous additional parentheses are placed in the script to structure the document.

4.2 Data exploration

This research is based on the results of a Google Scholar search. The results are the papers presented by Google Scholar. To replicate this example, the dataset used to study Topic modeling techniques is also based on the results of a search. This section describes the dataset in more detail and discusses the methods used for simple data analysis and exploration on the original dataset.

The dataset consists of various descriptive information from 1,700 articles that are all resulting papers for the search query 'Edge computing'. This is only the test case of this study, and the topic itself does not affect the topic models that will be used and trained to expedite a consultant's research phase. However, because all papers contain the same general topic, topic models may have difficulty finding underlying topics in the corpus. All articles in the dataset were published between 2003 and 2019. Descriptive information about the articles includes data, such as the number of cites, authors, title, year written, source, publisher, article URL, URL to cite, and GS rank of each article.

Since all articles were published within the period 2003-2019, the spread of publications over the years is examined. The distribution of the number of articles published has increased almost exponentially over this period. In Figure 4.1 (a) the articles within the dataset are presented, and a comparison with the increase according to Statista on the right (b) shows that it is a good approximation of the actual exponential increase over the years. In fact, according to Statista, in 2021 there have been published around 28,000 articles on the topic of 'Edge computing'. However, the dataset used in this research only includes articles from 2003 trough 2019. Nevertheless, this exponential growth substantiates the relevance of the topic in recent years. Moreover, with 28,000 newly published articles a year, it becomes increasingly difficult for researchers to conduct efficient research on the topic, demonstrating the need for an improvement in the current method of finding the right documents. [90]



(a) Articles inside the dataset (b) articles according to Statista

Figure 4.1: Distribution published articles over years

To elaborate on the dataset used for this study, some statistics are explained in Table 4.2. The table shows that the articles were published by a total of 188 different publishers, the most important of which can be seen in Table 4.1. These publishers used 930 different sources, with the largest sources being from the IEEE platform, such as 'IEEE Access', 'IEEE Transactions on ...', as the dataset comes from IEEEDataPort. The 1,700 articles were written by 3,597 different

authors, which is more than the number of articles, as some articles have up to eight authors. In addition, some authors published numerous articles. The three most publishing authors in the dataset are J. Wang, Y. Zhang, Y. Li with 25, 23 and 21 published articles, respectively, see Table 4.3. Furthermore, the most cited articles are shown in Table 4.4. Interestingly, two of the three papers were published in 2018 and are already so frequently cited in the early 2020s. For comparison on the recent exponential interest in the topic, this most cited article has been cited 1270 times today.

Variable	# unique instances
Publishers	188
Sources	930
Authors	3,597

Table 4.2: Number of different instances

Author	# publications
J. Wang	25
Y. Zhang	23
Y. Li	21

Table 4.3: Major authors

Title of the article	Authors	Year	# citations
'Learning IoT in edge: Deep learning for Internet of Things with edge computing'	H. Li, K. Ota, M. Dong	2018	356
'Authenticating query results in edge computing'	HH. Pang, KL. Tan	2004	195
'Edge computing technologies for Internet of Things: a primer'	Y. Ai, M. Peng, K. Zhang	2018	183

Table 4.4: Most cited articles

An important detail of each dataset is the number of missing values for key variables. In this case, the key variable is the URL of the article, since it is used to access the text of an article. This is important for the text scraping part of the research, as explained in the previous section. Only 69 of the 1700 articles have no article URL, which is about 4%, so this is not a significant number and these will be removed from the dataset as the text cannot be extracted.

When exploring the scraped abstract texts, interesting insights also arise based on the occurrence of words. In Figure 4.2, the ten most common words in the dataset are presented. This includes all words occurring in the dataset except all English stop words. The English stop words removed from the abstracts can be found in an open-source research package. [52] These words were removed to provide a clearer understanding of the meaningful words in the dataset, as language contains many words that by themselves do not provide information about the subject of the text. Since only word frequency is considered in this case, it is better to exclude these stop words.

Another interesting aspect of abstracts is the amount of words the texts consist of. An abstract can be a very short explanation of the article in a few sentences, or it can be a long paragraph of text about all the different parts discussed in the article. The length of the abstract texts was also examined, and the results ranged from 118 and 3255 words in the abstracts, before pruning the stop words. The distribution across the dataset is shown in Figure 4.3.

4.3 Data preprocessing

In any data-based research, the data used for the techniques must be properly prepared. All steps to prepare the abstract data for the use of the Topic Modeling techniques are explained in the next section.

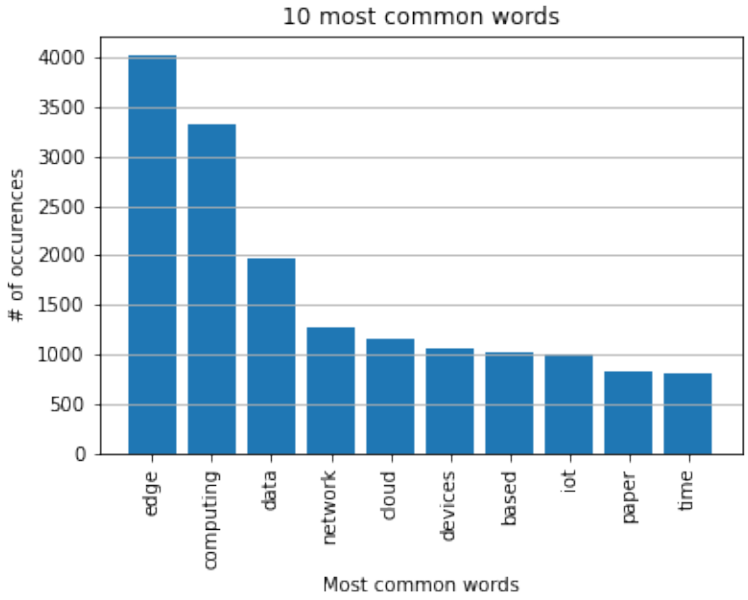


Figure 4.2: Most common words in the dataset

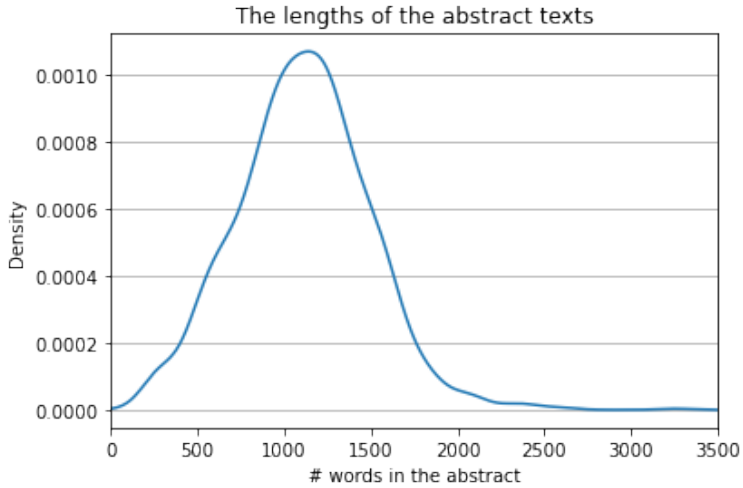


Figure 4.3: The density function of the lengths of the abstracts

The scraped abstracts consist of the full abstract displayed on the publishers' web page. In order for topic models to use this text, however, certain changes must be made. The step of removing or replacing irrelevant text in the abstract is considered an important part of the preprocessing, or also called cleaning up the text. Examples of changes made include converting to lowercase, removing punctuation, removing common stop words, expanding abbreviations, and word stemming and lemmatization.

Below is an example of part of an abstract. This will go through the entire preprocessing to give a clear understanding of how the abstract is actually formatted.

"Sensor networks are now one of the key technologies for the Internet of Things (IoT). An IP-based sensor network allows a natural participation of sensor nodes to the IoT. Smart IoT systems are always constrained by real-time processing. Edge computing can be used to address this constraint."

4.3.1 Abbreviation expansion

Sometimes common words or phrases within the topic are indicated by their abbreviations. Expanding these common abbreviations gives a clearer image of which words appear frequently in the text, rather than seeing them as two different words. Based on the literature, the remaining abbreviations in the text are left unchanged. It is simply considered another word, but since the abbreviation occurs normally in all texts, this will not cause distortion. Abbreviation expansion must be done before other preprocessing steps, since the letters of an abbreviation can also appear in other words. Therefore, abbreviation expansion is done first before other preprocessing steps. For example, the common abbreviation in these texts is EC, which stands for 'Edge Computing'. However, the word 'technology' also contains the letter 'ec'. If the text is lowercased first, the word 'tedge computinghnology' would be created. A list of all the abbreviations that are expanded is shown in the Appendix.

The expansion of the abbreviations is shown using the example and yields the following:

"Sensor networks are now one of the key technologies for the internet-of-things (internet-of-things). An IP-based sensor network allows a natural participation of sensor nodes to the internet-of-things. Smart internet-of-things systems are always constrained by real-time processing. Edge computing can be used to address this constraint."

It can also be seen that 'Internet of Things' is changed to 'internet-of-things'. This was done to make the abbreviation and the phrase actually the same.

4.3.2 Cleaning text

Next, one of the easiest steps in cleaning up abstract text is to convert it to lowercase. This is necessary because the models will not recognize the same word if the first letter is uppercase, for example, because it is in the beginning of a sentence. Next, sole numbers and punctuation are removed from the text. Punctuation would not play a role in topic models, but when words are tested, numbers also do not add value to the text because they are not connected to an informative noun. For example, '5G' is not removed from the text because it is not just numbers. Finally, irrelevant words, such as words that occur too often or do not contain information about the topic, are removed from the text. Examples of such words are articles, prepositions and conjunctions. Numerous different algorithms, word lists or libraries can be used for this purpose, all with similar but different results. Three commonly used methods are NLTK, Gensim, and SpaCy. In this research, the Gensim library will be used because it contains the largest amount of stop words (i.e. 337). [73] All removed stop words can be seen in the appendix.

The sample abstract is modified again and becomes the following:

"sensor networks key technologies internetofthings internetofthings ipbased sensor network allows natural participation sensor nodes internetofthings smart internetofthings systems constrained realtime processing edge computing address constraint"

4.3.3 Word stemming & Lemmatization

At this point, the text consists only of words that will be used in the topic model because of their connection to what the text is about. However, there are numerous forms of the same words. Therefore, the words must be modified to ensure that the topic model considers them equivalent. These adjustments are called word stemming or lemmatization.

Word stemming is the process of creating morphological variants of a base word. Nouns can be singular or plural, 'server' or 'servers', so a stemming algorithm would reduce both words to 'server'. For verbs, 'compute', 'computing', and 'computed' are all reduced to 'comput'.

Lemmatization is similar to word stemming and is the process of combining the different inflected forms of a word. Unlike stemming, lemmatization brings context to the words by linking words if they have a similar meaning. Therefore, it is often preferred to stemming because lemmatization does morphological analysis of the words. For lemmatizing a verb like 'compute', 'computing', and 'computed', the result would be 'compute'. Notice that this is the root form of the verb, instead of the word where the affix is just removed, which would be the case for word stemming.

Stemming does not result in a dictionary word because the affix is removed based on rules. The result after lemmatization is a dictionary word because it is reduced to a root form of the word. Text preprocessing involves both techniques and often the best way is to lemmatize the text first and then stem it, which was done in this study. Two examples of lemmatizing or stemming words are given in Table 4.5. In addition, the power of combining the techniques is shown. Whether two words are 'quick', 'quickly' and 'quicker', the meaning of the three words are the same. However, using only stemming or lemmatization will not produce in the same word, but combining the two will.

Word	**Lemmatization**	**Stemming**	**First lemmatization, then stemming**
quickly	quickly	quick	quick
quicker	quick	quicker	quick
manufactures	manufacture	manufactur	manufactur
manufacturing	manufacturing	manufactur	manufactur

Table 4.5: The combination of Lemmatization and stemming

The example is used again to clarify the method on real data. It is observed that many words are lemmatized and stemmed to the simplest forms, making it even more difficult for a human to understand the preprocessed texts. For a computer, however, many more words are interpreted as equivalent. See the example below:

"sensor network key technolog internetofth internetofth ipbas sensor network allow natur particip sensor node internetofth smart internetofth system constrain realtim process edg comput address constraint"

Chapter 5

Methodology

In this section, the research design is described and the methods used are explained. All information about how the method works is explained in Chapter 3 and its application to our data can be found in this section. First, the most commonly used method for Topic Modeling will be discussed, Latent Dirichlet Allocation. In addition, the method based on the TF-IDF measure is discussed, and how the clusters are created using K-means for this method. Then, the method based on the Word2Vec model is explained. Finally, the experimental design using the language model, BERT, in the BERTopic model is described.

