UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL DE CALCULATOARE

PROIECT DE DIPLOMĂ

Analiză de competențe științifice
Sesiunea vară 2023

Tudor-Mihai Pescaru

Coordonator(i) științific:

Prof. Habil. Ciprian Dobre, Ph. D.
S. L. Gabriel Gutu-Robu, Ph. D.

BUCUREȘTI
2023

UNIVERSITY POLITEHNICA OF BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS
COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

DIPLOMA PROJECT

Analysis of scientific competencies
Summer 2023

Tudor-Mihai Pescaru

Thesis advisor(s):

Ciprian Dobre, Professor, PhD., Habil.
Gabriel Guțu-Robu, Assist. Prof., PhD.

BUCHAREST
2023

CONTENTS

SINOPSIS

Acest proiect propune și prezintă două soluții pentru modelarea corpusului de articole academice publicate de către un anumit autor ca o colecție de subiecte sau cuvinte cheie. Această modelare este ulterior utilizată pentru a determina dacă un nou articol academic, publicat sub numele unui autor, abordează subiecte similare cu cele pe care autorul le-a acoperit anterior în publicații. Astfel, publicațiile care prezintă un grad ridicat de diferență față de alte lucrări ale acelorași autori pot fi semnalate ca excepții. Metodele propuse utilizează tehnici de învățare automată des utilizate în domeniul prelucrării limbajului natural, cum ar fi algoritmi supervizați și nesupervizați și modele pre-antrenate de tip transformer. De asemenea, analizăm și vizualizăm performanța acestui sistem pe un set de date de dimensiuni reduse și cum poate fi utilizat acest sistem pentru a determina dacă un nou articol academic ar trebui clasificat sau nu ca o excepție.

ABSTRACT

This project proposes and presents two solutions for modeling the corpus of academic papers published by a certain author as a collection of topics or keywords. This modeling is further used to determine if a new academic paper, published under the name of an author, tackles similar subjects to those that author has previously covered in publications. As such, publications that show a high degree of dissimilarity from other works of the same authors can be flagged as outliers. The methods proposed use machine learning techniques widely used in the field of natural language processing such as supervised and unsupervised algorithms and pre-trained transformer-based models. We also analyze and visualize the performance of this system on a small-sized dataset and how it can be used to determine whether or not a new academic paper should be classified as an outlier.

1 INTRODUCTION

This project proposes two methods that can be used to analyze the similarity between an academic paper and a corpus of multiple other papers by breaking these texts down into topics and keywords. Different formulas are used to compare the generated topics or keywords and the resulting similarity values are run through a classifier to predict whether the paper matches or not the themes of the corpus. This project aims to speed up the manual process of confirming an author's ownership over an academic publication by offering an automatic system that will make the preliminary decision.

1.1 Context and Motivation

The primary motivation behind this project stems from the need of the users of the CRESCDI platform[1] for better feedback that will help them approve or reject new publications imported into the platform as belonging to them. The CRESCDI platform is managed by the University Polithenica of Bucharest and serves as an aggregator of academic papers published by authors affiliated with the university. It also acts as a hub where each author can showcase their academic work through a profile that links them to the imported publications authored by them and that also contains a list of their interests in the form of a list of keywords.

Whenever a new academic paper is imported into the platform, the authors listed on it are automatically matched to their user accounts and each user that has already registered for an account will get a notification to approve the paper. Currently, there is limited information that the author can use to determine whether the paper belongs to them without having to go through it manually. Accidentally accepting a new paper as their own can occur due to something as trivial as a name clash in the author list and can have ramifications concerning intellectual property.

1.2 Task at Hand

This project aims to provide a solution to the aforementioned situation by researching ways to implement an automated system that can make a preliminary prediction on the ownership of a newly imported paper for an author. This prediction will serve as feedback for the user and will act as an indication of a potential miss match between the paper and the author. In practice, this prediction will not be final and user confirmation will still be required to approve

[1] https://crescdi.pub.ro

a paper and assign ownership over it.

1.3 Objectives

The main objective of this project is to compare the subject matters discussed in the new paper to those found in the preexisting corpus of publications the author has approved on the platform. This comparison will provide quantifiable data that will be used to make a binary classification of the paper as an outlier with a high degree of accuracy. As a byproduct, the methods utilized provide a modeling of the main topics and interests of the author that can be later used to improve the corresponding system on the CRESCDI platform.

1.4 Proposed solutions

To identify whether a newly uploaded paper matches the topics covered by the automatically associated authors on the platform two methods have been proposed that attempt to provide an automatic analysis system. The first method involves breaking down the corpus of papers for an author into multiple topics, each topic described by a set of keywords related to one another. The new paper is modeled as a set of topics and the topics from the two sets are compared to determine the similarity of the paper to the corpus. A classifier is also trained and used to determine the levels of similarity a new paper needs, to be considered a part of the corpus. The second method involves extracting the most relevant keywords as a whole from both the corpus and the paper and comparing these keywords to determine similarity in such a way that the context in which they appear is also relevant. Another classifier is further trained and used to determine the thresholds of similarity for this method.

1.5 Results

Due to privacy concerns, a limited dataset has been provided for this task. With the size of the dataset being reduced and with having limited processing power, the number of authors and publications per author used have been lowered resulting in smaller-than-ideal training datasets for the classifiers. Despite this, the classifiers for both methods achieved similar performance with F1 scores of 85% and 89% respectively with the second method based on keyword extraction showing better results. This performance improvement can be credited to the more advanced architecture of transformer-based models, pre-trained to solve similar NLP (Natural Language Processing) tasks as opposed to the other approach that uses LDA (Latent Dirichlet Allocation) which is a generative statistical model.

