



Universidad Politécnica  
de Madrid

**Escuela Técnica Superior de  
Ingenieros Informáticos**



Master in Data Science

Master Thesis

# **Topic Modeling for Research Software**

*Author:* María Ayuso  
*Supervisor:* Oscar Corcho  
*Co-supervisor:* Daniel Garijo

Madrid, June - 2022

This Master Thesis has been deposited in ETSI Informáticos de la Universidad Politécnica de Madrid.

*Master Thesis*

*Master in Data Science*

*Title:* Topic Modeling for Research Software  
June - 2022

*Author:* María Ayuso  
*Supervisor:* Oscar Corcho  
*Cosupervisor:* Daniel Garijo  
Artificial Intelligence Department  
ETSI Informáticos  
Universidad Politécnica de Madrid

# Resumen

Actualmente la cantidad de publicaciones diarias en las diferentes áreas del Aprendizaje Automático imposibilita a investigadores el estar al día e incluso encontrar lo que están buscando en un límite razonable de tiempo. Este problema también se puede aplicar al desarrollo del software, puesto que la reutilización de código previamente desarrollado también les podría ahorrar mucho tiempo de investigación.

En este trabajo se proponen diferentes métodos para procesar grandes cantidades de publicaciones junto con su software, basados en los resúmenes obtenidos de Papers With Code y sus correspondientes archivos README; con el objetivo de ahorrar trabajo a los investigadores y facilitarles la adopción de software. Este enfoque se basa en el modelado de tópicos para extraer los principales temas presentes y asignarlos a las distintas publicaciones sin tener que leerlas todas. Utilizamos tres algoritmos para encontrar los temas más coherentes. LDA es el algoritmo más utilizado para modelización de tópicos en un conjunto de textos y ha demostrado ser muy eficaz. A pesar de esto, también presenta ciertas limitaciones. Para cubrir su limitación a la hora de trabajar con documentos de poca longitud, proponemos aplicar también BTM, que fue diseñado específicamente para este tipo de textos. LDA también puede encontrar problemas para manejar las relaciones semánticas. Para ello utilizamos BERTopic, que utiliza la habilidad de modelos del lenguaje preentrenados para encontrar representaciones mejores y contextualizadas del texto.

Los resultados corroboran la supuesta eficacia del algoritmo LDA, a la vez que enfatizan la influencia del ajuste de hiperparámetros. BERTopic también proporciona tópicos coherentes e interpretables. En general, concluimos que nuestros modelos identifican efectivamente la mayoría de los tópicos y podrían ser útiles para futuras implementaciones de búsqueda de temas o sistemas de recomendaciones.



# Abstract

Currently, the amount of daily publications in different fields of Machine Learning makes it impossible for researchers to be up to date and even to find what they're looking for in a reasonable amount of time. This problem can be extended also for software developing, where reusing already developed code could save them up much research time.

In this thesis, we propose different methods for processing a large number of papers along with their software, based on abstracts obtained from Papers With Code and their corresponding README files, in order to save researchers time and facilitate software adoption . This approach relies on topic modeling to extract the main topics present and assign them to the different publications without having to read them all. We use three different algorithms to find the most coherent topics. LDA is the algorithm most used when modelling topics within text, and has proven to be effective although it also has some limitations . To cover its limitation on working with documents of short length we also propose applying BTM, which was specifically designed for these types of text . LDA can also find problems to handle semantic relationships. For this we used BERTopic, which uses the hability of pretained language models to find better and contextualized text representations.

Results corroborate the assumed effectiveness of LDA, while highlighting the influence of hyperparameter tuning. BERTopic also provided coherent and interpretable topics. Generally, we conclude that our models effectively identify most of the topics and could be useful for future topic search implementations or recommendation systems.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Research questions . . . . .	2
1.2	Structure of the document . . . . .	2
<b>2</b>	<b>State of the Art</b>	<b>5</b>
<b>3</b>	<b>Methods</b>	<b>9</b>
3.1	LDA . . . . .	9
3.2	BTM . . . . .	11
3.3	BERTopic . . . . .	12
<b>4</b>	<b>Experimental setup</b>	<b>15</b>
4.1	Datasets . . . . .	15
4.1.1	Data collection . . . . .	15
4.1.2	Preprocess . . . . .	16
4.2	Implementation . . . . .	18
4.3	Hyperparameter tuning . . . . .	19
4.4	Evaluation . . . . .	20
<b>5</b>	<b>Results</b>	<b>23</b>
<b>6</b>	<b>Conclusion and Future Work</b>	<b>31</b>
	<b>Bibliography</b>	<b>37</b>





# Chapter 1

## Introduction

Software development is increasingly a key part of the process of scientific research. On average, scientists spend approximately 30% of their work time developing scientific software [1]. This type of software is designed specifically to support research. In many cases, this process could be streamlined by taking advantage of the software that is already published. However, finding and comparing related software is a complex task due to the large amount of already developed software.

Nowadays there's a big encouragement to contribute to open source software (OSS). This type of software stands out for the control, stability and transparency that it provides to its users. Open source continues to expand not only by individual developers, but also is the recent move by software giants such as Microsoft, IBM, and Oracle. On 2015, the Future of Open Source Survey showed that already 78% of its respondents said their companies run part or all of its operations on OSS [2]. In spite of this figure, another survey [1] showed that 10.7% of the respondents believe that the most important software they use has less than 3 users worldwide. This contrast leads to think that the efforts put into open source for reusing and recycling software are not being fully exploited.

One could assume that the decision taking of what software to implement is driven by scientific reasons, which could lead to little amount of softwares to gather the most followers. Surprisingly, many scientists report to follow subjective perceptions on software adoption, they tend to rely on how easy it is to manipulate, if it has appeared in a peer-reviewed article and recommendations [3].

In order to compare scientific publications and their corresponding developed software, we propose a topic modeling approach. Topic modeling is a data mining task that refers to discovering the distribution of underlying topics of a collection of documents. This unsupervised machine learning technique is useful to understand and categorize large corpora of text. While the mathematics behind the algorithms are complex, we can explain a general overview of how a topic model works. Imagine having a collection of articles without any label, and wanting to categorize them without going through the tedious task of catalogue them by hand, one by one. Instead, a topic model treats each document as a bag-of-words and finds co-occurrence patterns in order to group certain words. This word groups will form the different topics. By doing so, the model is able to find the probability of a certain topic to be present in a specific document.

## **1.1 Research questions**

Motivated by the possibilities of this text analysis method, we analyze different parts of science publications, including parts of software. The goal is to detect commonalities among different tools based on their documentation, first on abstracts and then on README files. Abstracts are short summaries included at the beginning of research papers, therefore we assume these could have useful information for our research purpose. README files could be considered to software the same as abstracts to papers, these files include information for the user about the software such as description, requirements or installation. As we said, finding similarities among these types of documents could be useful for future researchers to find the best fit for their projects, relying on a scientific base and increasing the sustainability of scientific-software-development-and-use cycle. In short, we will try to answer the following research questions:

- **RQ1** Does topic modeling yield to coherent and interpretable topics on research based on different paper's abstracts? Do the results of any of the applied algorithms stand out?
- **RQ2** Can we achieve or even improve the results by applying topic modeling to README files? Does the same algorithm perform the best with these type of data?

Since our data, as will be explained in more depth later in the thesis, corresponds to publications already labeled, we will study the goodness of the models based on the categories we already have.

On the one hand, we will study the semantic coherence on the topics obtained using as input abstracts from different papers. Additionally, we will compare the results obtained with the taxonomies classified by hand by users provided by Papers With Code [4]. On the other hand, we will apply the same algorithms to README files and study analogously its coherence. Finally, we will compare if using different input data provides similar results since each paper is directly associated with its README file.

## **1.2 Structure of the document**

The content of this document is divided in six chapters, describing the different steps followed in this study.

In Chapter 2 the history of probabilistic approaches for topic modeling is introduced, with the motivation of each new algorithm that was developed. Additionally, two models are presented that tackle the possible semantic and sparsity problems that probabilistic methods may find. After this brief introduction to the algorithms that will be applied throughout the study different applications of topic models are named.

In Chapter 3 the assumptions and description of the different models are exposed. In the case of LDA and BTM the generative processes are explained with their corresponding plate models. For the BERTopic algorithm, its pipeline is described with a brief explanation of the methods included.

Once the foundations of the models have been explained, in Chapter 4 the process

## Introduction

---

followed for their implementation is explained. Firstly, the data collection process along with its following preprocess. Then, the software and libraries used are presented. The procedure of the hyperparameter tuning step and the different values used are identified. Finally, widely known evaluation measures (*UMASS* and *NPMI*) are defined as well as a method for comparing the results with predefined categories.

The results obtained for the different techniques and datasets are provided in Chapter 5. Additionally, in this Chapter an in-depth analysis and comparison of the models is made.

Lastly, in Chapter 6 the main conclusions of the study are presented, providing a summary of the obtained results and a comparison between the best models built for each dataset. It also includes weakness and possible future work.



## Chapter 2

# State of the Art

Topic models are born with Latent Semantic Analysis (LSA), originally known as Latent Semantic Indexing (LSI) [5]. This method was specially designed for information retrieval. “The problem is that users want to retrieve on the basis of conceptual content, and individual words provide unreliable evidence about the conceptual topic or meaning of a document.” [5] To solve this problem, the idea of an underlying latent semantic structure in the data arised, for the first time, instead of term-matching retrieval.

In 2001 Hoffman proposed Probabilistic Latent Semantic Analysis (PLSA), as the name indicates inspired in LSA. The motivation for this probabilistic approach was for once the unsatisfaction of the theoretical foundations of LSA. On the other hand, on a more technical level the deficiency of LSA to handle both synonymy and polysemy [5]. PLSA defines a proper generative model to perform a probabilistic mixture decomposition. PLSA models each word in a document as a sample from a mixture model, where the mixture components are multinomial random variables that can be viewed as representations of “topics” [6]. Thus, each word is generated by a single topic and each document is a proportion of this set of topics.