5.1 Input preparation

Several preparation steps are required for Topic models to perform preferentially. This section illustrates some common input preparation steps needed for multiple models. In particular, input preparation for the first three methods, LDA, TF-IDF, and Word2Vec, is necessary to make the models work. Examples of input preparation include tokenization of words and documents or the creation of word IDs to track different words in the documents. An important note, the preparations are done on the already preprocessed web scraped data discussed in Chapter 4.

5.1.1 Tokenization

In simple terms for this research, the process of tokenization implies that each string is cut up into tokens representing the words. Each word is associated with a unique token. Tokens are deterministic, meaning that the same token is always obtained for a given value. Therefore, a tokenized database can be easily searched by tokenizing the terms of a query and searching for them individually. This feature is used in finding words, keywords, symbols, and even phrases that appear in the document data set. The tokens in this study are mainly separated by white space and represent the words in the text documents. The tokens are not only used for model input. Moreover, the tokens are used to represent the important words within a cluster in the output. [4]

5.1.2 Number of topics

A fairly crucial input hyperparameter needed for Topic modeling methods is the number of topics. This variable depends on the amount, the distribution, and type of data, as well as which Topic model is applied. Finding the optimal number of topics to use is covered by an entirely different area of research and is too broad to delve into specifically in this study. There are simpler methods for approaching the optimal number of topics for a model: The elbow method. The elbow method is considered a heuristic used in determining the number of clusters in a data set. The graphical meaning is a point at which the graph is almost horizontal, after a strong decline. This graph is caused by the average distortion decreasing as the number of topics

increases. There will fewer constituent instances in each cluster, but the instances will be closer to their respective centers. However, the change in mean distortion decreases as the number of topics increases, making the graph horizontal. This creates the 'elbow'. [58]

The boundaries for the number of topics that are taken into account for all models are 4 and 40 and the evaluation metrics are measured with a step size of 4. This ensures that the topic models do need to make distinctions between the documents, while not creating an unrealistic amount of topics as texts are always quite dispersed.

5.2 Evaluation metrics

This study evaluates the performance of the methods using two distinct measures, Topic coherence, and Topic Diversity, and finally the human interpretability of the topic labels. The measures used are explained in detail in Chapter 3. The application of the method is discussed in this section.

For topic coherence, multiple metrics are examined and their combination is compared, since metrics by themselves can be misleading. Coherence scores measure how interpretable topics are to people. Topics are represented by the top k words of that cluster, which are the words that have the highest probability of fitting into that particular topic. The similarity between the top k words is measured by the coherence score metric. The metrics are the coherence Cv, u_{mass}, c_{uci} and c_{nmpi}. These metrics are chosen in combination because it helps interpret the results.

The problem that the evaluation metrics encounter is their dependence on the data. Cluster validation is considered difficult in almost every research, with a large subjective factor combined with the cohesion or separation between data that can be quite different. In the case of document comparison, the degree of difficulty is increased as texts are very complex data. Therefore, interpreting how good the validation metrics are is perceived as difficult. Some texts will have much higher coherence and diversity metrics than others, but for comparison of models based on the same data, the metrics can be used. Because of the great variety within texts that can be clustered in the same topic cluster, high metrics are rarely the result on the same topic. In the literature, good rating are indicated by high correlation with human ratings, which are not available in this study. The proposed solution involves the comparison between the different methods, since the better method will get a higher coherence score on the same data. Moreover, the value of the results is compared based on other text clustering by Topic modeling papers. [36] The perspective, created by similar papers, is that the coherence measures are represented within a paricular window. The windows of the Cv score are reported in Table 5.1.

C_v score window	≤ 0.3	0.4	0.5	0.6	0.7	0.8	≥ 0.9
Interpretation	Bad	Low	Average	Good	Very good	Unlikely	Probably wrong

Table 5.1: The Cv coherence score perspective

The U_{mass} score is generally a bit smaller and can deviate sharply to negative numbers when the probability of a word occurring is small. In this study, a score closer to 0 is considered better. The c_{uci} score also approaches 0, but can also diverge because it is not normalized. Higher scores of this metric indicate more co-occurrence between the main topic words, which is preferred in this paper. Normalization is included in the c_{npmi} which ensures it cannot diverge too much, and this coherence metric therefore falls between -1 and 1. A c_{npmi} score of -1 is interpreted if words never occur together, 0 if the words are independent, and closer to 1 if words occur together more often. [42] Therefore, values close to 0 are expected because the number of words is generally independent of each another. Results closer to 1 are interpreted as better results.

For Topic diversity, the default metric is chosen, which is calculated from the top k words and controls the diversity between the top k words of the clusters. This indicates the distinction of the clusters. At a high topic diversity, approaching 1, the model has correctly detected the different components within the search area, while a low topic diversity, closer to 0, means that the components are more scrambled among the topics. In this study, topic diversity is expected to be quite low since the entire dataset is on the same topic. However, the comparison provides an interesting additional perspective beyond measures of topic coherence.

The final method of determining the value the models can add to research is the human interpretability of the subjects. This measure is based on the labels created, which are determined by the keywords of each cluster. High human interpretability is observed for topic labels that contain very human-understandable connections and similarities. It should be noted that this measure is subjective. The labeled keywords of the clusters are kept in their lemmatized and stemmed form so as not to imply the words from which they originate. Yet this is not considered worse for comprehension, as this would imply that lemmatization and stemming are worse for human interpretability.

5.3 Topic Models

During the research phase of this study, a wide range of methods were examined. However, for clarification purposes, the most interesting ones are discussed in this section. In Chapter 3, the models considered for this research are LDA, NMF, TF-IDF, Word2Vec, and BERTopic. However, during the implementation phase of the study, it was found that the results of NMF were not significant and therefore this method was not tested further. This section details the methods used in this study.

5.3.1 LDA

The first method applied is the best known for Topic Modeling, Latent Dirichlet Allocation. Chapter 3 describes the characteristics of the method and numerous applications of this technique are available for use. LDA is a generative model, so the application of an LDA model replicates the mechanism of creating the output rather than the output itself. [63] The output is formed by the topic clusters in our study. Training an LDA model requires three ingredients: the corpus, the number of topics, and a mapping of word IDs to words. The corpus consists of the Term Document Frequency, that is, how much each word appears in each document. The number of topics is determined for each model based on the elbow method or maximum topic coherence. The mapping of word Ids to words is specific for the LDA method, since in the corpus all words are represented as a word ID. This mapping of word IDs is explained further in the subsection below. After that, both the implementation and the hyperparameter tuning are addressed. [15]

Mapping word IDs

The mapping of IDs to words is required for the input of the LDA model. Therefore, every token that is created by the tokenization step also has a unique id that can be searched. This ID is created alphabetically per document and is stored inside a word dictionary. To elaborate, the first alphabetically ranked word of the second document that has not occurred yet is given the first available ID. An example: the word 'ability' is the first word of document 2 if it was ranked alphabetically. In the situation that document 1 has 46 unique words, also alphabetically ranked, the word 'ability' is given an ID number of 47. The mapping of word IDs to words is needed for determining the vocabulary size, debugging and topic printing as the ID needs to be transformed back to the word.

Implementation

The tokenization required for inputting an LDA model is performed using the nltk package tokenized in Python and specifically with the function $word_tokenize$. To determine the number of requested topics, an approximation of the elbow method is used by looking at the change in coherence score. The number of topics is varied from 2 to 40 topics with a step size of 4. [49] In this study, the implementation of the LDA model is from the package OCTIS. [83] Optimizing and Comparing Topic models Is Simple, for short OCTIS, is an open-source evaluation framework built for comparing state-of-the-art topic models, which consists of multiple built-in models and capabilities for hyperparameter tuning via a Bayesian Optimization approach.

Hyperparameter tuning

The application of LDA in OCTIS provides additional hyperparameter tuning with the built-in 'optimizer' function. [83] Numerous hyperparameters exist for the LDA model, such as alpha, eta, as well as the aforementioned number of topics. Whereas adjusting the number of topics can have a large effect on the topic coherence score, the other hyperparameters have less influence. However, for both alpha and eta, a search space is created between about 0 till 5. The optimization function attempts to incrementally increase the evaluation score, which is chosen as the Cv topic coherence score.

5.3.2 TF-IDF clustering

The term frequency-inverse document frequency is not considered a complete topic model in the literature, as it is the statistic used to calculate the importance of a word in a document. However, this statistic is used for topic modeling, and it is interesting to look at the effect of using the TF-IDF measure to create topic clusters. The TF-IDF measure, shown in Chapter 3, allows the importance of each word in a paper to be calculated and compared to the occurrence and importance of that same word in another paper. A TF-IDF matrix is created that keeps track of all the TF-IDF features, which are then used for document clustering. The clustering of documents based on the TF-IDF matrix requires an unsupervised clustering method, for which K-means is well suited because it preserves the multidimensionality of all features of the matrix, so the results remain explicable because the similarity between documents is still based on the word importance. [21]

The TF-IDF matrix is calculated by multiplying the term frequency by inverse document frequency.

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

The n represents the number of documents in the collection, and $df(t)$ means the number of documents in the collection that contain the same word. The IDF term used differs from the textbook notation, which places the extra plus one that falls outside the fraction in the denominator. The document vocabulary contains words that have a very low document frequency, meaning that they do not appear in many other documents and are very specific to that document. In addition, it is common for words to have a very high document frequency, indicating that the words appear in almost every document in the document set. In the last category, words such as articles or pronouns are found. Both categories of words are considered irrelevant when using the TF-IDF statistic because a very low or very high score does not produce the desired result, as one causes documents to be considered non-matching and the other causes all documents to be similar, which would allow them to be grouped into a cluster. To solve this, when building vocabulary, a minimum and maximum is added to the document frequency of 0.5 and 0.95, respectively.

The TF-IDF vectors that are the result from multiplicating the remaining words are normalized using the Euclidean norm. The Euclidean norm or also referred to as the L^2 norm is used as it measures the shortest distance to the origin. This allows for easy comparison of vectors and fits well with the task of clustering document. The norm is defined as the root from the sum of the squares of the endpoints of the vector, see Equation 5.1.

$$\|\mathbf{x}\|_2 = \left(\sum_{i=1}^{N} |x_i|^2\right)^{1/2} = \sqrt{x_1^2 + x_2^2 + \ldots + x_N^2} \tag{5.1}$$

After normalization, the TF-IDF matrix is completed and all features of the model are represented in a document-term matrix.

Implementation

This implementation also uses the preprocessed abstract text data. The function 'TfidfVectorizer' from scikit-learn is used from the text feature extraction page. The function is a combination of the CountVectorizer which counts the occurrence of words per document and the TfidfTransformer which transforms the count vectors into a document-term matrix containing all TF-IDF scores. The resulting scores are used as features of the model and are used for clustering the documents. The clustering method used is the 'MiniBatchKMeans' method, which comes from the scikit-learn clustering package. The optimal number of topics or clusters is chosen by the same variation of the elbow method as for the LDA methods.

Hyperparameter tuning

Next to tuning the number of topics chosen, more hyperparameters can be tuned. The optimization process of the TF-IDF measure includes the tuning for the hyperparameters: n-grams, maximum document frequency and minimum document frequency. An n-gram is a contiguous sequence of n items from a given text sample. In the TF-IDF method, one can choose whether only unigrams, a one-word word sequence, or also multiple n-grams, word sequences of n words, should be considered. It is expected that only unigrams and possibly bigrams, word sequences of two words, will be considered. However, this parameter is considered during hyperparameter tuning. In addition, the minimum and maximum document frequency are hyperparameters that can be optimized. When a float, a number with a decimal, is entered for the minimum or maximum document frequency, it means the proportion of documents to which the word minimum or maximum should belong. However, when an integer, a number without a decimal point, is entered, it means the absolute frequency value that the word minimum or maximum should be part of. All hyperparameters are examined by incrementally trying different combinations to increase the Cv topic coherence score, but topic diversity is also taken into account.

5.3.3 Word2Vec clustering

The Word2Vec approach differs slightly from the previously mentioned methods in that it involves word embeddings. [28] The difference is that the context of the words is considered relevant for topic clustering. The Word2Vec approach is based on neural networks and a deep learning approach. The algorithm learns relationships between words by automatically traversing large amounts of unlabeled text. The method creates a word vector for each word including its relationship to other words, which is constructed from the context the words occur in. [9, 27]

The words are embedded in a lower-dimensional vector space using a shallow neural network. Words that appear close together in this vector space are assumed to have similar meanings. The word vectors, which are based on context and distance from each other, resemble each other.