1.6 Document Structure

In the following sections, we go in-depth into the theoretical machine learning background and prerequisites that this project makes use of and discuss the implementation and evaluation of both approaches presented in Section 1.4. In Chapter 2 we present all of the theoretical concepts that serve as the background of this project and discuss their relevance and usage within the project. In Chapter 3 we showcase the current State of the Art and most commonly used tools and techniques for solving problems related to assessing semantic similarity between two texts. In Chapters 4 and 5 we go into detail about the proposed solutions for this problem, how we decided on them, and how we applied some of the tools and techniques presented in Chapters 2 and 3, with Chapter 5 focusing on the actual code implementation of the project. Further, in Chapter 6 we present the process and metrics used to evaluate the performance of the system as well as visualize these metrics and interpret our findings. Finally, in Chapter 7 we summarize all that we have presented throughout this project, to offer an overview of the results obtained, and to discuss ways for this project to be expanded on.

2 BACKGROUND

This chapter aims to present some of the key concepts that form the basis of our project and some of the concepts present within the current state of the art for tasks such as our own and for similar tasks. We will also be presenting the data we will be working with and discussing which components of this data are crucial for our project.

2.1 Dataset

For this task, the main dataset we used was an extract of the data from the CRESCDI platform. It consisted of 10 thousand publications randomly selected out of the total number of about 195 million publications available on the platform and 486 publications from 3 specific users, these being all of the publications these 3 users have added on the platform. Each publication entry consists of various fields, out of which we only kept the most relevant ones for this task. These include the id of the publication, the title, the abstract text, a list of author-defined keywords, authors, and abstract language. All of the publications missing one or more of these fields have been dropped as they would not offer useful information for our implementation.

In addition to this, the dataset also contained a list of 48610 user entries, representing all the users who have authored the publications in the dataset and have created an account on the platform. Each entry contained multiple fields, with the most important being each user's first and last name as well as id. To link each user to their publications, the dataset offered a list of 133549 entries, mapping user ids to each id of the publications they authored or co-authored.

One of the challenges presented by this dataset was the formatting of the data in the CSV (Comma-Separated Values) format, which places all fields of an entry on the same row, separating each field using a comma (,) and having a header-style row at the start of the file to explain what each field represents. As this dataset came from a database dump, where there are few guidelines as to how the data should be structured, the resulting CSV files contained several irregularities that prevented a simple import into the project. As such, an initial cleanup of the data was required. The main challenge stemmed from the inconsistent usage of separator characters and string-delimiting characters. The implementation for the solution that fixed the formatting of the dataset will be presented in more detail in the Formatting the Dataset Section.

With the data now loaded, the first step we took was figuring out the languages we wanted to make use of for our research. Out of a list of multiple languages, consisting of English,

Romanian, Indonesian, French, Italian, and German, we settled upon only using the English and Romanian languages for this project. In further parts of the project, we resorted to only using publications in the English language since the number of publications in Romanian was very limited for training, testing, and evaluation. As such, the publications in Romanian were only used in the exploratory analysis stage of this project. This left us with 4705 distinct English publications from a multitude of authors to be used in the actual core of the project.

2.2 Topic Modeling

Topic modeling is a statistical modeling technique used to identify and obtain the fundamental concepts or subjects from a corpus. This seeks to reveal the latent semantic structure of the documents within the corpus.

Topic modeling aims to automatically detect and extract topics, which are collections of related words or phrases which frequently appear together in the documents. The primary subject matters covered in the text collection are represented by these topics. This technique can be used for a multitude of different NLP tasks including text classification, document clustering, retrieval of information, content recommendation, or sentiment analysis.

Topic modeling can provide important insights into the primary themes and trends existing in the data by revealing hidden topics in a group of documents. In the context of our solution, once a corpus is modeled into a set of topics that describe what the corpus refers to, it can be compared with another corpus modeled in the same way.

2.3 Keyword Extraction

Keyword extraction is the process of identifying and extracting the words or phrases that are most relevant or representative from a corpus. The goal of keyword extraction is to determine the keywords that best describe the topics of that corpus, i.e. the keywords that can summarise the contents of the corpus.

A keyword extraction algorithm typically selects words and phrases and ranks them to determine the most relevant ones. The ranking is generally based on metrics like term frequency, semantic meaning, syntactic structure, etc. Once the ranking is complete the top keywords up until a certain count or certain threshold of relevance are selected.

Since the extracted keywords are the most relevant ones for the corpus and describe the contents of it most accurately, this approach made sense for our solution as the sets of keywords could be compared to determine similarity.

2.4 BERT and Transformers

BERT (Bidirectional Encoder Representations from Transformers) is a machine learning model for natural language processing presented in "Bert: Pre-training of deep bidirectional transformers for language understanding" by Devlin et. al [10]. It makes use of the Transformer architecture presented in "Attention is all you need" by Vaswani et. al [18] that uses an attention mechanism to learn contextual relations between tokens, or words, in a text. The attention mechanism allows the model to analyze and process words in a sequence by taking into account the relations and dependencies between each word and the others in the sequence, as opposed to analyzing the words individually and with no regard for their context.

The Transformer is composed of two main elements, an encoder, and a decoder. The encoder processes the input text and the decoder creates a prediction for the output based on the task on which it is applied. The encoder processes the entire input at once thus allowing the model to learn the context behind each word based on all the other words in the input. As BERT is mainly designed to create a language model, it only makes use of the encoder component of the Transformer architecture.