Though this was a big step for probabilistic modeling, it was still missing a probabilistic model at the document level. Both LSI and PLSA are based on the bag-of-words assumption, i.e the exchangeability of words in a document. Following the bag-of-words each document is represented as a fixed-length vector with length equal to the size of the vocabulary. Each element of this vector is associated with a word and contains the occurrence of said word in the document. For example, let’s suppose a vocabulary defined as  $V = \{\text{today, tomorrow, is, monday, and, rainy, sunny}\}$  and the following document “Today is monday and today is rainy.” The representation of this document based on bag-of-words would be  $d = [2, 0, 2, 1, 1, 1, 0]$ . Note that in this representation word order is ignored. Latent Dirichlet Analysis (LDA) introduces the idea of exchangeability of documents as well. “LDA posits that each word of both the observed and unseen documents is generated by a randomly chosen topic which is drawn from a distribution with a randomly chosen parameter.” [7] Although LDA is the most popular topic modeling method for discovering the underlying topics from text-corpus [8], in [7] Blei et al. already showed its limitations. In particular, the bag-of-words assumption can lead to certain words that go along together and should be generated by the same topic, to fall into different ones.

---

To overcome the disregard of semantic relationships among words by these probabilistic approaches, word embedding techniques have also become popular in the natural language processing field. This NLP methodology is used to map words and phrases to vector of real numbers that are able to capture similarities between words [9]. Additionally, in order to lower the computational complexity clustering approaches on both word [10] and document [11] embeddings have been proposed. In 2021 BERTopic was created, this algorithm builds on top of the clustering embeddings approach and extends it by incorporating a class-based variant of Term Frequency Inverse Document Frequency (TF-IDF) for creating topic representations [12]. TF-IDF works by determining the relative frequency of words in a specific document compared to the inverse proportion of that word over the entire document corpus [13]. BERTopic was created to take advantage of the amazing results that BERT pre-trained language representations have shown. BERT uses a bidirectional transformer so that representations are jointly conditioned on both left and right context in all layers [14]. Although it was created specially for this type of embeddings, BERTopic allows a wide range of tuning, among these possibilities is custom embeddings. This customization is part of what makes BERTopic significant, since its performance will continue to grow with the current state-of-the-art embedding techniques.

Another inconvenience we may find when applying the popular probabilistic approaches (LDA, PLSA) is the sparsity of content found in short texts (in our study README files). The word co-occurrence information at individual document level becomes much sparser, inevitably compromising the performance of topic models that are based on this information. According to Lin et al. [15] probabilistic models perform sub-optimally in said situations even when hyperparameters are optimized. There have been many attempts at solving this problem such as text aggregation [16], or making strong assumptions like all words in each sentence or document to share the same topic ([17], [18]).

Biterm Topic Modeling (BTM) [19] is born due to the huge rise of short text documents and as we said the lack of representation of more popular topic modeling methods of these types of text. BTM learns topics over short texts by directly modeling the generation of biterms in the whole corpus. By using the word co-occurrence patterns in the whole corpus, instead of at document-level it avoids the problem of pattern sparsity. Their experimental results show that BTM produces discriminative topic representations as well as more coherent topics for short texts.

Nowadays, massive quantities of information is generated daily on many fields. The volume and complexity of the data have outstripped researchers' ability to keep up with the progress of science, even in their chosen fields. For this reason, text-mining tools have been developed to extract facts from scientific papers in different fields. In the biological researcher field, there has been effort in entity recognition and relationship extraction through abstracts, for example in [20] and [21]. In [22] topic modeling on abstracts was applied to examine research trends in the area of transportation. This techniques have not only been used on abstracts, in [23] topic modeling was implemented on several research papers keywords in order to detect their topics.

In the software engineer field topic modeling has been used many times. A study conducted by the University of New Zealand [24] showed that in this field source code, issue/bug reports and developer communication were the most frequent types of data used. However, the closest instances to software classification were related to

bug handling, such as software maintenance ([25], [26]) and defects prediction ([27], [28]).

Related to our approach to README files, Yu et al. performed topic modeling for third-party libraries recommendation on Android Mobile Apps [29]. Their goal was to bridge the gap between the large amount of libraries available and the developers need to use them, in like manner to our approach with Github software and researchers. Similarly, in [30] a multi-label classifier was designed to predict eight different categories for different README sections. Their goal was to enable the owners of software repositories to improve the quality of their documentation and make it easier for the software development community to discover relevant information in GitHub README files. This proposal eases the process of discovering where is the relevant information about software like what it is for or how to install it. However, it lacks the discovery of the main topic of each repository that we propose.





# Chapter 3

## Methods

### 3.1 LDA

LDA enables the processing of large scale collections of text in order to understand what they are about in minutes. This algorithm finds short descriptions of the data subjects while conserving statistical relationships for different tasks such as classification, summarization or similarity detection.

LDA is a three level generational model that assigns a probability distribution to almost every aspect on the language model. Before diving more in depth into what the algorithm actually does we define some basic concepts that will be used. First, a word is the basic unit of individual data. Words will be denoted as  $w$  and will also be referred to as ‘terms’ or ‘tokens’. In this context, a document is defined as sequence of  $N$  words that is usually represented as a vector  $d = (w_1, w_2, \dots, w_N)$  where  $w_n$  is the  $n$ th word present. A corpus is defined as a set of  $M$  documents that encompasses the whole dataset. Knowing this terms, in LDA each document is modeled as a finite mixture over a set of topics. Each document’s topic distribution is randomly sampled from a Dirichlet distribution with hyperparameter  $\alpha$ , and each topic’s word distribution is randomly sampled from a Dirichlet distribution with hyperparameter  $\beta$ . The generative process of the LDA model is the following:

1. For each topic  $k \in [1, K]$ :
  - (a) Draw a topic-word distribution  $\phi_k \sim \text{Dirichlet}(\beta)$
2. For each document  $d$  in a corpus  $D$ :
  - (a) Draw the number of words  $N_d \sim \text{Poisson}(\xi)$
  - (b) Draw a document-topic distribution  $\theta_d \sim \text{Dirichlet}(\alpha)$
  - (c) For each word  $w_i$  in  $d$ :
    - i. Draw a topic assignment  $z_{d,i} \sim \text{Multinomial}(\theta_d)$
    - ii. Draw a word  $w_{d,i} \sim \text{Multinomial}(\phi_{z_i})$

Plate models are a graphical way to represent the relations between random variables. The generative process explained above is presented by the plate model in Figure 3.1. Each circle represents a random variable. If it has a white background,

then the variable is hidden, while if it's grey, it's observable. The arrows represent dependencies between these variables, and the boxes, also known as plates, mean that the process is repeated. In the lower right corner of each plate we can see the number of iterations. In this Figure we can see clearly represented the three levels of this model. First,  $\alpha$  and  $\beta$  are corpus-level variables and assumed to be sampled only once. The variables  $\theta_d$  are document-level variables, as declared on the outer plate sampled  $M$  times, once per document. On the third level,  $z_{d,i}$  and  $w_{d,i}$  are word-level variables and are sampled  $N_d$  times per document, with  $N_d$  being the number of words in document  $d$ .

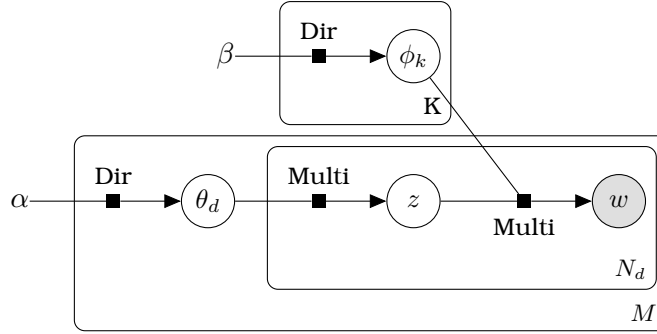


Figure 3.1: Graphical model representation of LDA

The key problem in topic modeling is computing the posterior distribution of the hidden variables given the observed data. Particularly, in LDA this involves solving:

$$p(\theta, \mathbf{z} | \mathbf{w}, \alpha, \beta) = \frac{p(\theta, \mathbf{z}, \mathbf{w} | \alpha, \beta)}{p(\mathbf{w}, | \alpha, \beta)} \quad (3.1)$$

Unfortunately, this distribution is intractable for exact inference therefore is necessary to use an approximate inference algorithm. In the original paper of LDA a simple convexity-based variational approach for inference is explained in depth. However, it's also mentioned that different approximate inference techniques could be considered such as Laplace approximation, higher-order variational techniques, and Monte Carlo methods [7]. In this study we used *models.ldamulticore* from Python's *gensim* package. This function employs computationally-efficient online variational Bayesian optimization as a learning algorithm [31].

The first and most deterministic hyperparameter in LDA is  $K$ , the number of topics. However, the parameters of the prior distribution over topic weights in each document ( $\alpha$ ) and over word weights in each topic ( $\beta$ ) are also significant. A higher value of  $\alpha$  results in a higher number of topics that are relatively dominant in each document, whereas a lower value places more weight on having each document composed of only a few dominant topics. Being able to identify multiple topics within a document is one the strengths of LDA, so it would be a shame to lose this by setting the  $\alpha$  parameter too low. However, if the topics are distributed too uniformly across the document we might not get very meaningful information so it's important to find a balance. Analogously, for a higher value of  $\beta$  the representation of each topic is evenly distributed across multiple words. And a lower value leads to little number of words being very representative of a certain topic.

### 3.2 BTM

LDA models documents as a mixture of topics, where each topic is considered as a probability distribution over words. However, when working with short texts some problems could appear due to the sparseness of word co-occurrence patterns and lack of context's richness. As a result, applying LDA to short texts can yield to suboptimal topics even when the hyperparameters are optimized [15]. In 2013 Yan, Guo, Lan, et al. proposed Biterm Topic Modeling [19] which instead of modeling the generation of documents, models the generation of biterms.