The word vectors are created using one of two models, a Skip-gram model or a Continuous-bag-of-words model (CBOW), as also explained in the Chapter 3. [27] The first model, the Skip-gram, takes in pairs generated by a sliding window. In this study, the optimal size of the sliding window is tested by tuning the hyperparameter. Context should be taken into account, but the window should not be too large so that the model considers words that are not relevant to the target word. A hidden layer neural network is trained based on the input word and predicts the probability distribution of the surrounding words. Projection weights are created by a virtual one-hot encoding of the words passing through a projection layer to the hidden layer. The weights are interpreted as word embeddings. The number of neurons in the hidden layer creates the number of dimensional word embeddings. [27] The second model, the Continuous-bag-of-words model, is also a neural network with one hidden layer. However, the average of multiple input-context words is plugged into the model instead of the other way around for target word prediction. The word embeddings are formed by the projection weights, which are used to turn the one-word words into vectors with the same width as the hidden layer.

Implementation

To implement the Word2Vec model, Gensim's Word2Vec model is used. First, the preprocessed abstract texts are tokenized to split up all the different words. Next, the Word2Vec model is trained based on the abstract texts with adjustable parameters, which are discussed in the hyperparameter tuning section. A vectorize definition creates a vectorized form of each document by combining the trained model and the tokenized documents. Then these vectorized documents are clustered using the K-means algorithm, "MiniBatchKMeans" method. This results in the final clustering of the documents according to their document embeddings.

Hyperparameter tuning

The method based on the Word2Vec algorithm combined with K-means is first tested for the different number of subjects. Then other hyperparameters can be tuned to the model, such as minimum count, sliding window size, vector size, alpha and minimum alpha. The minimum count means that a word must have a minimum occurrence, otherwise the word is ignored. If a word does not occur more than this minimum count, it is not included and the internal dictionary is pruned. The size of the sliding window is the maximum distance between the target word and the predicted word. A sliding window of 2 means that the two words before and the two words after the target word are considered in the window. The number of dimensions on which gensim Word2Vec classifies the words is set with the vector size parameter. Finally, alpha is the initial learning rate, which decreases linearly to the minimum alpha parameter.

5.3.4 BERTopic

The language model BERT has been used for many different applications in recent years. When applications require techniques and solutions related to Topic modeling, the "cousin" of BERT, BERTopic, is used. This application of BERT is built specifically for Topic Modeling. Since BERT is a language model already trained on millions of words, the input texts of both BERT and BERTopic do not need to be preprocessed. Therefore, this study applies the model to the scraped abstract texts and not the preprocessed abstract texts. [87]

The topics are then created in three steps. First, the documents are transformed into embeddings by the pre-trained language model SBERT, sentence-BERT. Next, the dimensionality of the embeddings is reduced to improve the clustering with UMAP, Uniform Manifold Approximation and Projection. Finally, topic representations of the clusters are created based on all documents located within the cluster with HDBSCAN, hierarchical density-based spatial clustering of applications with noise. [87]

Documents containing the same topic are assumed to be semantically similar, which is based on the Sentence-BERT (SBERT) framework. [60] Rather than generating a word embedding for each term, SBERTopic generates a representation of the context to the left and right of the targeted text. The framework allows sentences, as well as paragraphs, to be converted into dense vector representations. Its performance is considered state-of-the-art on several embedding tasks. [72] Nevertheless, this framework is mainly used for semantic clustering of similar documents rather than producing topics. Before applying the document clustering technique, the UMAP method is used to reduce dimensionality. UMAP is known to preserve more features, both local and global, of high-dimensional data when projected into lower dimensions than more common methods, such as PCA and t-SNE. [51] After that, HDBSCAN is applied as a document clustering method, which is the hierarchical clustering variation of DBSCAN. [43] The results of this clustering are represented as a topic using the c-TF-IDF formula. This class-based TF-IDF score is further explained in Subsection 3.3.3 and calculates the most prominent words per class instead of per document to create a good representation of the important words in the cluster. The formula is given as Equation 3.12.

The clustering of the BERTopic model also shows the stop words because there is no preprocessing. This has an effect on both the topic diversity score and human understandability. The topic diversity score is extracted from the top ten most important words from the final clusters. However, due to no preprocessing in the BERTopic model, stop words also appear here. However, these stop words appear in multiple clusters as the most important, so the topic diversity of this method is quite low. The same problem occurs with human interpretability. Therefore, to create human-understandable topics, these are removed during the labeling of the clusters. This ensures that the nouns that actually contain information about the topic are clearly visible. The results are not processed further, only the removal of the stop words. Therefore, different forms of the words can still occur in the labels because no lemmatization or stemming is done.

Implementation

Maarten Grootendorst's algorithm BERTopic is used to implement this method. [87] This is already a combination of all the necessary techniques, such as the Sentence-BERT, UMAP, HDBSCAN and c-TF-IDF methods. So the BERTopic model is easily accessible, but modifying the model is a bit more difficult. However, all the different individual methods can also be implemented in the framework.

Hyperparameter tuning

As the BERTopic model used for the implementation is a combination of the four methods mentioned above, many hyperparameters exist. However, as the model attempts to achieve the optimal clustering automatically, tuning the hyperparameters can be perceived as limiting the overall model. Though to examine what hyperparameter tuning has for an influence the number of topics, n-grams, number of top keywords taken into account and the minimum topic size are tested. The n-grams parameter again is expected to be optimal for only using unigrams or maybe also bigrams, which is similar to the Word2Vec method.

Since the BERTopic model used for implementation is a combination of the four methods mentioned above, many hyperparameters exist. However, since the model tries to automatically achieve the optimal clustering, tuning the hyperparameters can be seen as a limitation of the overall model. To investigate the impact of tuning the hyperparameters, the number of topics, n-grams, the number of top keywords considered, and the minimum topic size are tested. The n-grams parameter is again expected to be optimal for using only unigrams or perhaps also bigrams, which is similar to the Word2Vec method. For the other parameters, limiting the model is only expected to have a negative effect on the results.

Chapter 6

Results

This chapter presents the results of the methods. The results are a comparison between the different methods and the important aspects for each method. The results are presented after tuning the hyperparameters, trying to approximate the optimal scores for each method. Multiple experiments were conducted with the applied models on an unlabeled dataset of documents. The labels shown were created from the top keywords for each cluster resulting from the models.

6.1 Topic Modeling

In this section, the results for each topic model are discussed. At the end of the section, the comparison between the results of the topic models is illustrated. A comparison is made between the different techniques, but the optimized form of the different models is also discussed. The performance of the models is measured by the validation measures listed in Chapter 3, Topic coherence, topic diversity, and the human interpretability of the labeled topics. Three of the four methods require the number of desired topics as an input variable, which is expected to have a large impact on the result. Therefore, the distribution of the evaluation metrics is reviewed over the number of topics. The 'optimal' number of topics is primarily chosen based on the topic coherence, however, also topic diversity is taken into account. As discussed in Chapter 5, the number of topics are chosen with a method based on the elbow method.

As the number of subjects increases, the mean distortion decreases. Each cluster has fewer constituent instances and these are closer to their respective centers. However, the improvement in the mean distortion decreases as the number of topics increases, creating this "elbow" in the graph. This implies that a low number of topics is not chosen, since it makes sense that the topic coherence Cv is still higher for it. However, this is also not considered optimal in the case of this research problem because manageable clusters are the desired goal. Especially with these topics to be distinct and contain a specific topic. If no clear elbow can be observed, the optimal number of topics will be chosen based on the optimal combination between topic coherence and topic diversity. Topic diversity is also expected to be higher with a lower number of topics, but similarly to the topic coherence this is not considered optimal in this study.

6.1.1 LDA

First, the results of the Latent Dirichlet Allocation model are reviewed, and model performance is tested on the three measures. Since LDA is a probabilistic model that requires the number of topics as input. The results based on Cv topic coherence and topic diversity are shown in the graph, with the number of desired topics ranging from 4 to 40. The results of the measurements of topic coherence and topic diversity over the number of topics are shown in Figure 6.1.

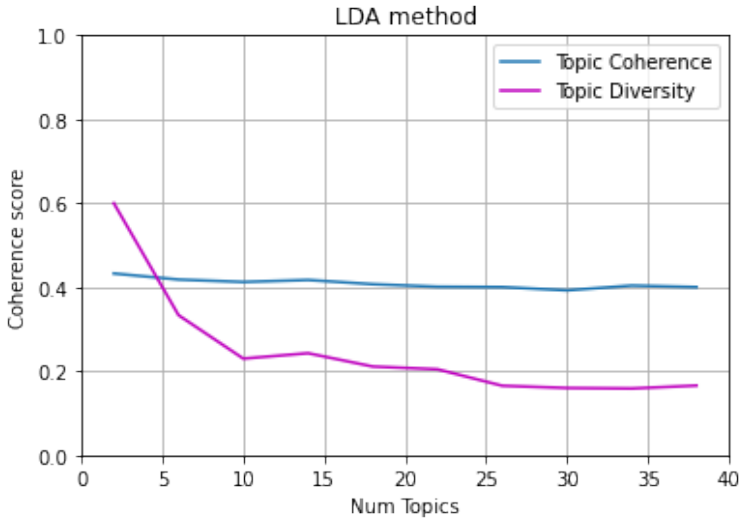


Figure 6.1: LDA number of topics

The Cv topic coherence measure is nearly horizontal for the entire window and is observed around 0.4. Topic diversity is higher for a low number of topics, dropping from 0.6 for only 2 topics to a value of around 0.2 for 10 topics and dropping slowly for the rest of the window. If too few topics are chosen, the results are not useful. In this study, the optimal number of topics was chosen to be 14, based on the graph of the topic diversity. As the topic coherence has no clear elbow to be recognized, but at number of topics 14 the topic diversity does. Here, topic coherence is about 0.42, and topic diversity is 0.25. The tuning of the hyperparameters alpha and eta was also done for this number of topics and shown in Table 6.1, resulting in an alpha of about 3.5 and an eta of about 1.4.

Hyperparameter	**LDA** (#topics 14)
alpha	3.515
eta	1.409

Table 6.1: LDA hyperparameters

The topic coherence of the model is tested with the use of four metrics, Cv coherence, u_{mass}, c_{uci} and c_{npmi}. The Cv measure and topic diversity are already seen in Table 6.1. In addition, all measures are presented in Table 6.2 for the optimal number of topics, which is chosen as 14 in the case of LDA. In Chapter 5, it is mentioned that the topic coherence is between -1 and 1 and a score of 0.4 is considered low. The scores of u_{mass}, c_{uci} and c_{npmi} are more difficult to interpret because they are discussed in dependence on the data. Nevertheless, u_{mass} is quite close to 0, indicating that it is quite good. Both the metrics c_{uci} and c_{npmi} are also below 0, meaning that the words are quite independent of each other. Moreover, topic diversity is rated as poor, with about 0.13 for the optimal number of topics.

The labels of the LDA method are expected to be less coherent because of the low coherence score of the topics and the low diversity of the topics. Examples of topic clusters are shown in Table 6.3. The keywords are not seen as descriptive words for specific topics that can be humanly interpreted. Moreover, words such as "servic" and "network" are shown as keywords for multiple clusters and "internetofth" is found in all three most coherent topic clusters. For comparison of the results, the three biggest clusters are shown for every model, meaning the clusters that

Metric	**LDA** (#topics 14)
Coherence Cv	0.426
u_{mass}	-0.801
c_{uci}	-0.023
c_{npmi}	-0.001
Topic diversity	0.129

Table 6.2: LDA coherence results

contain the most documents. The three largest resulting clusters for LDA are the same ones that generate the highest coherence. This shows that the overall clusters for LDA realize the best coherence, meaning that the clustering is unable to find individual compact clusters. Table 6.3 shows the number of documents to which cluster is the largest. The second cluster is the largest with 309 documents, followed by the third cluster with 175 and then the first with 116 documents on that topic.

Cluster	**9**	**12**	**3**
Top 3 keywords	{applic, servic, internetofth}	{resourc, internetofth, servic}	{servic, internetofth, network}
Coherence Cv	0.470022	0.438395	0.438119
Number of documents	116	309	175

Table 6.3: LDA topic coherence for three most coherent clusters

6.1.2 TF-IDF clustering

The results of the TF-IDF-based method are better than those of LDA. The input variable of the number of topics is also required for this method and is checked again for the same window of [4,40]. The effect of this variable on the Cv topic coherence and topic diversity is shown in Figure 6.2.