BERT goes through two pre-training tasks, Masked Language Model (MLM) and Next Sentence Prediction (NSP). For MLM, BERT is given sequences of consecutive words taken from a context, in which a small percentage of the words are masked. The model then attempts to predict the masked words using context from the unmasked words around them. For NSP, BERT is given pairs of sentences, with half the pairs containing two consecutive sentences from the original text and the other half containing two random non-correlating sentences. Using the task, the model learns to predict if the second sentence in each pair is subsequent to the first one or not. Further, BERT can be fine-tuned to solve various NLP tasks.

2.5 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised algorithm used in machine learning, most commonly in classification and regression tasks [5] [9]. In terms of classification, it offers good results for binary classification tasks. Some such examples centered around natural language processing are classifying a set of emails as spam or non-spam or performing sentiment analysis on them by classifying them as expressing positive or negative sentiments.

The concept used by SVM to classify data points is that of hyperplane separation. It attempts to determine the optimal hyperplane that will most accurately separate the different classes the data points belong to. In the case of binary classification, the hyperplane is used to establish the decision boundary between the two classes. During training, SVM attempts to find the decision boundary that represents the highest distance from the nearest data points from both classes to the separation hyperplane. In the case that the data cannot be separated using a linear hyperplane, SVM makes use of its kernel function to transform the data into

a feature space with higher dimensions. Some of the most common kernels used with SVM are the linear kernel, the polynomial kernel, and the RBF kernel [5] [9]. The RBF kernel is shown to have increased performance over the other two due to it being able to resolve more complex situations.

Since SVMs have generally good performance in binary classification tasks and due to them being able to generalize well it made sense to make use of it in the classification part of this project. In our case the classification is binary, a paper belongs to the author based on the comparison to that author's corpus or it does not. We also made use of the RBF kernel to ensure we had the best performance when classifying more complex data such as the data obtained from comparing topics modeled by LDA.

2.6 Evaluation Metrics

When dealing with binary classification tasks the results can be counted under 4 categories:

- True Positives (TP) - data points that were classified correctly as positive examples
- False Positives (FP) - data points that were classified incorrectly as positive examples
- True Negatives (TN) - data points that were classified correctly as negative examples
- False Negatives (FN) - data points that were classified incorrectly as negative examples

To determine the performance of our classifier we can use multiple metrics that make use of the categories presented above.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4}$$

While accuracy can give us a good indication of the performance of our classifier, the F1 score is preferred since it offers a more accurate metric for the cases in which the class distribution in the dataset is imbalanced. Accuracy serves as a good metric in the cases in which the class distributions are balanced, but this is not always true in real-life scenarios.

Additionally, to evaluate the performance of LDA models, we used a metric specific to topic modeling called the Coherence score, more precisely the C_V coherence measure [16]. This metric evaluates the interpretability and coherence of the topics extracted from a document. A higher coherence score indicates that the topics extracted are more meaningful and offer a more accurate representation of the contents of the document from where they were extracted.

This score will be calculated using a coherence module offered by the same framework used for the LDA model.

3 STATE OF THE ART

In this chapter, we will be taking a look at the current state-of-the-art technologies that are most commonly used in approaches similar to our own. Additionally, we will be discussing the broader usage of these technologies and the advancements they present. Some of the technologies presented here have been adapted and used in the implementation of this project as they cater to the objectives of the project and have shown great performance in solving tasks related to topic modeling and keyword extraction. The other technologies are presented as potential alternatives that are used widely within the industry and we will briefly showcase the differences between these and the ones we made use of for our project.

3.1 Topic Modeling

In this section, we will present some of the state of the art technologies for topic modeling and present which one we chose to integrate into our project and what other alternatives there are.

3.1.1 Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) is a statistical model and one of the most widely used technologies for topic modeling. It was introduced in "Latent Dirichlet Allocation" by Blei et al. [4]. It operates in an unsupervised manner to extract hidden (latent) topics from documents which are collections of related words that describe a certain subject matter. This is the technology we integrated into our project for the topic modeling part of the solution since it achieves good performance with most corpora [14].

LDA works under the assumption that both the distribution of topics in a document and the distribution of words in a topic are Dirichlet distributions. It attempts to find the best allocation of words to a topic and of topics to a document. It begins this process by taking a preset number of topics we want to generate for a given document and randomly assigning all words to each of the topics defined by the given number. It then proceeds to calculate the probability of each word belonging to the topic based on the initial assignment. The topic and word assignments are then adjusted based on this probability. These two steps are repeated until convergence, i.e. until the calculated probabilities no longer generate any modifications to the assignments. In the end, LDA outputs the topic distribution which represents the proportions each topic has in the given document, and the word distribution which represents the proportions each word has in its respective topic. In more recent works, such as "Online

Learning for LDA" by Hoffman et al. [13], various improvements over the original algorithm have been proposed. The LDA model offered by the Gensim framework[1] is designed to use the proposed improvements from the aforementioned paper.

One of the main disadvantages of LDA is the fact that pre-trained models are not readily available and each LDA model needs to be trained specifically on the corpora that needs to be analyzed. Due to LDA treating documents as a bag of words, essentially just a collection of words without any regard for syntactic order, words in the document lose their contextual meaning. This leads to very common words as well as inflections of base words losing their relevance in the context of LDA topics. As such, the text needs to be pre-processed before being analyzed by LDA. This pre-processing generally includes removing special characters, removing capitalization, eliminating stop words, and performing stemming or lemmatization to keep only the base variants of the words.

3.1.2 BERTopic

BERTopic [11] represents an alternative as well as an improvement on LDA. It uses a pre-trained BERT model to perform topic modeling. As BERT is a transformer-based model, it showcases powerful language processing and understanding capabilities. In addition to this, it uses a clustering algorithm to generate the topics.