BTM learns topics by directly modeling the generation of word co-occurrence patterns (i.e., biterms) in the whole corpus, making the inference effective with the rich corpus-level information. A biterm is defined as an unordered word-pair co-occurring in a short context. In this matter, we considered each document as an individual context unit. This means that any two distinct words in the same document form a biterm. For example if one document consisted only in the sentence "Today is Monday" the biterms formed would be:

(Today,is) (Today, Monday) (is, Monday)

After extracting the different biterms of each document, the corpus turns into a biterm set ( $B$ ). The generative process of BTM can be described as follows:

1. For each topic  $k \in [1, K]$ 
  - (a) Draw a topic-word distribution  $\phi_k \sim \text{Dirichlet}(\beta)$
2. Draw a topic distribution  $\theta \sim \text{Dirichlet}(\alpha)$  for the whole corpus
3. For each biterm  $b_i$  in a biterm set  $B$ 
  - (a) Draw a topic assignment  $z_i \sim \text{Multinomial}(\theta)$
  - (b) Draw two words  $w_i, w_j \sim \text{Multinomial}(\phi_{z_i})$

Analogously to LDA, we can observe this process graphically in the plate model in Figure 3.2. In this case,  $\theta$  is only sample one in the whole process. Also note, that  $w_i$  and  $w_j$  are drawn independently.

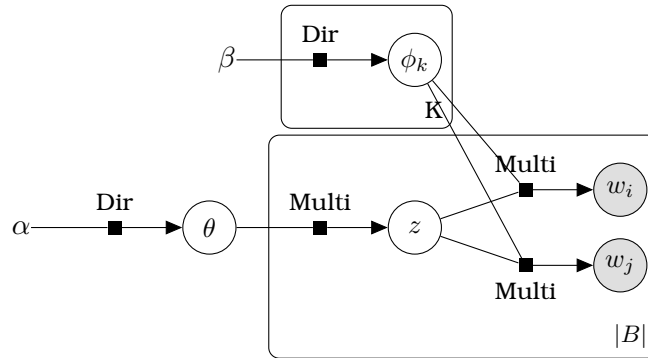


Figure 3.2: Graphical model representation of BTM

Similarly to LDA,  $\alpha$  and  $\beta$  control the distribution of topics and the distribution of words in topics respectively. Since  $\alpha$  and  $\beta$  are hyperparameters of symmetric Dirichlet distributions, a higher  $\alpha$  means that the topic assignments for each biterm will be

more evenly distributed. Similarly for  $\beta$ , a higher value means that more words are likely in each topic. Typically in short texts, we expect very few topics in each document, and the keywords that describe each topic should not be too many. Therefore, fairly low values of  $\alpha$  and  $\beta$  are to be expected [32].

### 3.3 BERTopic

Although probabilistic approaches have proven to be powerful for topic modeling, they disregard semantic relationships among words. Additionally, applying neural networks and deep learning to NLP tasks has brought a major breakthrough. More specifically, Bidirectional Encoder Representations from Transformers (BERT), which was the first to introduce deep bidirectional architectures, allows to tackle successfully numerous NLP tasks [14]. A major difference between the usage of language models and other popular approaches such as PLSA or LDA is that language models take advantage of the context of each word. Particularly, let's suppose that we have two sentences "I park across the street" and "The kids are playing in the park". The word "park" would have the same representation with context-free approaches in both sentences, while BERT would provide different embeddings according to the sentence. On the opposite side, probabilistic approaches don't recognize similarity of words such as big and large unlike language models.

In 2022 BERTopic [12] was proposed to exploit the representations of language models and the demonstrated viability of clustering embeddings to represent topics. These topics are built through four clear steps. First, each document is transformed into its embedding representation using a pre-trained model. Then, a dimensionality reduction technique is applied to optimize the posterior clustering. Finally, instead of using a centroid-based perspective for topic representation as in *top2vec*[11], a custom class-based variation of TF-IDF is applied.

To perform the embedding step, BERTopic uses Sentence-BERT (SBERT). "BERT uses a cross-encoder: Two sentences are passed to the transformer network and the target value is predicted. However, this setup is unsuitable for various pair regression tasks due to too many possible combinations." [33] To overcome this computational issue SBERT adds a pooling operation to the output of BERT to derive a fixed sized sentence embedding. In this step we found the first "hyperparameter" of this algorithm: the pre-trained embedding model.

After building the embeddings, the dimensionality reduction step allows for dense clusters of documents to be found more efficiently and accurately in the reduced space. Although t-SNE and PCA are popular techniques for reducing dimensionality, UMAP has shown to preserve better both local and global features from high-dimensional data into lower dimensions and provides better scaling [34]. This step brings 4 new hyperparameters: number of nearest neighbors, minimum distance, number of components and metric. The number of nearest neighbors controls the balance between preserving global structure versus local structure in the low dimensional embedding. The minimum distance controls how tightly UMAP is allowed to pack points together. The number of components controls the dimensionality of the resulting reduced dimension space that the data is embedded into. Finally, the metric controls how distance is computed in the ambient space of the input data.

The reduced embeddings are clustered using Hierarchical Density-Based Spatial Clus-

tering of Applications with Noise (HDBSCAN) [35], a hierarchical density-based clustering technique. This algorithm excels at finding clusters with different densities and sizes and even when there's noise or outliers. Essentially it looks for regions of the data that are denser than the surrounding space. The dense areas of identified document vectors will be used to calculate the topic vectors. A special characteristic of this algorithm, different from other clustering techniques like K-Means or agglomerative clustering, is that it allows data to be classified as noise. Such documents that are classified as noise can be seen as not being descriptive of any prominent topic. This algorithm has one main hyperparameter, minimum cluster size. This parameter represents the smallest size that should be considered a cluster by the algorithm.

Finally, in order to have a proper topic representation, a class-based variation of Term Frequency Inverse Document Frequency (TF-IDF) is used. TF-IDF works by determining the relative frequency of words in a specific document compared to the inverse proportion of that word over the entire document corpus [13]. Intuitively, this calculation represents the importance of a word to a document. Similarly, in BERTopic the most representative words for each topic is desired. For this purpose, all documents that fall in a specific cluster are merged together to form that cluster's class. Since each cluster obtained is considered a topic, the importance of the different terms for each topic is obtained as:

$$W_{t,c} = tf_{t,c} \times \log\left(1 + \frac{A}{tf_t}\right) \quad (3.2)$$

where  $t$  is a term in the corpus,  $c$  a class of a specific cluster,  $tf$  the term frequency and  $A$  the average words per class.

Note that this last step gives us the top words per topic, however, in contrast with LDA and BTM, BERTopic assumes that each document is represented by a single topic. To deal with this issue, it has been proposed to consider the results of HDBSCAN [12]. This algorithm is a soft-clustering technique, which means that each document is assigned a probability for each cluster, i.e the strength of the belonging to each cluster. Therefore we could take advantage of its resulting probability matrix as an approximation of the distribution of topics in each document. However, we didn't take this possibility into account in this study since we believe this approach could conduct to misleading results.



## Chapter 4

# Experimental setup

### 4.1 Datasets

#### 4.1.1 Data collection

To answer our research questions we took advantage of the Papers with Code platform [4]. This platform provides free and open Machine Learning papers, code, datasets, methods and evaluation tables. Although we will be applying unsupervised algorithms, we also draw upon the fact that this platform has all papers classified in different areas (Computer Vision, Natural Language Processing, Graphs, etc). As we will explain later, we will take advantage of this classification and use it as an additional “metric” to compare models.

On the web page we downloaded three different datasets which we accessed on the [About section](#) of Papers With Code. The same version of the data used in this thesis is uploaded in Zenodo <sup>1</sup>. All data is licensed under the [CC BY-SA](#) license. These datasets were stored in JSON format and had different properties, however for each we only considered the following:

- Papers with abstracts: *paper\_url*, *title* and *abstract*
- Links between papers and code: *paper\_url*, *repo\_url*
- Methods: *paper*, *collections*

From the first dataset we extracted the abstracts that will be used to answer our first research question. From ‘Links between papers and code’ we used the *repo\_url* values and through Python’s library `PyGithub` we accessed the Github API and retrieved the different README files for each repository. Finally, from the ‘Methods’ dataset we needed both the classification and url of the different papers. These values weren’t stored raw as *abstracts* or *repo\_url* in the other datasets so we had to analyze it more in depth. First the *paper* variable stored two values: *title* and *url*. We used this last value as an ID along with *paper\_url* in the other datasets to merge them. On the other hand, *collections* stored the following values: *collection*, *area\_id* and *area*. This last one corresponds to the categories that we needed. It’s important to note that some instances had more than one *area*, meaning that there could be multilabeled papers,

---

<sup>1</sup><https://doi.org/10.5281/zenodo.6788250>

which will have to be treated after collection. This process and the final structure of our dataset is represented in Figure 4.1.