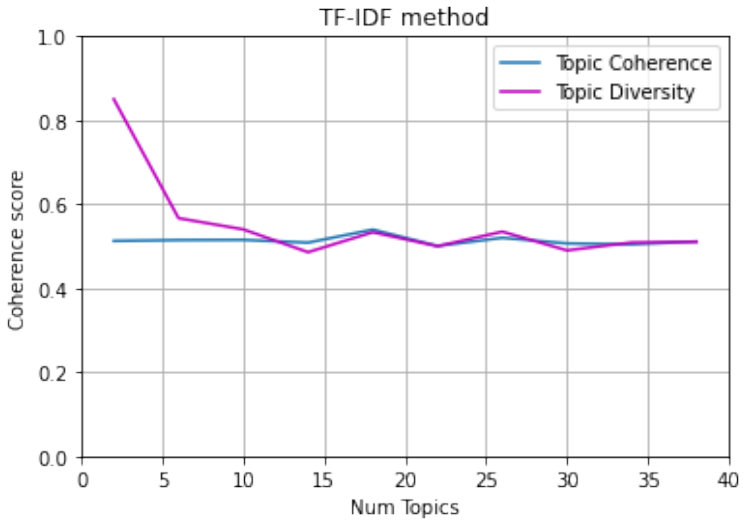


Figure 6.2: TF-IDF number of topics

Similar to the LDA, the topic coherence is almost horizontal and has a score of about 0.5 for the entire window. Topic diversity is higher for a very small number of topics. The topic diversity starts at 0.85 for only 2 topics and drops to an almost steady state at 0.5. The optimal number of topics for this method is chosen to be 18, where small peaks can be seen for both coherence and diversity. This is chosen as optimal because the topic diversity is also almost horizontal after this point in the graph, which based on the elbow method is optimal.

The study is continued with this optimal number of subjects and the hyperparameters are tuned. As mentioned in Chapter 5 for this method, there are three main hyperparameters, the n-grams, the maximum document frequency (max df) and the minimum document frequency (min df). The optimal hyperparameters are listed in Table 6.4.

Hyperparameter	**TF-IDF** (#topics 18)
n-grams	(1, 1)
max df	0.9
min df	6

Table 6.4: TF-IDF optimal hyperparameters

The result for the n-grams shows that it is better to use only the unigrams, individual words, rather than also bigrams, two words that are often said together. For the two measures of document frequency, minimum and maximum, for the maximum 0.9 is the optimal proportion of documents in which the word appears. If words occur in ninety percent of documents, it is not meaningful. For the minimum document frequency, an integer is chosen, meaning that if the word occurs no more than 6 times in the documents, it has no effect on the TF-IDF measure.

After tuning the hyperparameters, the four measures of topic coherence and the topic diversity are calculated. The results of the TF-IDF method with the 18 topics are shown in Table 6.5. Compared with the previous LDA-based method, the topic coherence improved to 0.533, which is between "average" and "good." The u_{mass} and c_uci scores are slightly further from 0, but with the knowledge that these scores generally decrease rapidly with the number of subjects, it is not a big difference. The other metric c_{npmi} is higher than 0, indicating higher co-occurrence of words within topic clusters. Finally, topic diversity, which can already be seen in Figure 6.2, is around 0.57 for the optimal number of topics.

Metric	**TF-IDF** (#topics 18)
Coherence Cv	0.533
u_{mass}	-1.466
c_{uci}	-0.187
c_{npmi}	0.049
Topic diversity	0.572

Table 6.5: TF-IDF coherence results

To better understand the numerical results, the three most coherent clusters are reported in Table 6.6. In addition, the top three keywords of these clusters and the number of documents falling into that category are noted. Then, in Table 6.7, the three largest topic clusters are observed. The difference in the number of documents found in the more coherent clusters rather than the larger clusters indicates how distinctive these smaller clusters are.

Both Tables 6.6 and 6.7 show the top three keywords. The words are still in lemmatized and stemmed form, as they are derived from multiple forms. The keywords better describe different aspects of the corpus, such as "price," "resourc," and "mechan," which are humanly interpreted as "price," "resource," and "mechanism. Moreover, the keywords overlap less than in the previous method. However, it still appears that the labels do not represent very specific components that appear in the corpus.

Cluster	**0**	**16**	**9**
Top 3 keywords	{technolog, internetofth, constitut}	{price, resourc, mechan}	{model, network, accuraci}
Coherence Cv	0.733648	0.680794	0.659399
Number of documents	19	14	69

Table 6.6: TF-IDF cluster information for three most coherent clusters

Cluster	**13**	**5**	**10**
Top 3 keywords	{network, architectur, latenc}	{time, citi, internetofth}	{platform, devic, architectur}
Coherence Cv	0.489046	0.517877	0.509533
Number of documents	164	130	99

Table 6.7: TF-IDF cluster information of three biggest clusters

6.1.3 Word2Vec clustering

The Word2Vec method is based on word embeddings and the comparison of the context of the words. The results of this approach are expected to elucidate the performance of these embeddings. The number of subjects is needed as an input variable, so this variable is examined over the same window of [4,40]. The distributions of both validation metrics, the Cv topic coherence and topic diversity, are shown in Figure 6.3.

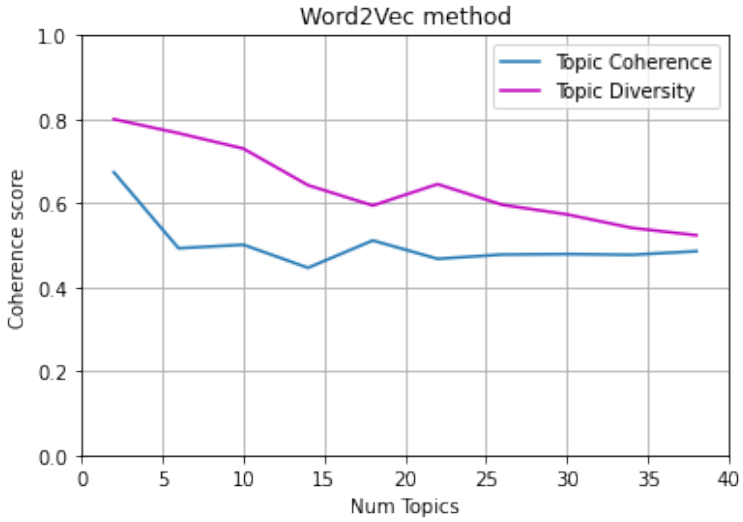


Figure 6.3: Word2Vec number of topics

The measure of topic coherence decreases, from 0.65 for 2 topics to 0.46 for 40 topics. At about 18 topics, a small increase in topic coherence is observed. Combined with the fact that the graph is horizontal after this point, which is confirmed optimal by the elbow method. Therefore, this number of topics is suggested as optimal. The topic diversity of the Word2Vec method is perceived differently from the first two methods, as the topic diversity does not decrease rapidly to a steady state. Moreover, it remains above the measure of topic coherence. Nevertheless, a steady decline from 0.8 to 0.53 is observed across the window. A topic diversity that is at least as high as the previous method is perceived as a great improvement.

The optimal number of subjects was chosen to be 18, but the hyperparameters still need to be tuned for this chosen number of subjects. The results of the tuning, which is done by changing them manually in different combinations, are shown in Table 6.8. The table shows that for the minimum occurrence of a word in the abstract, 3 is optimal, if the word occurs less than 3 times it contains too little knowledge about the topic, and if more than 3, words are removed that help create topic clusters. The sliding window parameter is also optimal at 3. If it is chosen at 2 or 4, then too little or too much context is considered, respectively. The vector size is optimal around 1000, but has a lesser effect than the hyperparameters minimal count and window. Finally, the alpha and minimum alpha, which is the value to which the alpha goes linearly, are 0.03 and 0.0007.

Hyperparameter	**Word2Vec** (#topics 18)
Min count	3
window	3
vector size	1000
alpha	0.03
min alpha	0.0007

Table 6.8: Word2Vec optimal hyperparameters

The first two hyperparameters have the most effect on the measures topic coherence and topic diversity, to which the model is fitted. Remarkably, an equilibrium is chosen as the optimal result. For a lower minimum count, the topic coherence result is slightly higher, and the same is true for taking a smaller window. In these cases, however, topic diversity is much lower, indicating that the created topics are less distinguishable. The same applies the other way around. To achieve a somewhat higher topic diversity, both parameters could be chosen larger, which also results in a decrease in topic coherence.

The results after tuning the hyperparameters of the evaluation measures are shown below in Table 6.9. The Cv coherence metric is again an improvement over previous methods, but for both the u_{mass} and c_{uci} the results are very low. The low value for both values is explained by the unlikely occurrence of the two words together. If the probability of occurring together is much smaller than the probability of occurrence of the single word, very low values are the result of the logarithm. Since the Word2vec method is based on context and thus on a sliding window, words that are somewhat further apart in the text are also calculated as co-occurring. However, their co-occurrence may be almost random in this particular case, making the measure exponentially low. The c_{npmi} score is less affected by this, because of the normalization of the measure. However, compared to the other method, the value is still lower. The high topic diversity score reveals the strength of the word embedding method. The substantiation is that the word embedding method can create fairly dense clusters and especially topics with keywords that are distinct from each other. As a result, the use of word embeddings and document embeddings is perceived as promising.

Metric	**Word2Vec** (#topics 18)
Coherence Cv	0.597
u_{mass}	-19.589
c_{uci}	-13.514
c_{npmi}	-0.470
Topic diversity	0.578

Table 6.9: Word2Vec coherence results

Human interpretability of the topics is used to validate the results and verify that the better numerical measures also lead to the desired end goal of observing topics appropriate to this study. The numerical results appear promising, with a topic coherence of about 0.6 and a topic diversity score of 0.6 for the chosen number of topics. For examination of the specifics of topic clusters in Table 6.10, the most coherent clusters are presented. In addition, the three largest clusters are also shown in Table 6.11 for clarification.

Cluster	**13**	**9**	**14**
Top 3 keywords	{june, held, submiss}	{handler, africa, pantilt}	{icv, puzzl, ultrahigh}
Coherence Cv	0.784770	0.643608	0.610721
Number of documents	8	53	32

Table 6.10: Word2Vec cluster information of three most coherent clusters

Cluster	**7**	**1**	**11**
Top 3 keywords	{icv, gg, networkingnetwork}	{networkingnetwork, icv, africa}	{alto, ilp, multiarm}
Coherence Cv	0.587316	0.434312	0.475987
Number of documents	202	163	161

Table 6.11: Word2Vec cluster information of three biggest clusters

The Tables 6.10 and 6.11 show the three most coherent and largest topic clusters created by the Word2Vec method. The labels created are seen as finding more specific components hidden in the corpus, such as "africa," "multiarm," or "icv," which stands for "Intelligent connected vehicles" in the abbreviation. However, these keywords do appear in several topic clusters as the most important ones. Moreover, the largest cluster is quite coherent, with more than 200 abstracts in that topic cluster.

6.1.4 BERTopic

The last method, BERTopic, differs from the previous methods in several ways. First, this method uses fully scraped abstracts, which makes for an interesting comparison with the other methods. The results include words that are similar to each other because they are not lemmatized and stemmed, as well as common words that are filtered out in the other methods. The HDBSCAN cluster method also groups hard-to-cluster documents into another "noise" cluster. This cluster is not included in the results, which also affects the comparison. Furthermore, the number of topics is not needed as an input variable. The BERTopic model is run and determines by itself the optimal number of topics. Nevertheless, the distribution of evaluation metrics across the number of topics is examined for comparison. Figure 6.4 shows the topic coherence

and topic diversity generated by the BERTopic model for the identical window of the number of topics. The expectation that the pre-trained language model outperforms the other probabilistic methods is tested in this subsection.

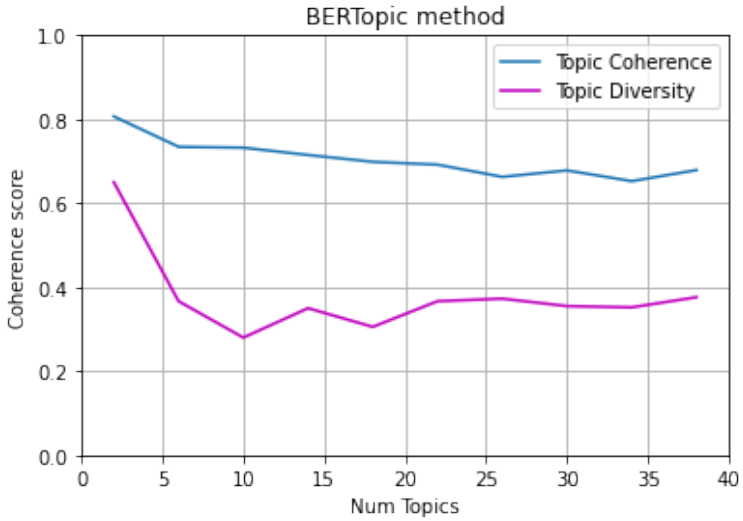


Figure 6.4: BERTopic number of topics

If we look at the graph of topic coherence in Figure 6.4 we notice that the measure remains above 0.65 for the entire window. Thus, the topic coherence is the highest of all applied methods. The measure starts around 0.82 and drops slightly to just above 0.65. According to the perspective in Chapter 5, a score above 0.7 is considered very good. The topic diversity score is slightly lower than expected, starting at around 0.65 and dropping asymptotically to a near steady state at 0.38.