The first step in the process is passing the documents analyzed through BERT to obtain the document embeddings, which are numerical representations of those documents that retain the semantic meaning [11]. The data then goes through a dimensionality reduction process using Uniform Manifold Approximation and Projection (UMAP) to be prepared for clustering [11]. The clustering is done using HDBSCAN which is a clustering algorithm that takes into account density. This is used to determine the topics and any potential outliers [11]. Similarly to LDA, BERTopic uses a bag-of-words representation of all words in the documents to ensure that no predefined cluster structure is taken into account. Alongside the bag-of-words representation, a c-TF-IDF (class-based TF-IDF) model is used. This is a variation of the classical TF-IDF model and it represents each cluster as a class. This allows for the comparisons of the importance of words to be performed between classes instead of between documents. This helps extract the most important words for each cluster and as such, the description of each topic modeled. [11]

BERTopic can solve some of the disadvantages of LDA. One disadvantage solved is the need to train the model specifically for the task. There are several BERT-based models available that are pre-trained for general purposes and are capable of solving a multitude of different tasks. The other disadvantage solved is the need for pre-processing of text. This is done by making use of document embeddings that hold semantic meaning and are best computed when the text is unaltered.

[1] https://radimrehurek.com/gensim/models/ldamodel.html

3.1.3 Large Language Models (LLMs)

Large language models represent the latest advancements in the field of natural language processing. Models like GPT-3 [6] or GPT-4 [15] use transformers at their core, similar to BERT, but on a much larger scale. These have billions of parameters and are trained on a very wide range of textual data from the Internet as well as from other sources. These have showcased an even more advanced language understanding than previous transformer models.

These can be used similarly to the way BERTopic uses BERT. The BERTopic framework currently offers support for using an LLM to fine-tune the topic representation. As these can still be considered new technology, their full potential and ways of use have still not been entirely explored.

3.2 Keyword Extraction

In this section, we will present some of the technologies considered state of the art for topic modeling and present which one we chose to integrate into our project and what other alternatives there are.

3.2.1 RAKE (Rapid Automatic Keyword Extraction)

Rapid Automatic Keyword Extraction (RAKE) [17] is an algorithm that can quickly extract the most relevant words or phrases from a given document. The words and phrases extracted are the ones that form the main topic of the document.

Similarly to LDA, RAKE performs no text pre-processing on its own and this is required as a precursory step. The pre-processed text is then split into words and phrases, which are groupings of words not separated by any stop words or punctuation. The next step is the calculation of word frequency for each word as well as the number of times two or more words appear in the same phrase, which is called co-occurrence. Based on the term frequency a list of candidate keywords is generated. For each of these candidates, a score is computed using the term's frequency and its co-occurrence with other words. Once the scores have been calculated, the candidates are ranked with the highest-ranking keywords being the most likely to represent a key concept of the document from which they were extracted. Finally, the top keywords are selected based on a threshold.

While RAKE offers satisfactory performance for general keyword extraction tasks, its value stems from the fact that it is a simple and efficient algorithm. Despite this, due to its simple approach, RAKE has the potential to struggle with very technical or domain-specific terminology and can produce keywords that are not relevant or descriptive of the main theme of the document.

3.2.2 YAKE (Yet Another Keyword Extractor)

Yet Another Keyword Extractor (YAKE) [7] is an extension of RAKE that builds upon the same ideas and introduces improvements that increase the relevance of extracted keywords as well as the general accuracy with which the most relevant keywords are extracted.

YAKE introduces a multitude of linguistic features such as identifying patterns, taking into account word boundaries and parts of speech for each word as well as grammatical rules to improve the relevance of extracted keywords. Additionally, it takes into account domain-specific information and terminology as well as context to improve the scoring of keywords. It also has improved methods of determining meaningful phrases and assigning a score to them as a whole. All this additional information is taken and introduced into the pre-existing RAKE algorithm to generate the top keywords for a document with improved relevance and accuracy.

These improvements come at the cost of reduced simplicity and increased computational intensity when compared to RAKE. Despite this, YAKE may be better suited to more diverse keyword extraction tasks because the limitations imposed by technical or domain-specific terminology have been alleviated.

3.2.3 KeyBERT

In addition to their usefulness in topic modeling, BERT-based models can be used to perform keyword extraction. The KeyBERT framework has been designed specifically to be used for this kind of task. For this reason, we picked KeyBERT as the technology we used for the keyword extraction approach to our solution.

KeyBERT uses word embeddings generated by a BERT transformer-based embedding model and cosine similarity to determine the key phrases, which can be of variable length, that are most similar to the document itself [2]. The way KeyBERT works is by first creating a document-level representation using document embeddings generated by the embedding model and subsequently extracting word embeddings from N-grams, which are words or phrases composed of two or more words [2]. Lastly, cosine similarity is used to identify the N-grams with the highest level of similarity to the document as these can be considered the words or phrases that offer the best description of the document [2]. One main advantage of using KeyBERT is the fact that it uses BERT-like models that are pre-trained for NLP tasks which eliminate the need to train a BERT model from scratch to cater to our task and thus speeds up development. The entire workflow described in this paragraph can be viewed in Figure 1.