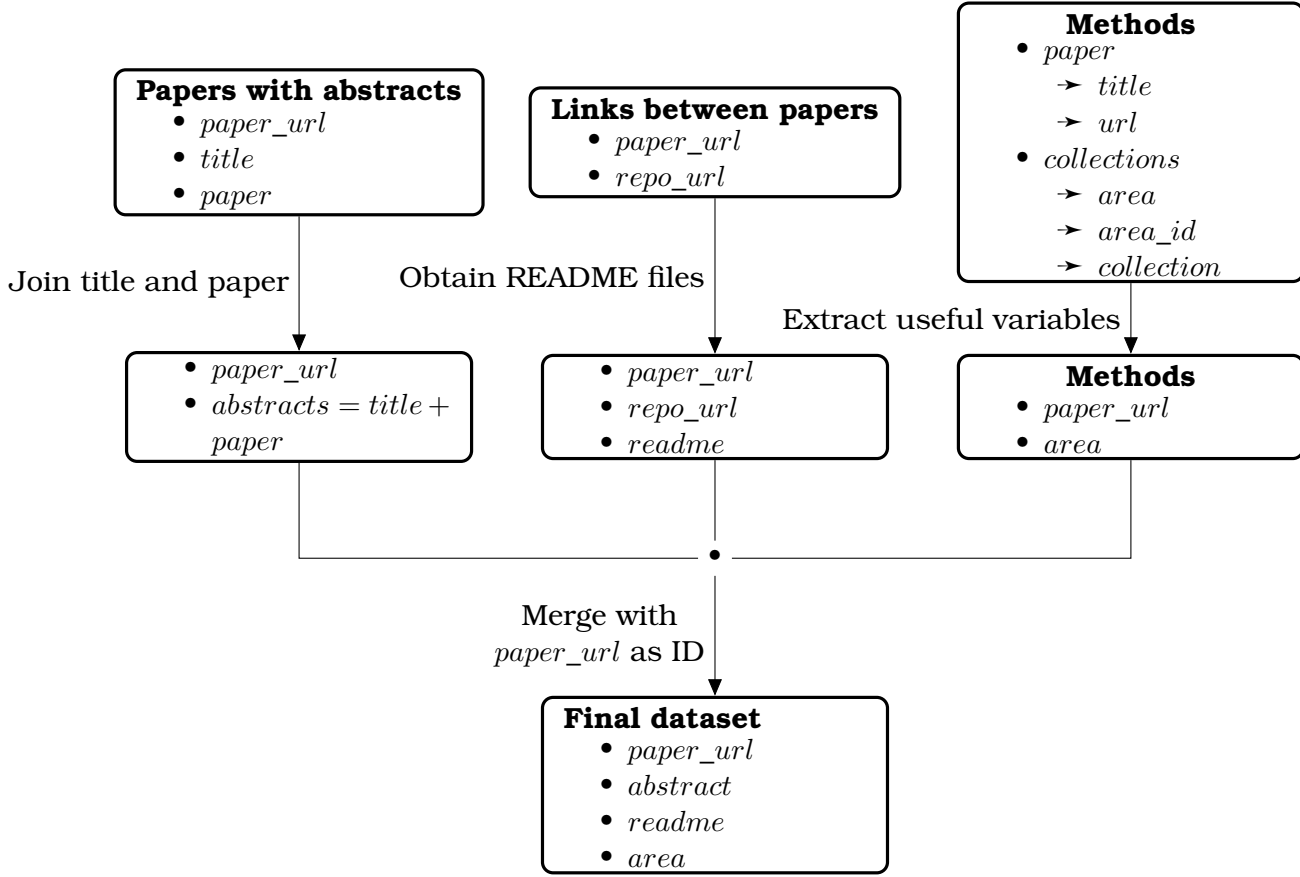


Figure 4.1: Data collection process and final structure

#### 4.1.2 Preprocess

Since our first research question focuses on topic modeling on abstracts we started by studying these. Before applying any topic modeling algorithms we needed to make sure our data was clean of noise to obtain the best results possible. However not all 3 algorithms need the same type of preprocessing, so we will start by explaining what was done for LDA and BTM. First, we noticed there were many abstract duplicates, this was due to different Github repositories to come from the same paper. This can happen for different reasons, for once a single paper may have more than one relevant finding which could be developed in different repositories or even by different users. For example, if one paper discovers two different algorithms, these can be developed in separate repositories. Additionally, one algorithm can be coded in many manners by many users and therefore each manner have a different README file that comes from the same paper. We proceeded to remove these duplicates as well as all instances categorized as 'General'. This last decision was taken looking at how unbalanced the categories were and considering that these instances would include more noise than useful information. From these two steps we came down from 28652 instances to 891. In Table 4.1 we can see the frequency table of the different labels on the final dataset.



## Experimental setup

---

Label	Absolute Frequency	Relative Frequency
Audio	31	0.0348
Computer Vision	524	0.5881
Graphs	81	0.0909
Natural Language Processing	151	0.1695
Reinforcement Learning	51	0.0572
Sequential	53	0.0594

Table 4.1: Frequency table of final dataset

After this huge cut down, we saw that there were some links as well as equations and some more special characters that should be treated, so we proceeded with:

- Removing links
- Removing equations
- Removing common abbreviations
- Removing special characters (for example (, ?, !, etc)
- Removing words with only one character
- Substituting long spaces for one space
- Removing new line characters
- Transforming text to lower case

Most of these steps were done using Python’s module `re` [36]. A regular expression is a sequence of characters that forms a search pattern. This module provides regular expression matching operations, allowing to remove all specific strings stated above. For example, for removing links we matched every string that started by ‘https:’ until a white space.

Stopwords are all words considered both frequent and non-informative for a specific task. The next step after using regular expressions was to filter these type of words. Examples of these words are: and, to, the, it. For this process we used NLTK’s list of 179 words, which can be accessed through its documentation.<sup>2</sup>

Natural Language Toolkit (NLTK) [37] is a library designed specifically to work with human language data. This library also provided us with text libraries for stemming and tokenization. Stemming is the process of cutting the suffixes of all words to the common root. This allowed us to group all variations under a single token and gain more knowledge. Once we obtained the common root of our corpus the last step of the preprocess was tokenization. Both LDA and BTM require as an input the text as a list of words. The process of transforming the abstracts to lists of words is called tokenizing.

In the BERTopic case it’s stated that “by using document embeddings there is typically no need to preprocess the data as all parts of a document are important in understanding the general topic of the document” [14]. However, if the data does

---

<sup>2</sup><https://www.nltk.org/>

contain a lot of noise or specific parts that don't contribute to the meaning of the document, these should be removed. In our case we only removed links, equations, common abbreviations and transformed the text to lower case.

Moving on to our second research question we targeted the README files. In this case we saw there was much more noise than with the abstracts. Again we started by removing the 'General' category and duplicates. In this case we suppose that duplicates come from forked repositories that keep the same README. Additionally, we saw many instances that don't have the README information. We believe this blanks could happen for different reasons like the README file wasn't in the default location or the name wasn't exactly 'README.md'. We also noticed there were some multilabeled instances. For all of these we kept the label with less frequency overall. After this process, our dataset had 8364 instances.

Now, moving on to the actual README data besides of what we did in the preceding section we had to clean more noise. We decided to remove HTML tags (format, images, audio, tables, etc), directory paths, both block and inline code and file names with their respective extension.

We explained before that BERTopic generally does not need preprocess, however we considered that with our data type it was necessary to apply the same extra steps as with LDA and BTM.

## 4.2 Implementation

To implement all different models we chose Python since it is a widely known open-source, high-level programming language that is both powerful and flexible. We worked with the widely used libraries `pandas`[38] and `numpy`[39] for data manipulation and `matplotlib`[40] and `seaborn`[41] for data visualization. For the actual model computations, we relied on different open-source Python packages, making the reproducibility of our results easier. These packages and the process followed are detailed below.

The code can be accessed through the Github repository <sup>3</sup>.

- **gensim** is an increasingly popular topic modeling package. It was designed specifically for managing large corpora and memory efficiency [42]. To build the LDA model we first built the dictionary in terms of the abstract's tokens. The dictionary basically assigns a numerical identifier to each different word. From this dictionary we filtered the "extreme words", those that appeared in less than 2 abstracts or in more than 80% of them. And finally transformed our corpus to a bag-of-words, i.e a list of tuples where each tuple is: (word identifier, word occurrence). Note that what is denoted as  $\beta$  in the LDA algorithm explained in section 3.1 in the gensim package is defined as eta due to a change of notation from the original paper [7] to the online variational bayes approach [31]. This package was also used for computing *UMASS* and *NPMI* metrics which will be defined in a following section.
- **bitermplus** is a package explicitly designed for implementing Biterm topic modeling. This package aims to provide Python bindings for the C code from the

---

<sup>3</sup><https://github.com/mariaayuso/topic-modeling-on-software>

original article [19]. As such, it uses the Gibbs algorithm described in it and includes different tools to evaluate models and analyze results. First we computed the document vs word matrix, dictionary and biterm set. With this data and setting the hyperparameters we fit the model and obtained different results like the top words per topic or the most probable topic for each document.

- **BERTopic** implementation in Python was done along with the original paper [12] and by the same author. This algorithm only needs a list of the documents as input since it doesn't compute any words or biterms distributions.

### 4.3 Hyperparameter tuning

Both LDA and BTM are non-deterministic algorithms, since their internal weights are updated via a stochastic sampling process. This non-determinism leads to great instability on the models [43]. Many papers comment that the results of these probabilistic approaches can be affected by tuning the hyperparameters, despite these concerns, researchers usually lean on “off the shelf” parameters (for example [44], [45], [46]).

To cope with this instability we decided to proceed with some hyperparameter tuning in both LDA and BTM. The number of topics  $K$  ranged between 5 and 14. This decision was made based on the number of human classified labels we had initially. The idea was to set a maximum not too low in order to capture enough information, but not too high to avoid microtopics. However, we also had to deal with the computational inefficiency of applying a exhaustive search over many combination of values for  $\alpha$  and  $\beta$ . For this reason, we followed a process based on the bisection method. We initialized a search of the best model of all combinations of  $\{0.01, 0.1, 10\}$  and  $\{\frac{10}{K}, \frac{20}{K}, \frac{50}{K}\}$  for  $\alpha$  and  $\beta$  respectively. Once we found the best models, we studied what pair of parameters worked best and tried in-between values. For example, let's suppose Table 4.2 was ordered, being model 1 the best and model 5 the worst:

Model	$\alpha$	$\beta$	$K$
1	0.01	$\frac{50}{K}$	8
2	0.1	$\frac{50}{K}$	8
3	0.1	$\frac{20}{K}$	4
4	0.1	$\frac{50}{K}$	10
5	0.01	$\frac{20}{K}$	7

Table 4.2: Example of best models after first round of hyperparameter tuning

Then we could suppose that the best value of  $\alpha$  is in the interval  $[0.1, 0.01]$  and the best value of  $\beta$  in  $[\frac{20}{K}, \frac{50}{K}]$ . In this case, for our second run we would try an additional combination of  $\{0.05\}$  and  $\{\frac{30}{K}, \frac{40}{K}\}$  for  $\alpha$  and  $\beta$ .

Now, focusing on BERTopic we mentioned 6 different hyperparameters. One in the embedding phase, four in the dimensionality reduction and one last one at clustering. Due to our computational and time resources we could not afford to perform fine tuning of all of these hyperparameters. We decided to study more in depth the relevance of the embedding model, since we considered it a major point of inflection in the training process. We considered the default “all-MiniLM-L6-v2”,

"all-mpnet-base-v2" and "allenai-specter". The two first language models were chosen due to their quality and proven performance, on the other hand the last one was chosen due to its specific pretraining to learn embeddings for scientific documents [47].