This second measure is lower than expected, but it is argued that without the data preprocessing, common words are also included in topic diversity. This has less effect on topic coherence, since coherence is based on the similarities between documents within a topic cluster. However, topic diversity relies on the top ten keywords for different clusters that may be similar in case stop words still intrude. The problem with removing the common words and calculating topic diversity afterwards is calculating topic diversity on far fewer words. This is expected to yield very high topic diversity if the clusters all have different keywords, but that would also be the case for the other methods.

In the BERTopic method, an optimal number of topics is not chosen because the model calculates it itself. However, tuning other hyperparameters can further improve the results. The hyperparameters considered are the minimum topic size, n-grams, and the highest number of predicted words per topic. The optimal hyperparameters are shown in Table 6.12.

Hyperparameter	**BERTopic** (#topics 21)
min topic size	10
n-grams	(1,1)
top n words	10

Table 6.12: BERTopic optimal hyperparameters

The results show that a number of 21 topics is considered optimal. The results of the hyper-parameter tuning are similar to the Word2Vec model, again considering only unigrams. The minimum topic size defaults to 10 and when it is increased, the topic coherence score decreases. However, when the minimum topic size is decreased, topic coherence results in an illogically large number of clusters. Therefore, it is chosen to keep the minimum topic size at 10. The highest number of words extracted per topic is preferably between [10,20]. The optimal result of topic coherence is achieved when 10 is chosen.

The result of running the BERTopic model is shown in Table 6.13. The high topic coherence score of 0.7 is considered "very good" by the evaluation metric perspective created in Section 5.2, and the other measures also show no obvious weakness of the model. However, the u_{mass} score is still slightly away from 0, quite logical for the relatively high number of topics of 21. The c_{uci} and c_{npmi} are both slightly above 0, indicating the co-occurrence of the important words per cluster. Only the topic diversity is relatively low compared to the other results, but that could be because the common words are not filtered out in this method. The comparison of only the top three keywords can be seen as a better indicator of topic diversity of the BERTopic model. This is shown in the Subsection 6.1.5.

Metric	**BERTopic** (#topics 21)
Coherence Cv	0.703
u_{mass}	-0.981
c_{uci}	0.279
c_{npmi}	0.119
Topic diversity	0.383

Table 6.13: BERTopic coherence results

For further analysis of the results in the Tables 6.14 and 6.15, specific cluster characteristics are shown. First, the coherence values shown are high per topic cluster, implying the density of the clusters is high. Moreover, both the larger and smaller ones receive high coherence scores. In addition, the top three keywords are very humanly interpretable as being similar to {health, healthcare, emergency} which are all closely related, which is also the case for {manufacturing, industrial, manufacturing} and {video, transcoding, vr}. But even the largest clusters are perceived as humanly comprehensible with "vehicular", "vehicles" and "vec", meaning "Vehicular edge computing," containing the same word almost three times. The combination of the two words "deep" and "learning" is also interesting, since the concept of "deep learning" is very much related to edge computing. The reason only two words are shown for this cluster is the removal of common words from the labeling.

Cluster	**6**	**8**	**20**
Top 3 keywords	{health, healthcare, emergency}	{manufacturing, industrial, production}	{video, transcoding, vr}
Coherence Cv	0.958	0.906	0.846
Number of documents	33	31	11

Table 6.14: BERTopic cluster information of three most coherent clusters

Cluster	**0**	**1**	**2**
Top 3 keywords	{problem, offloading, algorithm}	{learning, deep}	{vehicular, vehicles, vec}
Coherence Cv	0.691	0.746	0.773
Number of documents	139	114	92

Table 6.15: BERTopic cluster information of three biggest clusters

Moreover, because BERTopic uses HDBSCAN for clustering it has a noise cluster. This concept is explained in Chapter 3. It is interesting to see how many documents that are clustered as noise in the model. Therefore, the full set of topic clusters is presented in Table 6.16. The table shows that about 400 of the abstracts are clustered as noise, which is fairly large cluster. However, for all the other clusters, really specific topic labels are created and none of the clusters have unusual results.

Topic	Count	Name
−1	391	('Noise')
0	139	['problem', 'offloading', 'algorithm']
1	114	['learning', 'deep']
2	92	['vehicular', 'vehicles', 'vec']
3	84	['security', 'privacy', 'trust']
4	52	['migration', 'network']
5	36	['query', 'processing']
6	33	['health', 'healthcare', 'emergency']
7	32	['smart', 'power']
8	31	['manufacturing', 'industrial', 'production']
9	27	['blockchain', 'iot', 'security']
10	26	['caching', 'mobile', 'wireless']
11	23	['reinforcement', 'learning', 'energy']
12	22	['first', 'device', 'application']
13	21	['5g', 'network', 'mec']
14	20	['users', 'game', 'revenue']
15	20	['sdn', 'networking', 'network']
16	18	['vehicles', 'offloading', 'vehicular']
17	16	['energy', 'fpgas', 'power']
18	15	['miners', 'blockchain', 'resource']
19	11	['cloud', 'erp', 'markets']
20	11	['video', 'transcoding', 'vr']

Table 6.16: BERTopic results

6.1.5 Comparison of the methods

For a better comparison of the results, the overall results of the methods are shown in Table 6.17. The number of clusters differs for the different methods because of the optimal chosen for each model. The results depend on this variable, but not significantly enough to tip the balance in favor of one of the other methods. This can be seen in the flat graphs in the previous results sections, meaning that the results are not strongly dependent on the number of subjects. Regarding Cv coherence, the results show that the BERTopic method is the best, as its score is significantly higher than that of all other methods. For the distinctiveness of the resulting clustering represented by Topic Diversity, both methods applying Topic modeling with text clustering, the TF-IDF and Word2Vec methods outperform the other methods. For human interpretability, a subjective measure of cluster label interpretation, the BERTopic method again yields the best result.

Both the Word2Vec and BERTopic methods use embeddings for clustering. Both methods' reliance on word embeddings seems to yield better results. The use of embeddings takes into account not only the words but also their meaning and context, which in this study is believed to contribute to denser and clearer clusters. The TF-IDF clustering-based method also achieves results close to the Word2Vec method, but depends on the word importance of each document. The dataset used for this study consists of similar text articles, which understandably contain many words that appear frequently in the texts. It is argued that the good results of the TF-IDF method are caused by the elimination of those frequently occurring words, based on the hyperparameter maximum document frequency, and with the remaining words, the method can find good clustering based on the keywords that are not eliminated. Moreover, this method relies heavily on finding the exact keywords per cluster, and the calculation of the validation scores is also based on these keywords. Therefore, the TF-IDF method is expected to achieve high results on the coherence and diversity measure.

The Word2Vec and BERTopic methods are believed to have a better understanding of the similarity of synonyms because the synonyms are considered related due to similar context. The fact that the topic diversity of BERTopic is significantly lower is noted, but as discussed in the Chapter 5 it is argued that this is because no pre-processing occurs for this model. Removing or modifying the top keywords so that they are more similar to the results after preprocessing is considered modifying the results. Nevertheless, the resulting topic clusters can be assessed for human interpretability by seeing what words remain when stop words are extracted from the most important words per cluster. Moreover, the methods LDA, TF-IDF and Word2Vec can be fairly compared on the basis of topic diversity since these three methods are preprocessed according to the same steps.

Metric	**LDA**	**TF-IDF**	**Word2Vec**	**BERTopic**
Nr of clusters	14	18	18	26
Coherence Cv	0.426	0.533	0.597	0.703
u_{mass}	-0.801	-1.466	-19.589	0.981
c_{uci}	-0.023	-0.187	-13.514	0.279
c_{npmi}	-0.001	0.049	-0.470	0.119
TD(top10)	0.129	0.572	0.578	0.383

Table 6.17: All methods

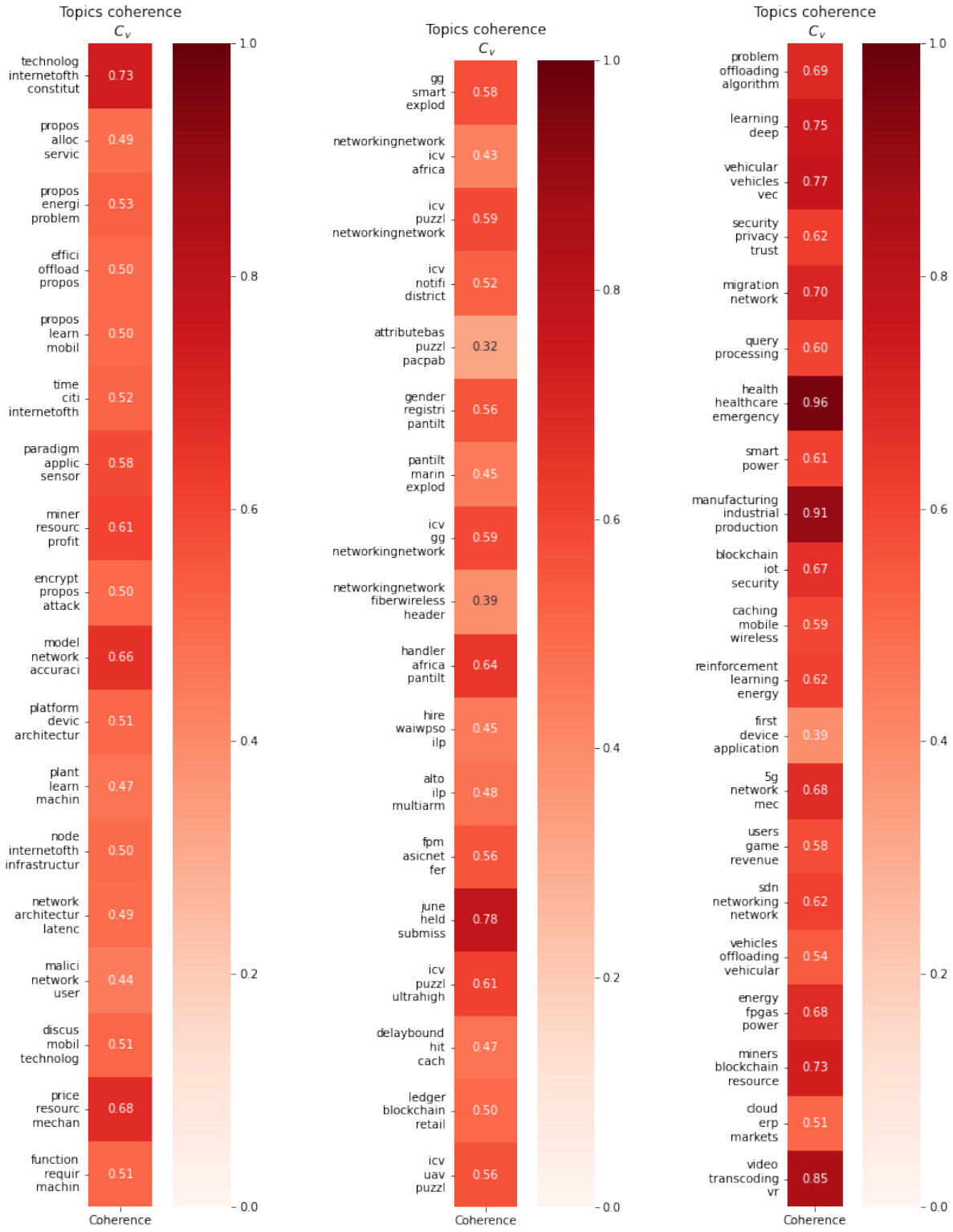
The results in Table 6.17 show that LDA is much less able to form dense clustering. Moreover, the least pronounced clusters are formed using this approach. The three largest and most coherent clusters of the LDA method are shown in Table 6.3, and it is concluded that not much information can be derived from these results. Therefore, Figure 6.5 reveals the in-depth cluster insights for

only the TF-IDF clustering, the Word2Vec clustering and the BERTopic method. For the within-cluster analysis of these three methods, the labeled clusters are shown in Figure 6.5.

The within-cluster coherence results show that for the BERTopic method, all clusters have a high density, meaning that the documents within the topic clusters are considered very similar. For both the TF-IDF and Word2Vec cluster methods, the results are less consistent. Clusters with high topic coherence and clusters with lower topic coherence are both represented, meaning that some clusters contain documents that are less related to others within the cluster.

The last evaluation measure, assessed as subjective, is indicated as human interpretability. The labels are presented as the three main keywords for each cluster and must have similar meanings for people to understand the actual topic. The labels per cluster are shown alongside the coherence score in Figure 6.5 to create an overview of the entire topic clustering, rather than just the most coherent or largest clusters. The measure topic diversity is calculated by determining whether the exact same word appears in the top ten words of the other cluster, and for the BERTopic method this resulted in a low topic diversity score. But for the labels in Figure 6.5 the stop words removed. Then the words that contain actual information about the type of topic the cluster contains are displayed.