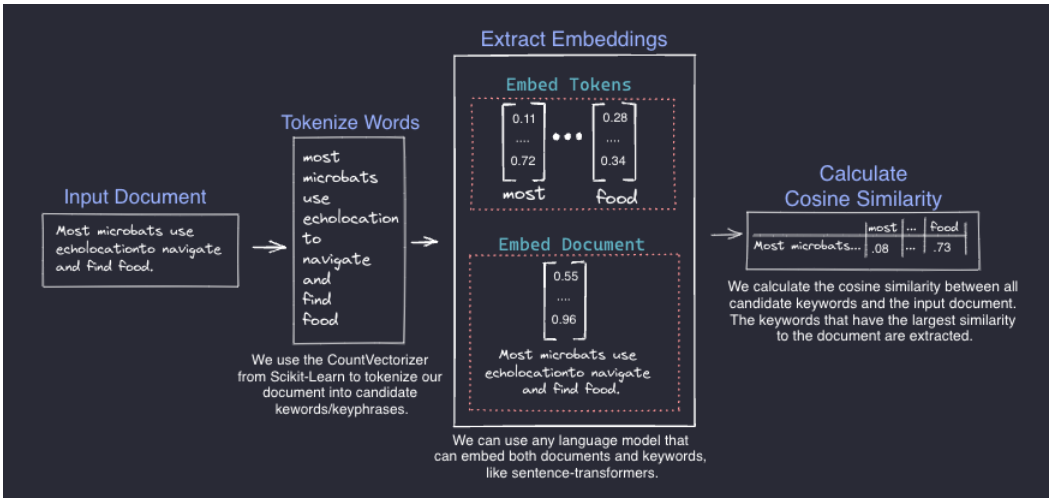


Figure 1: KeyBERT workflow diagram[2]

Due to the use of powerful, pre-trained BERT-based models and word embeddings which retain contextual information, this approach can offer better performance and cover a wider range of keyword extraction tasks than the alternatives mentioned previously.

[2]https://maartengr.github.io/KeyBERT/guides/quickstart.html

4 PROPOSED SOLUTIONS

This chapter will present the solutions we proposed for our task in depth both from the perspective of the architecture of the solutions as well as their usage. Initially, we will be presenting the pre-processing that needed to be done on the text from the dataset. We will follow up by presenting how we made use of LDA to model publications as a set of topics and compare them. Lastly, we will discuss how a variation of the BERT system has been used to extract sets of keywords from the various publications in the dataset and what metrics we used to determine the similarity of these sets of keywords.

4.1 Text Pre-processing

Because the entire contents of each paper were unavailable, difficult to aggregate, and fairly inefficient to use due to size, we resorted to using the abstract of each publication for our text analysis. As LDA does not do any text pre-processing and to ensure the text was as clean as possible so we can use only the most relevant parts of it we needed to perform some standard pre-processing beforehand.

For the pre-processing, we began by removing all unnecessary special characters from it that would ultimately offer no benefit in extracting the meaning from the text so that we can build the topics that would represent it. These special characters include punctuation characters, new line characters, tab characters, and other special characters that showed up throughout the various abstracts such as brackets and even list dots. In addition to this, we removed all numbers as they offer no descriptive information about the text. Finally, we converted all the remaining alphabetical characters to lowercase to remove the possibility of having duplicate words in topics due to capitalization.

The next step of the pre-processing was removing stop words as these words are very common words that may show up multiple times throughout a text and that offer no real meaning. This step was applied for both the English and Romanian languages with different stop word lists. The lists used represent the 40 most commonly used words in their respective language, including words like prepositions. After an initial analysis of the remaining words, using word clouds, which display all of the words in a text with the size of the word displayed being an indication of the rate at which it appears in the text, we noticed that a few other meaningless words remained with a high rate of usage, in the case of English publications and that some English language words showed up in the Romanian corpus of publications, most likely due to use of specific terms in English. As such, the lists of stop words were amended to include these words. Some examples of these words include use, result, datum, from, etc.

With only the most relevant words left in the text, we moved on to the next step of the process, which involved reducing all of the different variations of the same word, such as plurals or tenses, to their base, dictionary version. This process could be done using two methods, stemming and lemmatization. The main difference between these two techniques is that stemming involves a more raw process that removes prefixes and suffixes from words, in the attempt of getting the base version of that word, but without utilizing a predefined vocabulary to verify the correctness of the resulting word. On the other hand, lemmatization makes use of a predefined vocabulary and performs morphological analysis of the words to ensure the base versions of words are correct. While this process is more resource intensive and requires the use of a pre-trained model, the results are more accurate.

One issue that came up during this process was the fact that the lists of stop words used did not include all possible variations of those words. As such, performing the steps in the aforementioned order resulted in certain stop words still being present at the end of pre-processing. Because of this, the order was modified to perform lemmatization before stop word removal.

4.2 Topic Modeling with LDA

The first of the two approaches for solving the task at hand is using LDA to model publications or a corpus of publications as a set of topics. A topic is represented as a set of keywords, each with an attached weight that represents its importance to the topic. As the name suggests a topic represents a subject, an aspect discussed within the text from which it is extracted.

The LDA model requires the text to be processed as an input, in two different forms. Firstly, it requires the text, more specifically the words in the text, to be represented as a dictionary, in which each unique word gets an id assigned to it. Secondly, it requires the term document frequency of the text which is represented as a set of mappings from word id to the number of times that word appears in the text. After the text corpus has been inputted into the LDA model, the result will be a user-defined number of topics each represented by a set of related keywords relevant to the subject matter described by that topic.

In Figure 2 we can observe the topic distribution of a corpus of publications from author Gabriel Guțu-Robu with 10 topics modeled by LDA. Each circle in the plot on the left of the image represents a topic. The size of the circle is correlated to the prevalence of that respective topic. In essence, this means that the larger the circle is, the more prevalent that topic is in the given corpus. The overlap between the topic circles indicates that those topics are fairly related to one another, sharing a portion of their keywords. This results in those topics essentially offering less information in the greater context of the corpus. A good topic model will have larger circles with little to no overlap between them and scattered throughout the chart. A model built with too many topics will typically result in those topic circles having a high overlap, being smaller in size due to encompassing a smaller portion of the keywords,

and being more clustered in different parts of the chart.