In general, the other parameters were fixed as follows. Since the goal is to find dense areas of documents which would be close to each other in the high dimensional space, we consider important to emphasize the local structure. Therefore the number of neighbors was set to 15. Minimum distance was set to 0 since this results in potentially densely packed regions [34]. In case of the metric we opted for cosine similarity which is typically used ([48], [49]) due to its ability to capture similarity of documents irrespective of their size. For the number of components (i.e the dimensionality of the embeddings after reducing) the default of 5 is used in order to reduce dimensionality as much as possible while trying to maximize the information kept in the resulting embeddings. Finally, the minimum size of a cluster was set to 10.

In contrast with LDA and BTM, this algorithm does not need to input the number of topics. In some cases, we obtained a model with hundreds of topics. The analysis of these topics would not be congruent with our purpose since the main goal of topic modeling is avoiding the hard labour of inspecting the different documents. For this reason, in case we obtained this large number of topics we did tune either the number of nearest neighbors from UMAP or minimum cluster size in HDBSCAN.

## 4.4 Evaluation

The performance evaluation of a topic model is not trivial. Note, that the first and most important metric is the semantic coherence and interpretability of the topics of each model. Nevertheless, considering the amount of models we had to inspect it's also important to define numerical metrics of the goodness of fit. Perplexity is a popular metric that has been widely used in literature. This value is the probability of held-out data given the settings of the model. However, we won't be using it in this study for two main reasons: the marginal likelihoods of LDA and BTM are not comparable, since LDA optimizes the likelihood of word occurrences in documents, whereas, BTM optimizes the likelihood of bitterms occurrences in the corpus [19]; on the other hand Chang et al. [50] showed that this metric is not correlated with the understandability of the top words within a topic.

As we stated, evaluating topic models is a difficult task as an effort to quantify the interpretability of topics without human evaluation and due to the great variation of implementations and algorithms. However, it is also worth mentioning that the inconsistency of names of the different metrics that have been developed stand in the way of the evaluations efficiency. For example, the Coherence Score defined in [51] or [19] corresponds to the same metric defined as *UMass* coherence in [52] or [53]. Again, what is called *PMI-Score* in [8] references the same metric as *UCI* coherence in [54]. In this study, we will mainly base our metrics on the unifying framework developed in [54], which spans a configuration space of coherence definitions.

Among the available coherence metrics defined in [54], we used *UMass* and *NPMI*. Both of these metrics are used for measuring the quality and semantic relationship of the topics. Also, both have shown to be positively related with human judgement [51], [55]. For a specific topic  $z$  and considering its top  $N$  words  $W^z = \{w_1^z, \dots, w_N^z\}$

## Experimental setup

---

these metrics are defined as follows:

$$UMass(z, W^z) = \frac{2}{N(N-1)} \sum_{i=2}^N \sum_{j=1}^{i-1} \log \frac{P(w_i^z, w_j^z) + \epsilon}{P(w_j^z)} \quad (4.1)$$

$$NPMI(z, W^z) = \frac{2}{N(N-1)} \sum_{i=1}^N \sum_{j=i}^N \frac{\log \frac{P(w_i^z, w_j^z) + \epsilon}{P(w_i^z)P(w_j^z)}}{-\log(P(w_i^z, w_j^z) + \epsilon)} \quad (4.2)$$

Note that these metrics are not only differentiated by their equation. First, *UMass* requires an ordered word set, since each word is only compared with the preceding ones; while *NPMI* is symmetric and considers all the different combinations of words in the set independently of their order. The probabilities of these metrics are also computed differently. *UMass* uses boolean document counts, which means it estimates the probability of a single word as the number of documents in which the word occurs divided by the total number of documents. Analogously, the joint probability of two words is estimated by the number of documents containing both words divided by the total number of documents. On the contrary, *NPMI* uses a sliding window of a fixed size *sw*. This window moves over the documents, at one token per step. In each step, it considers the aforementioned window as an independent document and applies boolean document counts to it. Additionally, for both methods counts can be calculated based on the training corpus or on an external corpus such as Wikipedia.

The smoothing factor  $\epsilon$  has shown to be very influential in these calculations, following [56] we set  $\epsilon = 10^{-12}$ . For the sliding window we used the default  $sw = 10$ , the top number of words was set to  $N = 10$  and as for the reference corpus we tried only internal measures.

Though this numerical metrics are very useful to inspect rapidly the validity of different models, we also exploited the guidance of the classification provided by Papers With Code. According to this classification, each of our instances corresponds to either of the following : Computer Vision (CV), Sequential, Reinforcement Learning (RL), Graphs, Natural Language Processing (NLP) or Audio. Initially, there was also a General category and some multilabeled observations, however we handled those situations in section 4.1.2.

While topic modeling doesn't fall in a categorization problem, it's reasonable to assume that out of the topics found by any of the algorithms applied, some should resonate with the pre-specified labels. In other words, if a model was good in terms of semantic coherence, we expected each topic or each pair of topics to correspond to the different human classified labels. Along with this reasoning, we studied the distributions of the resulting topics across the different labels. For this process, we assigned to each document it's corresponding topic; or in case the algorithm assumed that it could have more than one topic, the one with the highest probability. Therefore we had for each model and each observation two labels: the topic number and the human classified area obtained from Papers With Code.



# Chapter 5

## Results

Ultimately, our goal is to find a topic model with characteristics that allow us to assign a human classified label to one or more topics. Focusing on abstracts, this type of model could be useful for finding papers similar to each other or discovering the different areas addressed in a paper without reading it.

When inspecting the highest values for  $NPMI$  and  $UMASS$  for the different settings in LDA, all hyperparameters differed. On the contrary, for BTM  $\beta = \frac{20}{k}$  resulted in both cases as the best results, while  $\alpha$  varied more. In Figure 5.1 we compared different situations that performed good for the two metrics. In Figure 5.1a the different settings compared are  $V1 = \{\alpha = 10, \beta = \frac{20}{K}\}$ ,  $V2 = \{\alpha = 1, \beta = \frac{10}{K}\}$  and  $V3 = \{\alpha = 1, \beta = \frac{1}{K}\}$ . In Figure 5.1b  $V1 = \{\alpha = 0.1, \beta = \frac{20}{K}\}$ ,  $V2 = \{\alpha = 0.01, \beta = \frac{20}{K}\}$  and  $V3 = \{\alpha = 10, \beta = \frac{20}{K}\}$ .

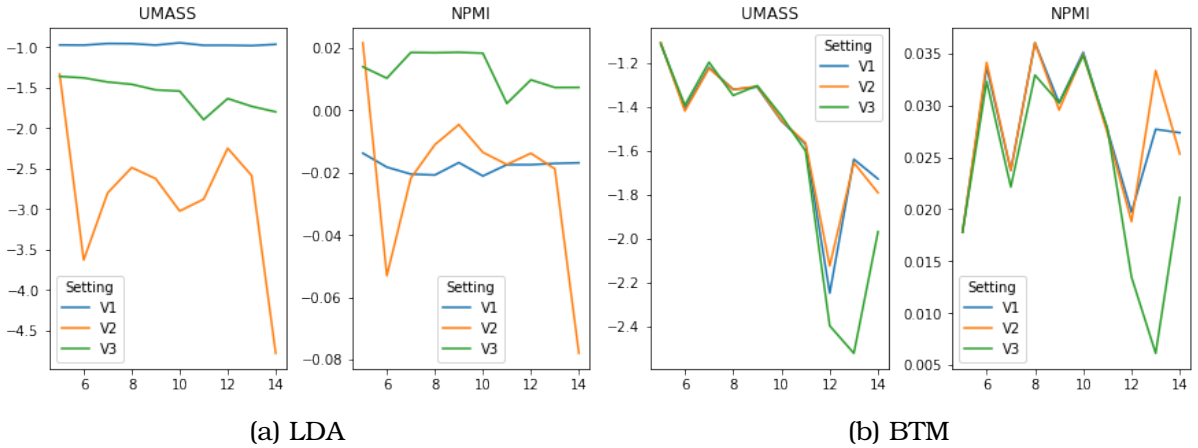


Figure 5.1:  $UMASS$  and  $NPMI$  values of different models on abstracts in terms of the number of topics  $k \in [5, 14]$

For the LDA models we see that there is no clear option that dominates the others, i.e. there's no model that outperforms the rest in both metrics. We opted for considering as the “best” model  $M1 = \{K = 10, \alpha = 1, \beta = \frac{1}{10} = 0.1\}$ , which corresponds to setting V3 with 10 topics because it seems to perform adequately in both graphs and maintain a balance between  $NPMI$  and  $UMASS$  coherence values. As for the BTM model, V1 and V2 have a very similar behaviour for most values of number of topics. In this

case we chose  $M2 = \{K = 8, \alpha = 0.01, \beta = \frac{20}{8} = 2.5\}$ , which corresponds to setting V2 with 8 topics.

Moving on to the models built with BERTopic, as we explained in section 4.3, we only altered the embedding model. In Table 5.1 we study the number of topics, UMASS and NPMI obtained for these variations. In the rest of the study, the maximum values of *UMASS* and *NPMI* will be set to bold in the different result tables. Recall that this algorithm finds automatically the optimal number of topics as opposed to LDA and BTM. Under these circumstances, we found a model that outperforms the others in both NPMI and UMASS. Therefore we can define, analogously to  $M1$  and  $M2$ ,  $M3 = \{K = 12, embedding = all-mpnet-base-v2\}$ .

Embedding model	K	UMASS	NPMI
<i>all-MiniLM-L6-v2</i>	14	-3.1757	-0.0334
<i>all-mpnet-base-v2</i>	12	<b>-2.5905</b>	<b>-0.0093</b>
<i>allenai-specter</i>	17	-2.7263	-0.0568

Table 5.1: BERT values obtained for different embeddings on abstracts

In Table 5.2 we can see an overview of the resulting metrics for the different models chosen for each algorithm. Once we obtained this last 3 models to compare, we proceeded with the study of the correlations of the different topics with the hand-curated labels.