The results in Figure 6.5 show that for each topic cluster created by the BERTopic method, a human interpretable topic can be derived. Topics such as "Deep learning," "5G network," "Healthcare," "Vehicles," and "Blockchain" are all topics closely related to the search query "Edge Computing. These would be expected to be topics if the topic clusters were created by humans. BERTopic's resulting labels reflect such human understanding of the dataset. For both TF-IDF clustering and Word2Vec clustering, the results are less coherent and less human understanding. Although the topic diversity score is higher for these methods, this is not reflected in the usefulness of the resulting clusters.



(a) TF-IDF method (b) Word2Vec method (c) BERTopic method

Figure 6.5: Topic coherence per topic cluster

Chapter 7

Discussion & Future work

This study compares different Topic Modeling techniques based on how well the models are suited to solve the research problem. Each model requires its own interpretation of what best fits the research problem. However, each study has limitations and possible improvements. The results are put into perspective in the Discussion section, and the limitations and improvements of the study are discussed in the Limitations section. Followed by the Future Work section, which describes interesting next steps in this research field that contribute to these improvements, remove limitations, or are good applications of this research.

7.1 Discussion

The purpose of this section is to evaluate the results and put them in perspective with previous research. It further discusses whether the results were expected and, if not, what might be the reason for the deviation of the results from expectations.

When interpreting the results, the basic Topic Modeling LDA method is considered the worst applied method because its results are the lowest for all three measures. Next, both TF-IDF clustering and Word2Vec clustering, methods that apply Topic Modeling with text clustering, achieve higher performance. Finally, Topic Modeling with a language model performs very well on both topic coherence and human interpretability, but less well on topic diversity.

The results of the basic Topic Modeling technique LDA are disappointing. The Cv topic coherence score is below 0.45, which is considered a 'low' score according to Section 5.2. For a model that is considered one of the most popular in Topic modeling, this is unexpected. [56] Topic models based on LDA are known for their good results in various fields, such as in Social Networks [11, 20, 41], Political Science [25, 44], Software engineering [13, 16, 23], and Medical science [18, 26, 47]. However, the main problem in this research is that all the papers are based on one specific topic and thus contain similar vocabulary. Therefore, the task of Topic Modeling is complicated. Moreover, only the abstract text are used as data, and the removing words in preprocessing steps leaves quite short texts. LDA learns word co-occurrences by interpreting each document as a mixture of topics. This reasoning suffers from the sparsity of word co-occurrence patterns in shorter texts. [30, 32, 50, 61] This is expected to be the cause of the poor performance of the LDA method.

The Topic modeling techniques with text clustering did achieve better results. The topic coherence of both methods is already considered between "average" and "good" according to the explanation of the validation measure in Section 5.2. Moreover, the topic diversity score is very high. The TF-IDF measure, as described in Chapter 3, is a measure known for its use to determine the word importance for documents. It is used for text preprocessing to understand which words are insignificant or occur too often. [19] Combining the TF-IDF measure with clustering

is less common in literature. In some cases it is done for verification of the TF-IDF results, the word importance matrix. [37] In recent years, however, the comparison between TF-IDF-based methods and word embeddings has become more common, with the results showing that the word embedding-based methods achieve slightly better results than the TF-IDF-based ones. [75, 80]

Therefore, the Word2Vec method, which uses word embeddings, is expected to understand the texts better. The results shown in Chapter 6 are also slightly better than those of TF-IDF, but for the human interpretability, the difference is more difficult to quantify. Figure 6.5 shows that the TF-IDF created labels include more general words, such as "model", "service" and "technology", rather than nouns that are considered more explanatory from a human perspective. More explanatory words in the results of Word2Vec would be "icv", "uav", "africa", which are words that are more specific for a smaller domain of documents. The Word2Vec model does achieve much better results than the LDA method, which is to be expected because in cases that involve short texts Word2Vec has been shown to outperform LDA models. [50] Thus, this study shows results that were to be expected. Both methods outperform the LDA-based method, and Word2Vec slightly outperforms the TF-IDF based one. It is understood that the combination of feature selection, using word importance and embeddings for TF-IDF and Word2Vec, respectively, with text clustering, using K-means, leads to better results for Topic Modeling. Especially when smaller text documents are taken.

BERTopic has already been praised in many other studies for its understanding of the texts and making clear, dense clusters. [71, 74]. The results in this study substantiate this in the case of topic coherence and human interpretability, with a Cv topic coherence of "very good" according to the validation measure explanation in Section 5.2 and the most human understandable clusters of all methods. Recent literature has shown that even on Twitter posts, which are very small texts, their BERTopic model is able to outperform LDA and other word embedding models. [86] LDA struggled with smaller texts, but BERTopic can thus also outperform both TF-IDF clustering and Word2Vec clustering. So the recent study and results for Twitter messages achieve similar results to this study. [86] The results of our BERTopic method substantiate that with a language model, Topic modeling can achieve better interpretation of texts, and create clearer and denser clusters for small text data. Be it Twitter data or abstract data.

7.2 Limitations

This chapter discusses limitations and possible improvements. The research is still leading to contributions in the field. However, improvements are always possible, and it is important to know which decisions can be investigated further.

The first limitation is based on the final comparison of the different models, since the output of the LDA, TF-IDF and Word2Vec methods is preprocessed prior to the implementation of the methods and the BERTopic method is not. The techniques are compared based on three metrics: topic coherence, topic diversity and human interpretability based on labeling. The comparison of the resulting metrics becomes much more difficult for topic diversity and human interpretability, as the BERTopic results still include stop words and words in different forms. The measurement of topic coherence is calculated based on the most important words found per topic compared to the documents found in that topic cluster, so that the coherence of a cluster is not affected by stop words that also appear in the results. The cluster coherence calculation uses probability estimation, which means that the common words do not affect the topic coherence results when the words appear in all topics. For topic diversity, however, it should be noted that this measure is calculated on the difference of the resulting top ten keywords alone. Thus, if the resulting top lists contain many similar stop words, the achieved topic diversity is low.

For human interpretability of the BERTopic method, a subjective measure of the most common "meaningful" words, the stop words are removed. The results of BERTopic show that when these stop words are removed, very similar words per cluster are the most descriptive. On the other hand, different word forms or similar words together are also still considered most descriptive. The example of "vehicles," "vehicular" and "vec" provides three words that could have been preprocessed into the same word. Removing the stop words attempts to create the fairest possible comparison between models. Nevertheless, it is important to understand that the comparison is not completely equal. Section 7.3 explains next steps to achieve an even fairer comparison.

Another aspect to be addressed is HDBSCAN versus K-means as a clustering method. For both TF-IDF and Word2Vec, the K-means clustering technique is used because in both methods, the word importance matrices of TF-IDF and the document embeddings of Word2Vec have high dimensionality. If one were to use HDBSCAN clustering for these methods, a large percentage of the documents are clustered in the noise cluster because the similarity between the documents is lower. HDBSCAN requires dimensionality reduction and BERTopic uses UMAP for this purpose. The problem of using dimensionality reduction on the TF-IDF and Word2Vec methods is that the knowledge is lost which word significance or word embeddings cause the similarity between two documents. The comparison of the two methods is based on the probability of words at specific places in the feature matrix, and comparing documents on all word importance for TF-IDF and word embeddings for Word2Vec provides interesting insights. For the BERTopic model, it is much more difficult to discern the degree of similarity of documents at the word level in addition to keywords. Therefore, the K-means clustering method was chosen for both TF-IDF clustering and Word2Vec clustering. However, this distinction complicates the comparison.

The noise cluster introduces another interesting difference. In K-means clustering methods, all documents must be part of a cluster. But as in any dataset, there are noise documents or outliers. These outliers lower topic coherence in the clusters for these methods because the outliers are still placed in the cluster closest to them, rather than actually being similar to the other documents in that cluster. Therefore, when topic coherence is calculated this document has a much lower similarity than the other documents and the coherence in the cluster is lower. On the other hand, the measure of topic diversity may be higher because several words of these outliers may appear in the topic labels. In the LDA method, clustering is done based on the topic with the highest probability per document, which also leads to the phenomenon of outliers ending up in a cluster. The clustering from BERTopic does result in a noise cluster, which provides a higher topic coherence for the actual clusters because the documents that do not fit well into the cluster fall into the noise cluster. This provides an advantage for the Cv topic coherence in the BERTopic method.

The next limitation to be discussed is the fact that applying the models for the desired purpose of improving the research phase of consultants requires that them to operate in real time. For a language model or most data science models, this poses a problem. The running time of the models was examined and BERTopic in particular took longer than the other three methods by consistently running longer than 2 minutes. The Word2Vec clustering and TF-IDF clustering methods can cluster documents within a minute. LDA is the fastest method, which can calculate the output of the model within three seconds. However, LDA, TF-IDF clustering and Word2Vec clustering have the time disadvantage of requiring pre-processing of the text. The preprocessing is not required for the BERTopic model. Preprocessing takes a total of almost 2 minutes. This is mainly due to lemmatization and stemming, which already took more than a minute and a half. Thus, compared to the other two methods that achieved reasonable results, TF-IDF clustering and Word2Vec clustering, the total running time of BERTopic is even lower if preprocessing is included for the other two. Of course, the above scenario depends on whether the abstracts were

already available in a database. If the abstracts were already available earlier, the preprocessing could have already been done. In that case, LDA would take only a few seconds and BERTopic would take over 2 more minutes. This would imply that it depends on the situation which model is preferred.
However, another important aspect of the proposed solution for Topic Modeling is the required scraping. In the case that an abstract database is available, this is not required, however if the text still need to be scraped this also needs to be taken into account. For all methods, scraping takes up a lot of time because the texts must be retrieved as HTML script and decoded into usable text. The scraping at all different publishers took more than 20 minutes because all publisher had different scripts and the retrieving of the HTML script takes time. This is not a limitation for specific models as this is required for all methods, but needs to be taken into account.
However, another important aspect of the proposed solution for Topic Modeling is the required scraping. In the case where an abstract database is available, this is not necessary, but if the text still needs to be scraped, this should also be taken into account. For all methods, scraping is time-consuming because the texts must be retrieved as HTML scripts and decoded into usable text. Scrapping at all different publishers took more than 20 minutes because retrieving the HTML script takes time. This is not a limitation for specific models because it is required for all methods, but it should be taken into account. Besides, building the scripts for all publishers that are in the database, requires more work beforehand.

7.3 Future work

This section discusses interesting next steps that can be taken in the field of research. This study attempts to answer the question of which techniques are most useful for a consultant's research phase and for finding the right documents, but many more aspects can be explored. It can be argued what are the most important, promising or even logical steps to continue this research. A short list is presented in this chapter.

The first step that could add depth to this research is to expand the comparison of techniques based on the actual documents resulting from document clustering. Metrics are calculated for the most important words of the cluster and based on the resulting words for both preprocessed and unprocessed documents. For even better comparison, metrics can be calculated on the full summaries and all stop words and non-lemmatized and non-stemmed words. This is already the case for the BERTopic method since no preprocessing is required, but to more accurately compare it to the other three methods.

In addition, the results of the models could be better compared by creating a summary of the documents in the topic cluster. Multi-document summarization is an interesting new area that is considered quite complex, but would provide much insight if a summary of the collection of all documents in a cluster could be made. What aspects are most important in such a summary, perhaps keywords and findings in that area or a listing of facts about it.

Another interesting aspect is to further investigate the use of abstracts as summary for articles. What is the effect on clustering, topic coherence, topic diversity and human interpretability when full texts, titles or even just keywords are used.[45] Already a small insight into using only the titles taught us that clear clusters could already be calculated, but this was not tested further. Such a solution has a great effect on creating an application that works in real-time, because they do not have to be scraped from the Web. A document database could have already stored them, which would lead to a much lower runtime and thus a better application in real-time.

Furthermore, investigate an actual application that uses the models examined in this study. The application must be user-friendly, understandable, efficient and accurate for consultants to want

to use it. It would be interesting to further investigate what consultants prefer and what creates the most clarity. A real test case with an accompanying survey would be an interesting follow-up.

Then document diversity opportunities. Consulting firms work with SPEs (Subject Project Experts), who provide analysts and consultants new to a project with appropriate documentation, previous work and findings for that project. This documentation can range from Excel sheets to models to PowerPoints, all of which contain text. For example, when much documentation consists of PowerPoints, the pptx library can be used to collect all the text from the PowerPoints and cluster these documents. The main difference is that the documents may differ more than those used in this study. But if that is the case, it will only simplify the clustering.

Another interesting point in applying the techniques is the chosen importance of the documents. The importance of each document can be based on its similarity to other documents within that cluster. Such a document can be called a centroid because it contains the most representative information of the cluster. However, documents can also be ranked by cluster based on the most citations, the most groundbreaking or popular articles that the cluster contains. Moreover, it can be argued that the newest article is the most relevant to the time, as older articles may be more outdated. In future work, it would be interesting to argue which importance of the articles would be most beneficial in a similar research problem.