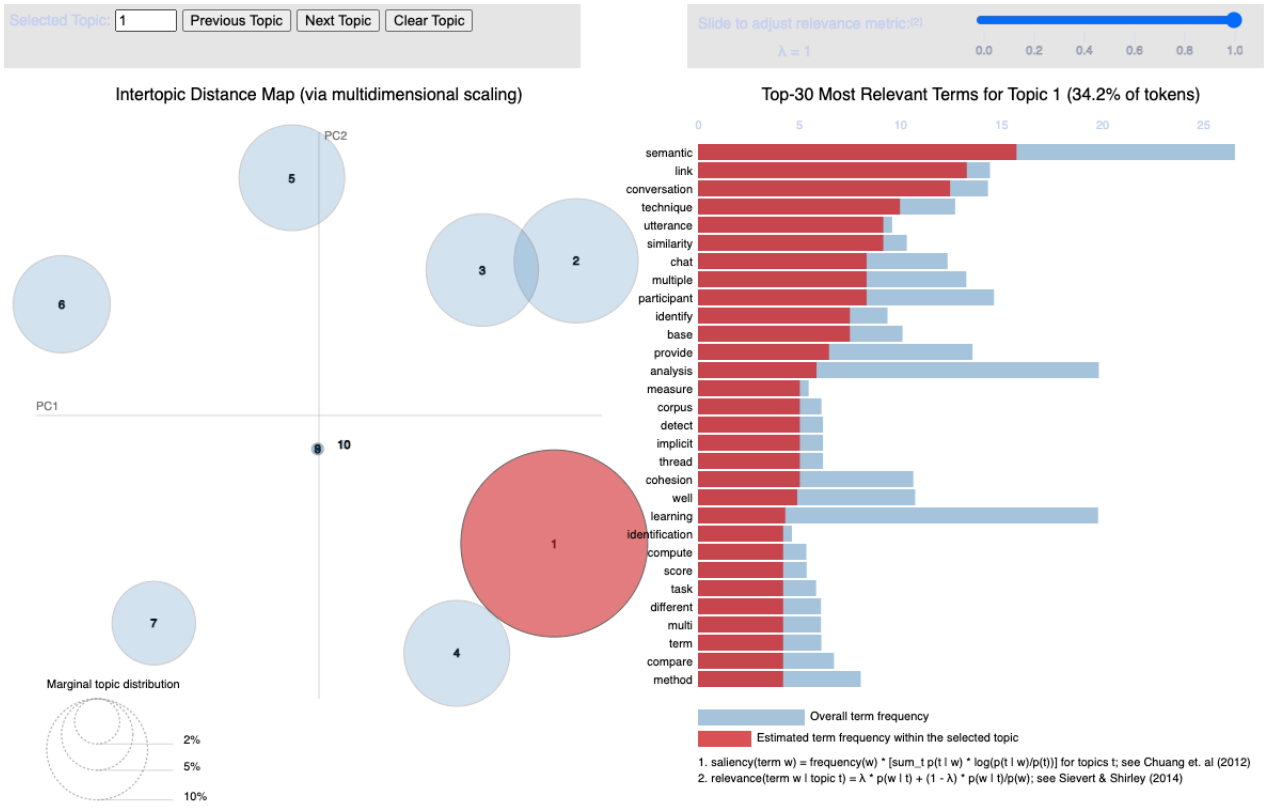


Figure 2: Topic distribution of corpus of publications from Gabriel Guțu-Robu

On the right-hand side of the figure, we can see, for the selected topic, some of the most relevant keywords for that topic. For each one of them, we can view the frequency of that term overall in the entire corpus (visualized as the blue bars next to each word) as well as the estimated frequency of that term relative to the topic to which it belongs (visualized as the red bars next to each word). We can also view, for the selected topic, the percentage of the total number of tokens, i.e. words, from the entire corpus that are part of that topic.

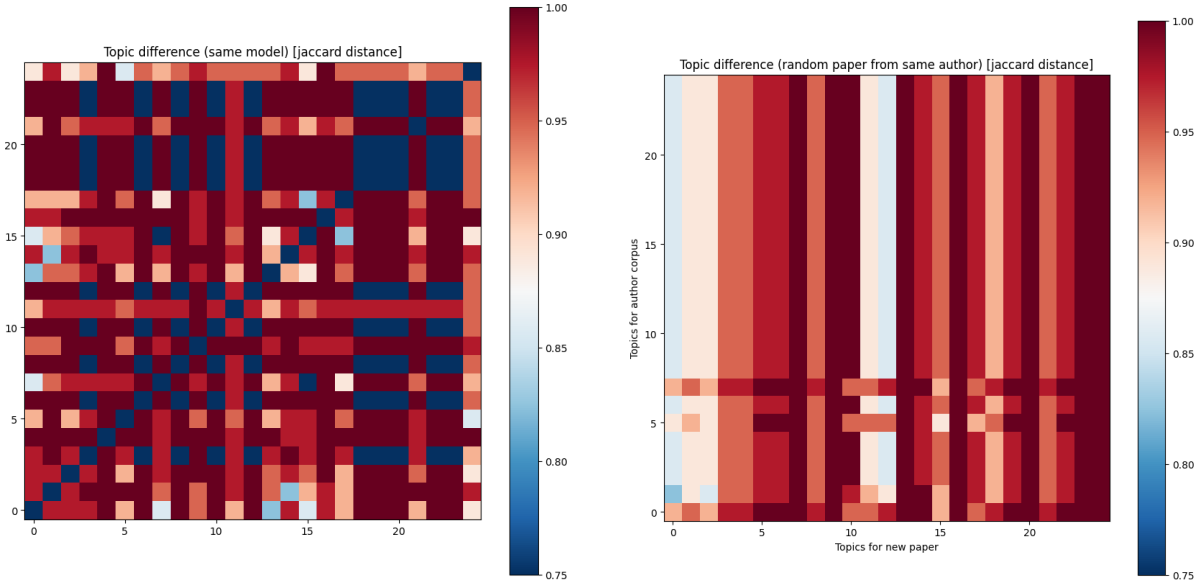
In the Tuning the Models Section we will discuss more about the approach taken to determine the optimal number of publications that the LDA model should extract to get the best understanding we can of the corpus analyzed.