Model	K	UMASS	NPMI
$M1$	10	-1.4895	0.0244
$M2$	8	<b>-1.3203</b>	<b>0.0360</b>
$M3$	12	-2.5905	-0.0093

Table 5.2: Description of best models for each algorithm for abstract instances

For all models we found a different topic that resembled clearly each of these labels: Audio, NLP, Graph and RL. Nevertheless for Computer Vision and Sequential it was not as transparent. On the one hand, our dataset was very imbalanced in terms of these labels as we saw in Table 4.1, where 524 out of 891 instances were classified as Computer Vision. We supposed the lack of representation of one single topic as Computer Vision’s category could be due to the big amount of these observations. This may have lead the algorithms to learn different micro-topics on this area. On the other hand, at first hand we couldn’t find a reasonable explanation for the deficiency to identify Sequential observations. A big difference we found between these cases is that while Computer Vision spanned new “unassigned” topics, Sequential seemed to be a mixture of Audio and NLP in LDA and BERTopic models or NLP and Graphs in BTM.

In order to choose the model we thought had the best qualities we relied on Table 5.2 results and how discriminative the models were. Taking into account the first issue, we discarded  $M3$  for its low values in both *UMASS* and *NPMI*. Regarding the discrimination of the models, we chose  $M1$ , i.e the model built by the LDA algorithm, though the results were surprisingly similar to BTM’s.



## Results

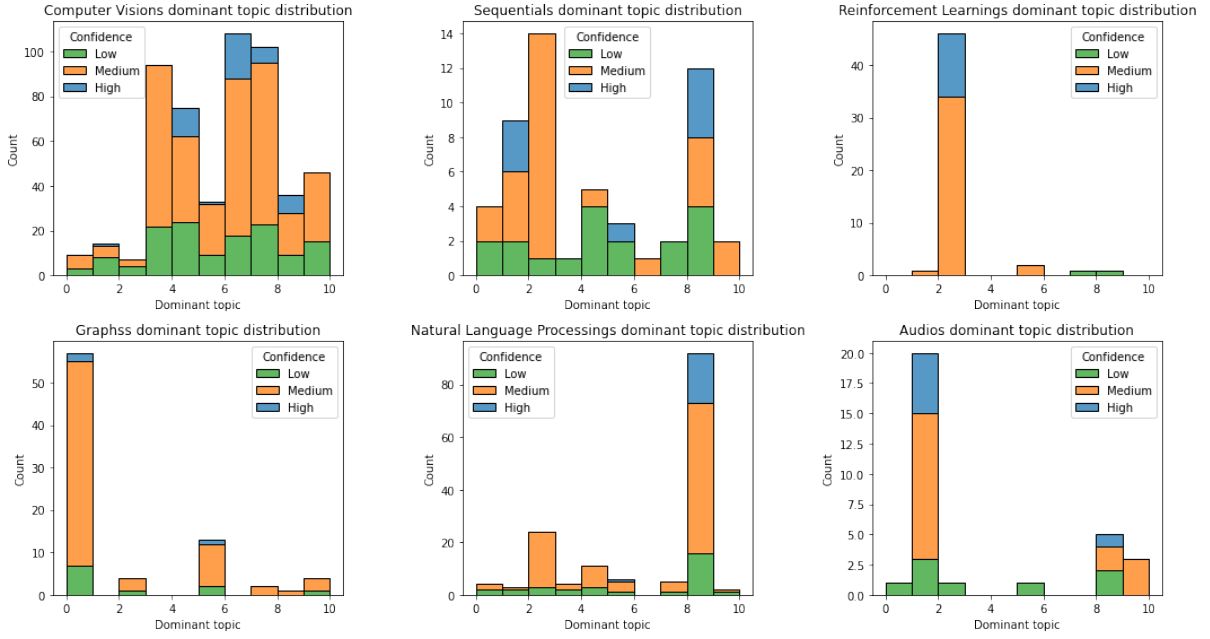


Figure 5.2: Distribution of M1's resulting topics across categories

In Figure 5.2 we can inspect the resulting topics of M1 across the categories. Since LDA calculated the probabilities of each topic to belong to the different abstracts, we selected the one with highest probability. Additionally, to include more information we divided this probabilities in 3 levels. Defining *prob* as this said probability, the three levels were:

- Low if  $prob \in [0, 0.33)$
- Medium if  $prob \in [0.33, 0.66)$
- High if  $prob \in [0.66, 1]$

We can identify clearly topics 0, 1, 2 and 8 with Graphs, Audio, Reinforcement Learning and Natural Language Processing respectively. In most cases this attribution is made with mainly a medium confidence. Focusing on the Sequential label graph, on the top center, we distinguish a mixture between topics 1, 2 and 8. Due to our lack of specialization in this area from their top 10 words (Table 5.3) we are also not able to perceive the goodness of this mixture. However, what we do notice looking at this table is that many words are repeated over different topics such as model or network.

Moving on to the most represented label in our dataset, we can see the resulting topics most related to it on the top left of Figure 5.2. As we said, we believe this excessive number of topics correlated to Computer Vision could be related to different sub-domains on this area. In the Papers With Code web there are more than 50 subcategories within Computer Vision methods. Because of this, we tried to relate the dominant topics 3, 4, 6, 7 with some of this subcategories by inspecting the top 10 words of each topic in Table 5.3. On topic 3 we found words such as 'imag', 'spatial', and '3d' which resonated to us to Image Models. On topic 4 we couldn't assign a specific category since we believed that most words were very general. To our consideration words like 'model', 'accuracy', 'network' or 'paramet' may belong to any Machine Learning technique. The first two words on topic 6 are 'object' and 'detect',

---

Topic 0	Topic 1	Topic 2	Topic 3	Topic 4
graph	video	attent	imag	model
network	model	learn	transform	architectur
structur	generat	self	segment	effici
neural	end	sequenc	semant	comput
inform	speech	transform	local	accuracy
propos	qualiti	perfrom	featur	network
method	audio	algorithm	spatial	paramet
predict	extract	polici	3d	achiev
interact	tempr	use	base	convolut
data	relat	state	resolut	imagenet

Topic 5	Topic 6	Topic 7	Topic 8	Topic 9
learn	object	imag	model	network
represent	detect	train	task	convolut
use	instanc	generat	train	layer
model	propos	data	languag	featur
supervis	segment	model	pre	propos
data	detector	augment	text	multi
network	point	method	transform	perform
space	stage	face	token	dataset
embed	featur	use	encod	block
deep	mask	sampl	modal	two

Table 5.3: Top 10 words of model  $M1$

which lead us directly to Object Detection. Finally, on topic 7 though there are diverse words we could relate 'generat', 'data' and 'augment' with Data Augmentation. After this examination we could name the resulted topics as:

- Topic 0 as Graphs
- Topic 1 as Audio
- Topic 2 as Reinforcement Learning
- Topic 3 as Image Models
- Topic 4 as filler
- Topic 5 as filler
- Topic 6 as Object Detection
- Topic 7 as Data Augmentation
- Topic 8 as Natural Language Processing
- Topic 9 as filler

where fillers refer to topics whose words could be generalized to any category such as 'model', 'represent' or 'dataset'.

## Results

Analogously for README files, we can examine the best values of *UMASS* and *NPMI* for both LDA and BTM in Figure 5.3. Here in Figure 5.3a  $V1 = \{\alpha = 10, \beta = \frac{10}{K}\}$ ,  $V2 = \{\alpha = 0.01, \beta = \frac{10}{K}\}$  and  $V3 = \{\alpha = 10, \beta = \frac{1}{K}\}$ . For BTM we doubted between four different settings  $V1 = \{\alpha = 0.01, \beta = \frac{1}{K}\}$ ,  $V2 = \{\alpha = 0.1, \beta = \frac{1}{K}\}$ ,  $V3 = \{\alpha = 10, \beta = \frac{10}{K}\}$  and  $V4 = \{\alpha = 0.01, \beta = \frac{10}{K}\}$ .

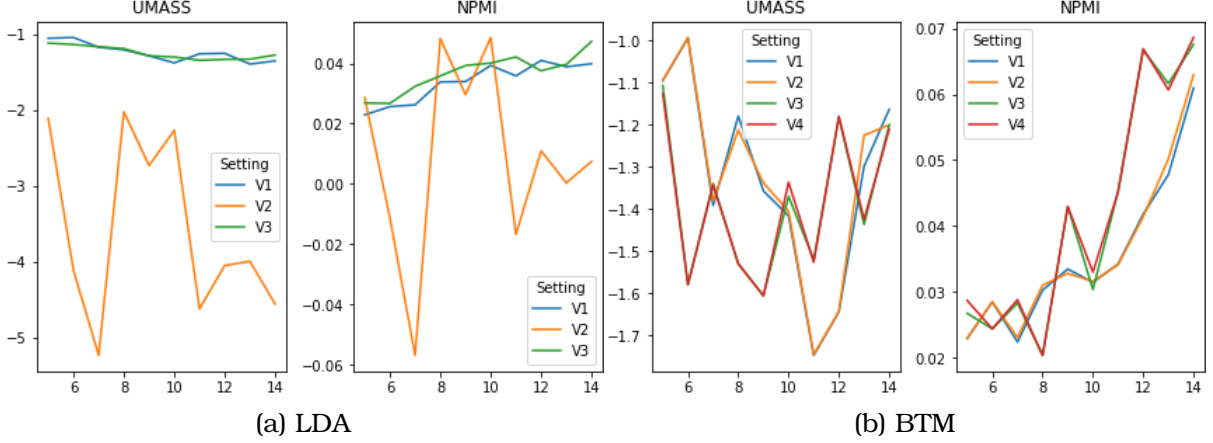


Figure 5.3: *UMASS* and *NPMI* values of different models on README files in terms of the number of topics  $k \in [5, 14]$

For the LDA models, inspecting Figure 5.3a we notice that V1 and V3 have a similar behaviours and for the most part outperform the models adjusted to V2. On this note, we decided to keep V3 settings with  $K = 10$ , i.e  $M4 = \{K = 10, \alpha = 10, \beta = \frac{10}{10} = 1\}$ . Similarly for BTM, V3 and V4 overlap on most values of  $K$  on Figure 5.3b. We notice that generally V1 and V2 perform better with lower values of  $K$  whereas V3 and V4 improve from  $K = 9$ . V4 seemed to achieve the best *NPMI* results with  $K = 12$  while maintaining high values of *UMASS*, therefore we defined model  $M5 = \{K = 12, \alpha = 0.01, \beta = \frac{10}{12} = 0.83\}$ .