Chapter 8

Conclusion

Topic modeling is considered a complex task within Text mining. Finding "hidden" topics in a bundle of articles can already be a complex task for any human, and for machines that do not understand the context, meaning or interpretation of text, it becomes even more complex. Moreover, in cases where topics have been extracted and validation measures calculated, it remains difficult to explain the true meaning of the topics. This study makes an attempt to find out which Topic modeling techniques work best for solving the research problem and the reason why this technique performs as it does. This thesis focuses on finding and testing different Topic modeling techniques to answer the research question: How can Topic Modeling and clustering be useful in improving the research phase for consultants in Google Scholar articles?

To answer this question, a dataset of 1266 usable abstracts all based on the test topic "Edge Computing" is used. This dataset comes from IEEEDataPort and is an open access dataset. This dataset serves as a test case of the Topic Modeling techniques and represents the results of a search query that would be of interest during the research phase of consultants. The dataset was created from Google Scholar indexed articles from 2003-2019 on the test topic.

In recent years, the focus in Natural Language Processing and specifically in Topic Modeling has been on comparing methods such as LDA, TF-IDF clustering and Word2Vec clustering with language models such as BERTopic. LDA is a method that has long shown good performance for Topic Modeling, while TF-IDF and Word2Vec embrace newer techniques for Topic Modeling. It is interesting to compare the results of these methods with the use of a language model, which has only been known for Topic Modeling in recent years. Therefore, this thesis examines these four methods and explores their potential for solving the research question. All four methods are considered good working options in document clustering, but the problem addressed in this thesis is the similarity of documents in the corpus. The entire corpus consists of articles on "Edge computing" and in a diverse corpus on different topics, this entire corpus could be seen as one topic cluster.

The final comparison between the results of the techniques is shown in Table 6.17. The highest result in topic coherence, the main evaluation metric of this thesis, is clearly obtained by the BERTopic method. As discussed in Chapter 5, achieving a topic coherence score above 0.7 is probably considered the best possible, confirming the strength of this result. The combination of using text embeddings and the class-based TF-IDF score outperforms all other techniques. The text embeddings help in understanding synonyms and context of the texts, while the class-based TF-IDF score helps in finding the correct clustering based on the most important words for each cluster. The good results of the BERTopic method are contradicted by the results for topic diversity, but it is argued that this low score is due to the fact that for this method it is optimal for the topic coherence not to use certain pre-processing steps, such as removal of stop words and lemmatization and stemming. However, this has a negative effect on the topic diversity score.

The human interpretability, subjectively assessed, of the BERTopic method is again rated as the best of all the methods tested. The justification that the resulting clusters are in fact discrete clusters and low topic diversity is related to the pre-processing steps.

The results of the Word2Vec model are also useful for understanding the quality of word embeddings. Its results are also considered coherent, while high performance is also obtained on the basis of topic diversity. The comparison on topic diversity with BERTopic is complex because different word forms are also taken into account for that score. However, combined with human interpretability, which is much higher for the BERTopic model, the overall performance of Word2Vec is considered lower. The TF-IDF clustering achieved similarly high results for topic coherence, but it is argued that this is mainly based on the similarity of the basis of the clustering on the most important keywords, which are the exact words used to calculate the Cv topic coherence. In a sense, it can be said that the combination of embeddings and a TF-IDF score in the BERTopic model combines the two positive features of these two other methods.

Looking back at the research question to improve the research phase of consultants, high topic coherence combined with very high human interpretability is considered the most important. For a consultant researching a search area, the most important thing is that when directed to a specific topic cluster or clusters, the documents within that cluster are actually very similar. With these dense clusters, the labels show up better because all the documents are linked to these keywords and the documents are actually similar. Another important observation about the result of the BERTopic model is that the least coherent cluster is not considered bad, meaning that all clusters contain similar documents and no cluster contains documents that are more diverse. The importance of human interpretability is examined in terms of the actual usefulness of the topic cluster. For a consultant to choose an appropriate cluster of documents in which to continue his research, he must understand the topic that comprises the cluster. Topic diversity would be less appropriate in this case because for a consultant looking for specific documents on a topic that encompasses multiple clusters, multiple clusters may need to be examined. This means that the consultant's search area would be limited to multiple clusters instead of just one, which is already an improvement for the research phase.

In conclusion, this thesis focused on finding the technique that provides the most value for the research phase of consultants. The research led to the conclusion that using the BERTopic model provides the most appropriate results for the research problem. The resulting topic clusters have the highest average coherence, the least diverse results and the highest human interpretability. With a Cv topic coherence score of about 0.7, the model achieves high performance on this complex problem of a very dense corpus. Based on the results of this thesis, it is argued that Topic modeling and clustering can be very useful to improve a consultant's research phase in Google Scholar articles. The results are promising and represent exceptional first steps toward automating the research phase of consultants with Topic Modeling and using the text of the articles themselves in finding the right information.

Appendix A

A.1 Abbreviations

Abbreviation dictionary:

{ WBI : web based intermediaries,
IoT : internet-of-things,
Internet of Things : internet-of-things,
AR : augmented reality,
MEC : mobile edge computing,
ESP : edge computing service provider,
VM : virtual machine,
VEC : vehicular edge computing,
NFV : network function virtualization,
SDN : software defined networking,
RNS : responsive neurostimulation,
LSE : large-scale smart environment,
IMCS : intelligent manhole cover management system,
V2X : vehicle-to-everything,
MDP : markov decision process,
OO : ordinal optimization,
IOV : internet-of-vehicles,
SC : service chaining,
NSH : network service header,
IoMT : internet of mobile things,
DNN : deep neural network,
ECC : edge-centric computing,
FC : fog computing,
ICT : information and communications technology,
EC : edge computing}

A.2 Stop words

Stop words removed by Gensim:
{'all', 'six', 'just', 'less', 'being', 'indeed', 'over', 'move', 'anyway', 'four', 'not', 'own', 'through', 'using', 'fifty', 'where', 'mill', 'only', 'find', 'before', 'one', 'whose', 'system', 'how', 'somewhere', 'much', 'thick', 'show', 'had', 'enough', 'should', 'to', 'must', 'whom', 'seeming', 'yourselves', 'under', 'ours', 'two', 'has', 'might', 'thereafter', 'latterly', 'do', 'them', 'his', 'around', 'than', 'get', 'very', 'de', 'none', 'cannot', 'every', 'un', 'they', 'front', 'during', 'thus', 'now', 'him', 'nor', 'name', 'regarding', 'several', 'hereafter', 'did', 'always', 'who', 'didn', 'whither', 'this', 'someone', 'either', 'each', 'become', 'thereupon', 'sometime', 'side', 'towards', 'therein', 'twelve', 'because', 'often', 'ten', 'our', 'doing', 'km', 'eg', 'some', 'back', 'used', 'up', 'go', 'namely', 'computer', 'are', 'further', 'beyond', 'ourselves', 'yet', 'out', 'even', 'will', 'what', 'still', 'for', 'bottom', 'mine', 'since', 'please', 'forty', 'per', 'its', 'everything', 'behind', 'does', 'various', 'above', 'between', 'it', 'neither', 'seemed', 'ever', 'across', 'she', 'somehow', 'be', 'we', 'full', 'never', 'sixty', 'however', 'here', 'otherwise', 'were', 'whereupon', 'nowhere', 'although', 'found', 'alone', 're', 'along', 'quite', 'fifteen', 'by', 'both', 'about', 'last', 'would', 'anything', 'via', 'many', 'could', 'thence', 'put', 'against', 'keep', 'etc', 'amount', 'became', 'ltd', 'hence', 'onto', 'or', 'con', 'among', 'already', 'co', 'afterwards', 'formerly', 'within', 'seems', 'into', 'others', 'while', 'whatever', 'except', 'down', 'hers', 'everyone', 'done', 'least', 'another', 'whoever', 'moreover', 'couldnt', 'throughout', 'anyhow', 'yourself', 'three', 'from', 'her', 'few', 'together', 'top', 'there', 'due', 'been', 'next', 'anyone', 'eleven', 'cry', 'call', 'therefore', 'interest', 'then', 'thru', 'themselves', 'hundred', 'really', 'sincere', 'empty', 'more', 'himself', 'elsewhere', 'mostly', 'on', 'fire', 'am', 'becoming', 'hereby', 'amongst', 'else', 'part', 'everywhere', 'too', 'kg', 'herself', 'former', 'those', 'he', 'me', 'myself', 'made', 'twenty', 'these', 'was', 'bill', 'cant', 'us', 'until', 'besides', 'nevertheless', 'below', 'anywhere', 'nine', 'can', 'whether', 'of', 'your', 'toward', 'my', 'say', 'something', 'and', 'whereafter', 'whenever', 'give', 'almost', 'wherever', 'is', 'describe', 'beforehand', 'herein', 'doesn', 'an', 'as', 'itself', 'at', 'have', 'in', 'seem', 'whence', 'ie', 'any', 'fill', 'again', 'hasnt', 'inc', 'thereby', 'thin', 'no', 'perhaps', 'latter', 'meanwhile', 'when', 'detail', 'same', 'wherein', 'beside', 'also', 'that', 'other', 'take', 'which', 'becomes', 'you', 'if', 'nobody', 'unless', 'whereas', 'see', 'though', 'may', 'after', 'upon', 'most', 'hereupon', 'eight', 'but', 'serious', 'nothing', 'such', 'why', 'off', 'a', 'don', 'whereby', 'third', 'i', 'whole', 'noone', 'sometimes', 'well', 'amoungst', 'yours', 'their', 'rather', 'without', 'so', 'five', 'the', 'first', 'with', 'make', 'once'}

Bibliography

[1] Edward W Forgy. "Cluster analysis of multivariate data: efficiency versus interpretability of classifications". In: *biometrics* 21 (1965), pp. 768–769.

[2] J MacQueen. "Classification and analysis of multivariate observations". In: *5th Berkeley Symp. Math. Statist. Probability*. 1967, pp. 281–297.

[3] Stuart Lloyd. "Least squares quantization in PCM". In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.

[4] Jonathan J Webster and Chunyu Kit. "Tokenization as the initial phase in NLP". In: *COLING 1992 volume 4: The 14th international conference on computational linguistics*. 1992.

[5] Kevin Beyer et al. "When is "nearest neighbor" meaningful?" In: *International conference on database theory*. Springer. 1999, pp. 217–235.

[6] Michael Steinbach, George Karypis, and Vipin Kumar. "A comparison of document clustering techniques". In: (2000).

[7] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. "On the surprising behavior of distance metrics in high dimensional space". In: *International conference on database theory*. Springer. 2001, pp. 420–434.

[8] David M Blei, Andrew Y Ng, and Michael I Jordan. "Latent dirichlet allocation". In: *Journal of machine Learning research* 3.Jan (2003), pp. 993–1022.

[9] Michael D Lee, Brandon Pincombe, and Matthew Welsh. "An empirical evaluation of models of text document similarity". In: *Proceedings of the annual meeting of the cognitive science society*. Vol. 27. 27. 2005.

[10] Hans-Hermann Bock. "Clustering Methods: A History of k-Means Algorithms". In: *Selected Contributions in Data Analysis and Classification*. Ed. by Paula Brito et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 161–172. ISBN: 978-3-540-73560-1. DOI: 10.1007/978-3-540-73560-1_15. URL: https://doi.org/10.1007/978-3-540-73560-1_15.

[11] Andrew McCallum, Xuerui Wang, and Andrés Corrada-Emmanuel. "Topic and role discovery in social networks with experiments on enron and academic email". In: *Journal of artificial intelligence research* 30 (2007), pp. 249–272.

[12] Leonard Richardson. "Beautiful soup documentation". In: *Dosegljivo: https://www. crummy. com/software/BeautifulSoup/bs4/doc/.[Dostopano: 7. 7. 2018]* (2007).

[13] Erik Linstead, Cristina Lopes, and Pierre Baldi. "An application of latent Dirichlet allocation to analyzing software evolution". In: *2008 seventh international conference on machine learning and applications*. IEEE. 2008, pp. 813–818.

[14] Gerlof Bouma. "Normalized (pointwise) mutual information in collocation extraction". In: *Proceedings of GSCL* 30 (2009), pp. 31–40.

[15] Hanna M Wallach et al. "Evaluation methods for topic models". In: *Proceedings of the 26th annual international conference on machine learning*. 2009, pp. 1105–1112.

[16] Malcom Gethers and Denys Poshyvanyk. "Using relational topic models to capture coupling among classes in object-oriented software systems". In: *2010 IEEE international conference on software maintenance*. IEEE. 2010, pp. 1–10.

[17] Arif E Jinha. "Article 50 million: an estimate of the number of scholarly articles in existence". In: *Learned publishing* 23.3 (2010), pp. 258–263.