When utilizing LDA to compare a new publication to an author's existing corpus of publications, both elements are modeled using an equal number of topics to facilitate comparison. The existing corpus of publications is represented as a larger, singular text, with all preprocessed abstracts of each of the individual publications merged. The similarity between the two is computed by calculating the difference in topic distribution between the two models. The LDA model offered by the Gensim framework provides us with the means to quickly compute the diff between the two models. This is done by applying one of various distance metrics, in our case, the Jaccard distance, also known as the Jaccard index, which calcu-

lates the dissimilarity between two sample sets. To ensure the topics from both models are comparable, we will only look at the top N words in each topic, by the weight with which they contribute to the topic. In this case, the N is another user-defined value. The difference between the two sets of topics is represented as a TxT matrix, where T represents the number of topics. Each element of this matrix, identified by the indexes i for row and j for column, is a value in the interval [0, 1] which represents the dissimilarity between the topic i from one corpus and the topic j from the other corpus. The closer to 1 the value is, the more dissimilar the two topics are.

In Figures 3 and 4 we can see multiple visualizations of the topic difference matrix between two sets of topics. These have been displayed as heat maps utilizing a color gradient from deep red to deep blue to indicate the level of dissimilarity between the two topics. The more red a square is, the higher the level of dissimilarity between those two topics. The two figures display 2 different types of scenarios, one positive and the other negative, from the perspective of the levels of dissimilarity.

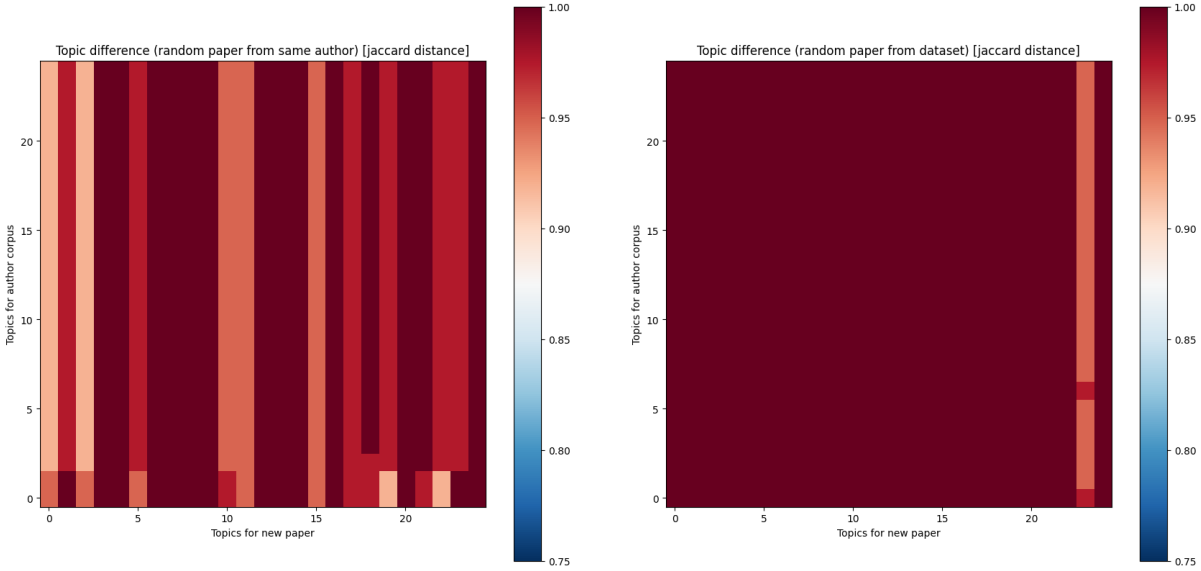
Due to each academic publication attempting to tackle a new problem with a new approach in a broad domain, the level of dissimilarity between two sets of topics is generally high, as we can rarely expect two papers to cover the same subject matter as well as the same approach. This required us to reduce the interval of dissimilarity values at which we look to the [0.75, 1] interval.



(a) Comparing one LDA model with itself

(b) Comparing topics of the corpus from one author with a new random paper from the same author

Figure 3: Visualizing the topic diff matrix as a heat map in positive scenarios



(a) Comparing topics of the corpus from one author with a new random paper from the same author that tackles a completely new subject matter

(b) Comparing topics of the corpus from one author with a new random paper from another author with unrelated work

Figure 4: Visualizing the topic diff matrix as a heat map in negative scenarios

Sub-figure 3a displays the topic difference matrix of an LDA model compared to itself. This could be considered the ideal case, in which there is perfect correlation represented by the deep blue squares on the main diagonal, each of these squares displaying the comparison of the topics in the model with themselves. In some cases, we can see a good correlation between different topics of the same model but most of the other pairs of topics have a low level of similarity.

Sub-figures 3b and 4a show the topic differences between papers from the same author and that author's corpus of published papers. We can see here significant differences from the ideal case and lower correlation only in the case of some topics. The difference between the two sub-figures stems from the fact that Sub-figure 4a represents a paper that covers an entirely different subject from what the author normally publishes and it could be considered a false-negative example. We will discuss further how this case is interpreted and how it can be fixed in the upcoming section regarding the Classification of similarity using SVM. The final sub-figure, Sub-figure 4b displays the comparison in the case of a random paper from a completely different author with a different area of expertise. In this case, we can see that the level of dissimilarity between this paper and the corpus of publications from the author used in previous comparisons is very high across all topics.