Nextly, through the BERTopic algorithm using the three different embedding models and maintaining for the rest of the parameters the same values as for the abstracts, we obtained too many topics in all cases. Specifically 129 topics for `all-MiniLM-L6-v2`, 168 topics for `all-mpnet-base-v2` and 132 for `allenai-specter`. Analyzing these models would be incongruous with the purpose of topic modeling in our task due to the amount of time and effort it would take. Additionally, though computational time was not defined as a metric and wasn't mentioned in previous models we find important to remark that these models took on average 2 hours to build as opposed to the maximum of 10 minutes of any other model. Under these conditions, we decided to tune the minimum cluster size in order to obtain more efficient models on both computation and evaluation terms. Setting this value to 50 and 100, much higher than the previous of 10, we obtained better results.

In Table 5.4 we can see the values obtained for number of topics, *UMASS* and *NPMI*. Firstly, in terms of running time the `all-MiniLM-L6-v2` took a maximum of 2 minutes to train and both `all-mpnet-base-v2` and `allenai-specter` 4 minutes. In this aspect it's clear that there is at least an improvement on computational efficiency. We also managed to reduce significantly the number of topics. In order to choose one of these models we tried to obtain a balance between *UMASS* and *NPMI*.

Embedding model	Minimum cluster size	K	UMASS	NPMI
<i>all-MiniLM-L6-v2</i>	50	21	-1.8107	0.0967
<i>all-mpnet-base-v2</i>	50	30	-1.9257	<b>0.1028</b>
<i>allenai-specter</i>	50	24	-1.9717	0.0877
<i>all-MiniLM-L6-v2</i>	100	8	-1.0857	0.0948
<i>all-mpnet-base-v2</i>	100	6	-1.2950	0.0829
<i>allenai-specter</i>	100	5	<b>-1.0693</b>	0.0868

Table 5.4: BERT values obtained for different embeddings on README files

Additionally, in case of models under similar conditions we preferred those with a lower number of topics for an easier interpretability of the results. On this basis, we selected  $M6 = \{K = 8, embedding = all-miniLM-L6-v2\}$  which does a good job on both metrics.

Following the same process as before, we compared the final models for each algorithm taking into account their topics coherence with the predefined categories. We can see in Table 5.5 a summary of these models results.

Model	K	UMASS	NPMI
<i>M4</i>	10	-1.3514	0.0367
<i>M5</i>	12	-1.1800	0.0669
<i>M6</i>	8	<b>-1.0857</b>	<b>0.0948</b>

Table 5.5: Description of best models for each algorithm for README instances

Though the biterm topic model was designed specifically for short texts, in our experience it resulted as the worst of the three models. This model had really unstable topics across the different categories. On the contrary, LDA and BERTopic performed similarly both between them and in relation to the models studied before. We found the most consistent to be *M6*.

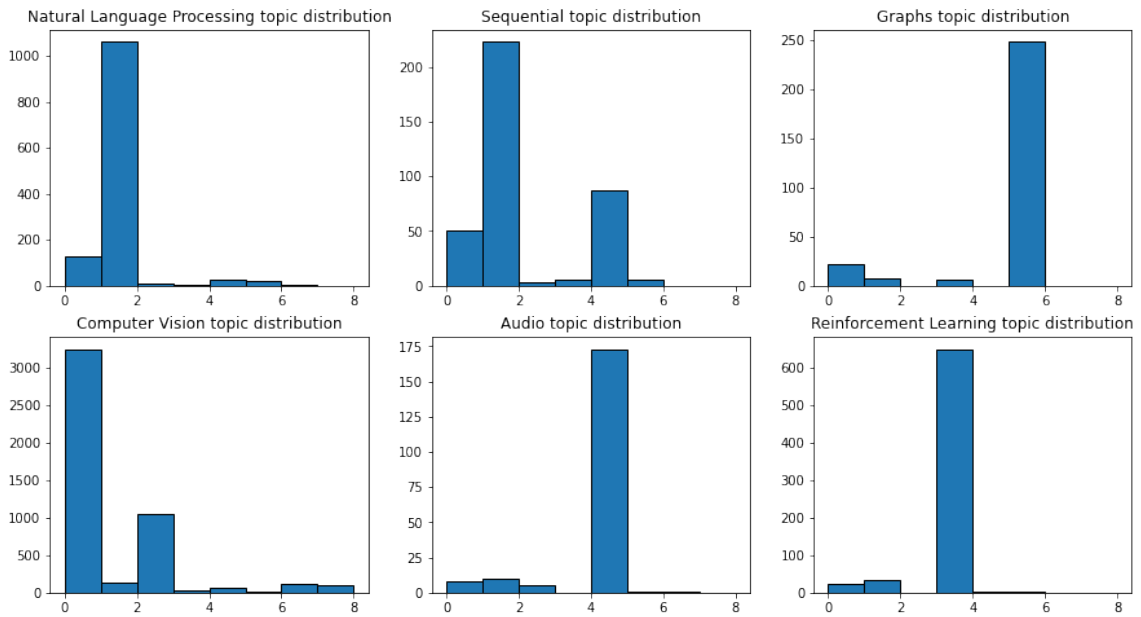


Figure 5.4: Distribution of *M6*'s resulting topics across categories

## Results

In Figure 5.4 we can see the distribution of the topics across the labels. In this case, since this algorithm only assigns one topic to each document there are not confidence levels. Starting at the top we see Natural Language Processing represented by topic 1, Sequential by 1 and 4, and Graphs by 5. Below, on Computer Vision topics 0 and 2 stand out, topic 4 on Audio and topic 3 on Reinforcement Learning. Comparing with the model obtained for abstracts again we see that Sequential seems to be a mixture of NLP and Audio which will be studied more in depth with the top words. On the other hand, the embedding model seems to have captured well Computer Vision even with the present disparity. However, we do notice that topics 6 and 7 are significantly smaller than the rest, and seem poorly represented by any of the labels.

Topic 0	Topic 1	Topic 2	Topic 3
model	model	images	agent
image	bert	gan	learning
training	text	image	environment
data	models	generator	reinforcement
dataset	data	training	policy
images	language	stylegan	agents
learning	training	adversarial	action
models	word	discriminator	reward
using	pre	generative	actor
use	use	model	deep

Topic 4	Topic 5	Topic 6	Topic 7
audio	graph	vae	ssd
speech	node	variational	voc
model	networks	latent	http
wavenet	gcn	autoencoder	model
speaker	graphs	mnist	boxes
training	learning	training	multibox
tts	embedding	model	vgg
dataset	model	png	shot
use	paper	auto	models
spectrogram	neural	space	dataset

Table 5.6: Top 10 words of model  $M_6$

Inspecting the top words of each topic in Table 5.6 we notice consistency with what we perceived with the distributions in Figure 5.4. Before we related NLP with topic 1, which includes words such as 'bert', 'text' and 'language'. Next, topic 3 with 'reinforcement' and 'learning' explicitly in its top words was attributed to RL. Topic 4 was assigned to Audio and its top 2 words are 'audio' and 'speech'. Following with topic 5, again its top word was 'graph' accordingly to its assigned label 'Graphs'. As for topics 0, 2, 6 and 7 we found them more difficult to associate, since we wanted to differentiate possible subcategories inside Computer Vision. On topic 0 we believed it could be a filler due to the presence of words like 'model', 'data', 'datasets' or 'use'. Then, topic 2 seems correctly about Computer Vision and leaning specifically towards General Adversarial Networks (GANs). On the other hand, topic 6 seems to represent

---

another kind of deep generative models, Variational Autoencoders (VAEs). On both of these topics there are also words related to image models such as 'image', 'png' or 'mnist' which is an image database [57]. For this reason we believed they are correctly assigned to Computer Vision since the topics are specifically correlated with this network architectures applied to images as input data. Finally, topic 7 is notably related to Object Detection. With words like 'ssd', 'multibox' and 'shot' alluding to Single Shot Multibox Detector (SSD), a method for detecting objects in a single forward pass of the network [58]. Thus, we can name the topics as follows:

- Topic 0 as filler
- Topic 1 as Natural Language Processing
- Topic 2 as General Adversarial Networks on images
- Topic 3 as Reinforcement Learning
- Topic 4 as Audio
- Topic 5 as Graphs
- Topic 6 as Variational Autoencoders on images
- Topic 7 as Object Detection

## Chapter 6

# Conclusion and Future Work

Having knowledge of the latest techniques and algorithms developed is crucial for researchers. However, the amount of papers published nowadays make it intolerably difficult for them to be completely up to date. Our goal was to build a model that could process tons of papers abstracts in little amount of time for researchers to find papers related to their tasks. Additionally, if not only they knew about recent findings but also found along with it software already developed they would also save up time on that step. For this reason we proceeded to build different models not only on papers abstracts but also on their software README files.

In order to answer our first research question, we started by building topic models on different paper's abstracts. When inspecting these models, we found the best results to be from the LDA algorithm. This model performed adequately to identify topics on Graphs, Audio, Reinforcement Learning and Natural Language Processing. Due to the nature of our dataset, it even encountered subcategories within Computer Vision like Image Models, Object Detection and Data Augmentation. Unfortunately the Sequential area, which we know was present in the dataset due to a predefined categorization was not identified by the model and studying the distribution of topics in this category it seemed like a mixture of NLP and Audio. The algorithm also had trouble unifying in one topic words that could be generalized to almost any area.

Focusing on the software associated with the papers we also tried to discover the latent mixture of topics, as was stated in our second research question. In this matter, the model built with BERTopic turned out to be the most consistent, followed closely by LDA. This model found eight to be the optimal number of topics. Four of these topics represented clearly NLP, Graphs, Audio and RL; identically to the model for the first dataset. It also discovered effectively different subtopics of Computer Vision: GANs, VAEs and Object Detection. In contrast with the abstracts model, it accomplished to integrate all 'filler' words in one topic. However, it did not identify an independent topic for Sequential as it appeared to be, once again, a mixture of NLP and Audio.