[18] Bing Liu et al. "Identifying functional miRNA–mRNA regulatory modules with correspondence latent dirichlet allocation". In: *Bioinformatics* 26.24 (2010), pp. 3105–3111.

[19] V Srividhya and R Anitha. "Evaluating preprocessing techniques in text categorization". In: *International journal of computer science and application* 47.11 (2010), pp. 49–51.

[20] Liangjie Hong, Ovidiu Dan, and Brian D Davison. "Predicting popular messages in twitter". In: *Proceedings of the 20th international conference companion on World wide web*. 2011, pp. 57–58.

[21] Christian S. Perone. "Machine Learning :: Text feature extraction (tf-idf) – Part II". In: Terra Incognita, Oct. 2011.

[22] Wen Zhang, Taketoshi Yoshida, and Xijin Tang. "A comparative study of TF* IDF, LSI and multi-words for text classification". In: *Expert systems with applications* 38.3 (2011), pp. 2758–2765.

[23] Tse-Hsun Chen et al. "Explaining software defects using topic models". In: *2012 9th IEEE working conference on mining software repositories (MSR)*. IEEE. 2012, pp. 189–198.

[24] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. "Density-based clustering based on hierarchical density estimates". In: *Pacific-Asia conference on knowledge discovery and data mining*. Springer. 2013, pp. 160–172.

[25] Raviv Cohen and Derek Ruths. "Classifying political orientation on Twitter: It's not easy!" In: *Proceedings of the International AAAI Conference on Web and Social Media*. Vol. 7. 1. 2013, pp. 91–99.

[26] Zhengxing Huang, Xudong Lu, and Huilong Duan. "Latent treatment pattern discovery for clinical processes". In: *Journal of medical systems* 37.2 (2013), pp. 1–10.

[27] Tomas Mikolov et al. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems* 26 (2013).

[28] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[29] Noam Slonim, Ehud Aharoni, and Koby Crammer. "Hartigan's K-means vs. Lloyd's K means–is it time for a change?" In: *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*. 2013.

[30] Xiaohui Yan et al. "A biterm topic model for short texts". In: *Proceedings of the 22nd international conference on World Wide Web*. 2013, pp. 1445–1456.

[31] Goutam Chakraborty, Murali Pagolu, and Satish Garla. *Text mining and analysis: practical methods, examples, and case studies using SAS*. SAS Institute, 2014.

[32] Xueqi Cheng et al. "Btm: Topic modeling over short texts". In: *IEEE Transactions on Knowledge and Data Engineering* 26.12 (2014), pp. 2928–2941.

[33] Ricardo JGB Campello et al. "Hierarchical density estimates for data clustering, visualization, and outlier detection". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 10.1 (2015), pp. 1–51.

[34] Justin Eldridge, Mikhail Belkin, and Yusu Wang. "Beyond hartigan consistency: Merge distortion metric for hierarchical clustering". In: *Conference on Learning Theory*. PMLR. 2015, pp. 588–606.

[35] Joseph Lilleberg, Yun Zhu, and Yanqing Zhang. "Support vector machines and word2vec for text classification with semantic features". In: *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*. IEEE. 2015, pp. 136–140.

[36] Michael Röder, Andreas Both, and Alexander Hinneburg. "Exploring the space of topic coherence measures". In: *Proceedings of the eighth ACM international conference on Web search and data mining*. 2015, pp. 399–408.

[37] Prafulla Bafna, Dhanya Pramod, and Anagha Vaidya. "Document clustering: TF-IDF approach". In: *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*. IEEE. 2016, pp. 61–66.

[38] Aleksandr Drozd, Anna Gladkova, and Satoshi Matsuoka. "Word embeddings, analogies, and machine learning: Beyond king-man+ woman= queen". In: *Proceedings of coling 2016, the 26th international conference on computational linguistics: Technical papers*. 2016, pp. 3519–3530.

[39] Fabrizio Esposito, Anna Corazza, and Francesco Cutugno. "Topic Modelling with Word Embeddings." In: *CLiC-it/EVALITA*. 2016.

[40] Marıa del Mar Roldán Garcıa, José Garcıa-Nieto, and José F Aldana-Montes. "An ontology-based data integration approach for web analytics in e-commerce". In: *Expert Systems with Applications* 63 (2016), pp. 20–34.

[41] Ye Liu et al. "Fortune teller: predicting your career path". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 30. 1. 2016.

[42] Hesam Amoualian et al. "Topical Coherence in LDA-based Models through Induced Segmentation". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 1799–1809. DOI: 10.18653/v1/P17-1165. URL: https://aclanthology.org/P17-1165.

[43] Leland McInnes, John Healy, and Steve Astels. "hdbscan: Hierarchical density based clustering." In: *J. Open Source Softw.* 2.11 (2017), p. 205.

[44] Daniel Preoţiuc-Pietro et al. "Beyond binary labels: political ideology prediction of twitter users". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2017, pp. 729–740.

[45] Shaheen Syed and Marco Spruit. "Full-text or abstract? examining topic coherence scores using latent dirichlet allocation". In: *2017 IEEE International conference on data science and advanced analytics (DSAA)*. IEEE. 2017, pp. 165–174.

[46] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[47] Yin Zhang et al. "iDoctor: Personalized and professionalized medical recommendations based on hybrid matrix factorization". In: *Future Generation Computer Systems* 66 (2017), pp. 30–35.

[48] Dhruvil Karani. "Introduction to Word Embedding and Word2Vec". In: Sept. 2018.

[49] Kamal Kumar. "Evaluation of Topic Modeling: Topic Coherence". In: May 2018.

[50] Changzhou Li et al. "LDA meets Word2Vec: a novel model for academic abstract clustering". In: *Companion proceedings of the the web conference 2018*. 2018, pp. 1699–1706.

[51] Leland McInnes, John Healy, and James Melville. "Umap: Uniform manifold approximation and projection for dimension reduction". In: *arXiv preprint arXiv:1802.03426* (2018).

[52] Joel Nothman, Hanmin Qin, and Roman Yurchak. "Stop Word Lists in Free Open-source Software Packages". In: *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 7–12. DOI: 10.18653/v1/W18-2502. URL: https://aclanthology.org/W18-2502.

[53] Dipanjan Sarkar. "Implementing Deep Learning Methods and Feature Engineering for Text Data: The Continuous Bag of Words (CBOW)". In: Apr. 2018.

[54] "An Introduction to Web Scraping for Research". In: *Research Data services* (2019).

[55] Aditya Anantharaman et al. "Performance evaluation of topic modeling algorithms for text classification". In: *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*. IEEE. 2019, pp. 704–708.

[56] Hamed Jelodar et al. "Latent Dirichlet allocation (LDA) and topic modeling: models, applications, a survey". In: *Multimedia Tools and Applications* 78.11 (2019), pp. 15169–15211.

[57] Patrick van Kessel. "Interpreting and validating topic models". In: Aug. 2019.

[58] Fedor Krasnov and Anastasiia Sen. "The Number of Topics Optimization: Clustering Approach". In: *Machine Learning and Knowledge Extraction* 1 (Jan. 2019), pp. 416–426. DOI: 10.3390/make1010025.

[59] Federico Pascual. "Topic Modeling: An Introduction". In: MonkeyLearn, Sept. 2019.

[60] Nils Reimers and Iryna Gurevych. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. 2019. DOI: 10.48550/ARXIV.1908.10084. URL: https://arxiv.org/abs/1908.10084.

[61] Rubina F Rizvi et al. "Analyzing social media data to understand consumer information needs on dietary supplements". In: *Studies in health technology and informatics* 264 (2019), p. 323.

[62] Mohd Sanad. "Demystifying BERT: A Comprehensive Guide to the Groundbreaking NLP Framework". In: Sept. 2019.

[63] Muzafar Rasool Bhat et al. "Deep LDA: A new way to topic model". In: *Journal of Information and Optimization Sciences* 41.3 (2020), pp. 823–834.

[64] KR1442 Chowdhary. "Natural language processing". In: *Fundamentals of artificial intelligence* (2020), pp. 603–649.

[65] Stephan A Curiskis et al. "An evaluation of document clustering and topic modelling in two online social networks: Twitter and Reddit". In: *Information Processing & Management* 57.2 (2020), p. 102034.

[66] Maarten Grootendorst. "Topic Modeling with BERT". In: Oct. 2020.

[67] Ioana. "Latent Dirichlet Allocation: Intuition, math, implementation and visualisation with pyLDAvis". In: Sept. 2020.

[68] Priyanshu Jain. "Unsupervised Machine Learning: Validation Techniques". In: Guavus, Jan. 2020.

[69] Renu Khandelwal. "Intuitive Explanation of BERT- Bidirectional Transformers for NLP". In: Apr. 2020.

[70] Suhyeon Kim, Haecheong Park, and Junghye Lee. "Word2vec-based latent semantic analysis (W2V-LSA) for topic modeling: A study on blockchain technology trend analysis". In: *Expert Systems with Applications* 152 (2020), p. 113401.

[71] Mauro Di Pietro. "Text Classification with NLP: Tf-Idf vs Word2Vec vs BERT". In: July 2020. DOI: https://towardsdatascience.com/text-classification-with-nlp-tf-idf-vs-word2vec-vs-bert-41ff868d1794.

[72] Nils Reimers and Iryna Gurevych. "Making monolingual sentence embeddings multilingual using knowledge distillation". In: *arXiv preprint arXiv:2004.09813* (2020).

[73] Manmohan Singh. "Stop the Stopwords using Different Python Libraries". In: Towards AI, Apr. 2020. DOI: https://pub.towardsai.net/stop-the-stopwords-using-different-python-libraries-ffa6df941653.

[74] Abeer Abuzayed and Hend Al-Khalifa. "BERT for Arabic topic modeling: an experimental study on BERTopic technique". In: *Procedia Computer Science* 189 (2021), pp. 191–194.

[75] Adem Akdogan. "Word Embedding Techniques: Word2Vec and TF-IDF Explained". In: July 2021. DOI: https://towardsdatascience.com/word-embedding-techniques-word2vec-and-tf-idf-explained-c5d02e34d08.

[76] "Brief History of Web scraping". In: May 2021.

[77] Dylan Castillo. "How to Cluster Documents Using Word2Vec and K-means". In: Jan. 2021.

[78] Amaury Faure. "An introduction to BERT". In: May 2021.

[79] Maarten Grootendorst. "Interactive Topic Modeling with BERTopic". In: Jan. 2021.

[80] Michał Marcińczuk et al. "Text Document Clustering: Wordnet vs. TF-IDF vs. Word Embeddings". In: *Proceedings of the 11th Global Wordnet Conference*. University of South Africa (UNISA): Global Wordnet Association, Jan. 2021, pp. 207–214. URL: https://aclanthology.org/2021.gwc-1.24.

[81] Eric Oti et al. "Comprehensive Review of K-Means Clustering Algorithms". In: *International Journal of Advances in Scientific Research and Engineering* 07 (Jan. 2021), pp. 64–69. DOI: 10.31695/IJASRE.2021.34050.

[82] Neha Seth. "Part 2: Topic Modeling and Latent Dirichlet Allocation (LDA) using Gensim and Sklearn". In: Analytics Vidhya, June 2021.

[83] Silvia Terragni et al. "Octis: comparing and optimizing topic models is simple!" In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. 2021, pp. 263–270.

[84] Vatsal. "Word2Vec Explained". In: July 2021.

[85] Enes Zvornicanin. "When Coherence Score is Good or Bad in Topic Modeling?" In: Baeldung, Dec. 2021.

[86] Roman Egger and Joanne Yu. "A Topic Modeling Comparison Between LDA, NMF, Top2Vec, and BERTopic to Demystify Twitter Posts". In: *Frontiers in Sociology* 7 (2022).

[87] Maarten Grootendorst. "BERTopic: Neural topic modeling with a class-based TF-IDF procedure". In: *arXiv preprint arXiv:2203.05794* (2022).

[88] Joao Pedro. "Understanding Topic Coherence Measures". In: Jan. 2022.

[89] Poonam Vijay Tijare and Jhansi Rani Prathuri. "Correlation Between K-means Clustering and Topic Modeling Methods on Twitter Datasets". In: *Cyber Security and Digital Forensics*. Ed. by Kavita Khanna, Vania Vieira Estrela, and Joel José Puga Coelho Rodrigues. Singapore: Springer Singapore, 2022, pp. 459–477. ISBN: 978-981-16-3961-6.

[90] L. Vailshery. "Number of new papers related to edge computing published on Google Scholar from 2010 to 2021". In: Aug. 2022. DOI: https://www.statista.com/statistics/1193762/worldwide-edge-computing-google-scholar-paper/.

[91] STM An. "STM Global Brief 2021–Economics & Market Size". In: (). DOI: stm-assoc.org/wp-content/uploads/2021_10_19_STM_Global_Brief_2021_Economics_and_Market_Size-1.pdf.