Generally, and excluding Sequential, we consider the models built reasonably discriminative. We saw that LDA worked well for different types of data, accordingly to the fact that is probably the most known and used algorithm for topic modeling. On the contrary, although BTM is designed specifically for short texts it was not able to find coherent topics, considering either *UMASS*, *NPMI* or consistency across labels

---

in this study. We found the hyperparameter tuning step to be crucial for both of these algorithms. The optimal values varied both depending on the algorithm and the dataset, with  $\alpha = 0.01, 0.1, 10$  and  $\beta = \frac{1}{K}, \frac{10}{K}, \frac{20}{K}$  present on different choices of models. This evidence supports the incertitude of results coming from models that use 'off-the-shelf' hyperparameter values, which can lead to misleading conclusions due to LDA's and BTM's instability [59].

BERTopic algorithm was developed very recently, in 2021, and has proven to function fairly good in the present work. Taking it's recency into account we found it to provide more than satisfactory results and even outperforming the popular LDA in some situations. In this case, the embedding model was probably the most decisive parameter to tune. We found both `all-mpnet-base-v2` and `all-MiniLM-L6-v2`, which have been trained for general purpose models, to provide the best quality.

In this paper, we built different models which are able to derive topics and cluster research papers based on its abstracts and part of its software documentation. Our algorithms have a variety of applications. In future work, it would be useful to implement them in systems for topic search, recommendations and software comparison. Additionally before these implementations, we would propose to corroborate the results obtained with different papers and try to improve the model in terms of the Sequential area.

One weakness worth mentioning is the nature of the method used for optimizing the hyperparameters of both LDA and BTM. The simple fact of tuning these instead of accepting the usual values used in literature is relevant. However, we recognize that the best values obtained the first time could possibly be local optimals and assuming close values can give the best results possible is questionable. In future work, genetic algorithms or other optimizing tools like LDADE [59] could be explored instead of overlooking this fact.

We also feel important to emphasize that the relationships done in section 5 were done with a superficial knowledge on all areas. In future work it would be interesting to have specialized researchers in the different areas match the topics found, specially before implementing the algorithms in application systems.



# Bibliography

- [1] Jo Erskine Hannay, Carolyn MacLeod, Janice Singer, Hans Petter Langtangen, Dietmar Pfahl, and Greg Wilson. How do scientists develop and use scientific software? In *2009 ICSE Workshop on Software Engineering for Computational Science and Engineering*, pages 1–8, 2009.
- [2] Steven J Vaughan-Nichols. It’s an open-source world: 78 percent of companies run open-source software. *zdnet.com*, 2015.
- [3] Lucas Joppa, Greg Mcinerny, Richard Harper, Lara Salido, Kenji Takeda, Kenton O’Hara, David Gavaghan, and Stephen Emmott. Troubling trends in scientific software use. *Science (New York, N.Y.)*, 340:814–815, 05 2013.
- [4] Papers with code. <https://paperswithcode.com/>.
- [5] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.
- [6] Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine learning*, 42(1):177–196, 2001.
- [7] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.
- [8] Mohammad Abdul Hadi and Fatemeh H. Fard. AOBTM: adaptive online biterm topic modeling for version sensitive short-texts analysis. *CoRR*, abs/2009.09930, 2020.
- [9] Thorben Schomacker and Marina Tropmann-Frick. Language representation models: An overview. *Entropy*, 23(11), 2021.
- [10] Suzanna Sia, Ayush Dalmia, and Sabrina J Mielke. Tired of topic models? clusters of pretrained word embeddings make for fast and good topics too! *arXiv preprint arXiv:2004.14914*, 2020.
- [11] Dimo Angelov. Top2vec: Distributed representations of topics. *arXiv preprint arXiv:2008.09470*, 2020.
- [12] Maarten Grootendorst. Bertopic: Neural topic modeling with a class-based tf-idf procedure. *arXiv preprint arXiv:2203.05794*, 2022.
- [13] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer, 2003.

- 
- [14] Maarten Grootendorst. Bertopic: Leveraging bert and c-tf-idf to create easily interpretable topics., 2020.
  - [15] Tianyi Lin, Wentao Tian, Qiaozhu Mei, and Hong Cheng. The dual-sparse topic model: mining focused topics and focused terms in short text. In *Proceedings of the 23rd international conference on World wide web*, pages 539–550, 2014.
  - [16] Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. Twiterrank: finding topic-sensitive influential twitterers. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 261–270, 2010.
  - [17] Amit Gruber, Yair Weiss, and Michal Rosen-Zvi. Hidden topic markov models. In *Artificial intelligence and statistics*, pages 163–170. PMLR, 2007.
  - [18] Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. Comparing twitter and traditional media using topic models. In *European conference on information retrieval*, pages 338–349. Springer, 2011.
  - [19] Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. A biterm topic model for short texts. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1445–1456, 2013.
  - [20] Harnessing the power of content. Technical report, Elsevier, 2014.
  - [21] David Westergaard, Hans-Henrik Stærfeldt, Christian Tønsberg, Lars Juhl Jensen, and Søren Brunak. Text mining of 15 million full-text scientific articles. *Biorxiv*, page 162099, 2017.
  - [22] Subasish Das, Xiaoduan Sun, and Anandi Dutta. Text mining and topic modeling of compendiums of papers from transportation research board annual meetings. *Transportation Research Record*, 2552(1):48–56, 2016.
  - [23] Kumar Shubankar, AdityaPratap Singh, and Vikram Pudi. A frequent keyword-set based algorithm for topic modeling and clustering of research papers. In *2011 3rd Conference on Data Mining and Optimization (DMO)*, pages 96–102. IEEE, 2011.
  - [24] Camila Costa Silva, Matthias Galster, and Fabian Gilson. Topic modeling in software engineering research. *Empirical Software Engineering*, 26(6):1–62, 2021.
  - [25] Tamer Mohamed Abdellatif, Luiz Fernando Capretz, and Danny Ho. Automatic recall of software lessons learned for software project managers. *Information and Software Technology*, 115:44–57, 2019.
  - [26] Luciana L Silva, Marco Tulio Valente, and Marcelo A Maia. Co-change patterns: A large scale empirical study. *Journal of Systems and Software*, 152:196–214, 2019.
  - [27] Nengwen Zhao, Junjie Chen, Zhou Wang, Xiao Peng, Gang Wang, Yong Wu, Fang Zhou, Zhen Feng, Xiaohui Nie, Wenchi Zhang, et al. Real-time incident prediction for online service systems. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 315–326, 2020.

- [28] Cuiyun Gao, Jichuan Zeng, Michael R Lyu, and Irwin King. Online app review analysis for identifying emerging issues. In *Proceedings of the 40th International Conference on Software Engineering*, pages 48–58, 2018.
- [29] Huan Yu, Xin Xia, Xiaoqiong Zhao, and Weiwei Qiu. Combining collaborative filtering and topic modeling for more accurate android mobile app library recommendation. In *Proceedings of the 9th Asia-Pacific Symposium on Internetware*, pages 1–6, 2017.
- [30] Gede Artha Azriadi Prana, Christoph Treude, Ferdian Thung, Thushari Atapattu, and David Lo. Categorizing the content of github readme files. *Empirical Software Engineering*, 24(3):1296–1327, 2019.
- [31] Matthew Hoffman, Francis Bach, and David Blei. Online learning for latent dirichlet allocation. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.
- [32] Myky Tran and Michael Truong. Clustering short text messages using unsupervised machine learning. *LU-CS-EX 2019-20*, 2019.
- [33] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.
- [34] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2018.
- [35] 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*, pages 160–172. Springer, 2013.
- [36] Guido Van Rossum. *The Python Library Reference, release 3.10.5*. Python Software Foundation, 2020.
- [37] Edward Loper Bird, Steven and Ewan Klein. *Natural Language Processing with Python*. O’Reilly Media Inc., 2009.
- [38] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.
- [39] Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020.
- [40] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90, 2007.
- [41] Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021.

- 
- [42] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. <http://is.muni.cz/publication/884893/en>.
  - [43] Amritanshu Agrawal, Wei Fu, and Tim Menzies. What is wrong with topic modeling? and how to fix it using search-based software engineering. *Information and Software Technology*, 98:74–88, 2018.
  - [44] Anton Barua, Stephen W Thomas, and Ahmed E Hassan. What are developers talking about? an analysis of topics and trends in stack overflow. *Empirical Software Engineering*, 19(3):619–654, 2014.
  - [45] Ying Fu, Meng Yan, Xiaohong Zhang, Ling Xu, Dan Yang, and Jeffrey D Kymer. Automated classification of software change messages by semi-supervised latent dirichlet allocation. *Information and Software Technology*, 57:369–377, 2015.
  - [46] Tse-Hsun Chen, Stephen W Thomas, Meiyappan Nagappan, and Ahmed E Hassan. Explaining software defects using topic models. In *2012 9th IEEE working conference on mining software repositories (MSR)*, pages 189–198. IEEE, 2012.
  - [47] Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S. Weld. Specter: Document-level representation learning using citation-informed transformers, 2020.
  - [48] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.
  - [49] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality, 2013.
  - [50] Jonathan Chang, Sean Gerrish, Chong Wang, Jordan Boyd-Graber, and David Blei. Reading tea leaves: How humans interpret topic models. *Advances in neural information processing systems*, 22, 2009.
  - [51] David Mimno, Hanna Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. Optimizing semantic coherence in topic models. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 262–272, 2011.
  - [52] Yingcheng Sun, Kenneth Loparo, and Richard Kolacinski. Conversational structure aware and context sensitive topic model for online discussions. In *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*, pages 85–92. IEEE, 2020.
  - [53] Oscar Liessens and Eugen Pircalabelu. Unsupervised topic modeling for short documents. Master’s thesis, Louvain School of Statistics, 2021.
  - [54] 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*, pages 399–408, 2015.
  - [55] Jey Han Lau, David Newman, and Timothy Baldwin. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings*

- of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 530–539, 2014.
- [56] Keith Stevens, Philip Kegelmeyer, David Andrzejewski, and David Buttler. Exploring topic coherence over many models and many topics. 07 2012.
- [57] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.
- [58] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot MultiBox detector. In *Computer Vision – ECCV 2016*, pages 21–37. Springer International Publishing, 2016.
- [59] Amritanshu Agrawal, Wei Fu, and Tim Menzies. What is wrong with topic modeling? and how to fix it using search-based software engineering. *Information and Software Technology*, 98:74–88, 2018.