One additional benefit to modeling the topics using LDA is that it allows us to determine the dominant topic, alongside its keywords, for a document or each of the documents in the corpus. This could be further used to aggregate a list of the most common and most relevant keywords for that author's work, which, in essence, describes that author's interests and areas of research.

4.3 Keyword Extraction with KeyBERT

Before discussing how the results of the comparison are interpreted and how a paper is classified, we will first present the other method chosen for comparing a paper to an author corpus, as the classification methods are similar for both approaches. For our second approach, we used BERT, a transformer-based model, that is pre-trained and able to solve various NLP tasks. In our approach, BERT has been used to extract keywords from a single piece of text, a publication, or a corpus of texts, the multitude of papers published by an author.

For our task, we used KeyBERT [12], which offers a framework similar to the use of a BERT model, designed specifically for keyword extraction. By default, KeyBERT makes use of the "all-MiniLM-L6-v2" embedding model from Sentence Transformers [3] to generate the word embeddings. In the Tuning the Models Section we will discuss and compare the performance of using different embedding models offered by Sentence Transformers.

KeyBERT requires no pre-processing as all elements of a document are crucial for BERT to determine the general topic of the document, due to the use of document embeddings. As such, we can provide the model with the entire corpus of raw abstracts and a range that will dictate the size of N-grams to be considered. For example, using the range (1,1) would limit the extracted N-grams to just words, and using the range (1,2) would mean extracting both words and bi-grams, which are phrases composed of two words. The lower limit can also be increased in case singular words should not be extracted or in case the N-grams should have a minimum number of words that is higher. A comparison between using different N-gram lengths will later be presented in the Tuning the Models Section.

After extracting the keywords or key phrases using KeyBERT from both a singular, new paper from an author and the corpus of publications of that author, we needed to determine the similarity between the two. This was the main challenge of this approach as computing the similarity of the two sets of keywords was not as trivial as computing the difference between topics in the case of LDA. We decided to make use of the contextual information stored in the word embeddings and come up with a formula, taking inspiration from the formula for computing the similarity of two texts proposed in "Measuring the Semantic Similarity of Texts" by Corley et. al [8].

In our approach, we are replacing the semantic similarity metric that would have been taken from a source such as WordNet[1] with a contextual similarity computed by calculating the cosine similarity of the word embedding vectors that retain contextual information. When comparing two lists of keywords, the word relatedness will be calculated for each pair of matching keywords and we compute the sum of all of the relatedness scores to get the relatedness of the two keyword lists. When comparing a paper, represented as a list of keywords, with a corpus, represented as a list of lists of keywords, the total similarity score will be computed as the mean of all similarity scores between the paper's list of keywords and

[1] https://wordnet.princeton.edu

each of the lists of keywords extracted from each paper in the corpus.

Similarly to the approach in the case of LDA, a classifier will be trained and utilized to evaluate whether the paper can be considered a part of the author's corpus or an outlier based on the computed similarity score.

5 IMPLEMENTATION

In this chapter, we will be presenting the implementation-specific details of the project that are meant to offer more context behind the various technologies and frameworks used in the code of this project. The entire implementation can be found in the GitHub Repository created for this project [1].

5.1 Formatting the Dataset

As mentioned in the Dataset Section we needed to change some formatting elements of the dataset to allow for proper parsing and field separation.

Most commonly, CSV files make use of the comma character to delimit fields and double-quote characters (") to delimit strings. In this case, a string most commonly refers to a group of words or to any set of characters that also contains special characters, such as those used for delimiting or other characters that may break formatting. The issue encountered is the fact that various fields contained these delimiting characters within them causing the parsing of the entire file to fail. As a solution, we implemented a simple Python script to replace these characters with other non-conflicting characters to ensure field separation was successful. A challenge here was replacing these characters globally in such an order that ensured no correct delimiting was broken. The simplest way to do this was to replace the current string-delimiting characters, double quotes, with characters that could serve as delimiters and that were less likely to appear within the contents of these fields. We used single quotes (') to delimit characters and replaced all text occurrences of that character with back-ticks (‘). The order of the replacements ensured that all strings were now properly delimited and we did not need to worry about replacing the field separators as any time those appeared within a string, they would be ignored.

5.2 Data Access Functions

To ensure easy access to the dataset we implemented some functions that would allow us to retrieve various publications based on author. These have been used mainly when building datasets for classifiers. These functions have been built to be easily usable by future contributors to the project. From the perspective of someone unfamiliar with the dataset, the easiest way to identify an author is by their full name, as the user ids are internal to the CRESCDI platform and have no real meaning. With this aspect in mind, the functions are centered

around the author's full name, which can be easily inputted by the user.

The function that is mainly used in the project is "get_publications_by_user_name". In the case of this function, the term user is used from the perspective of the CRESCDI platform, where a user is someone who has an account and as such, has publications on the platform. To maintain consistency within the nomenclature of this documentation, the user name is equivalent to the author's name. The function first identifies the author id based on the inputted name, by checking if both the first and last name appear in the author list, regardless of capitalization and other middle names that may appear. Next, the author id is linked to the id's of the publications of this author and those are retrieved from the publication list. This function is extended by a separate one that allows extracting a paper, chosen by title or randomly from the author's corpus, and returning both that paper and the rest of the corpus separately. This is mainly used when we need to analyze a certain publication that is known to present a corner case in the context of our implementation.

For use within the code and use when building the dataset for the classifiers we also implemented functions that can retrieve publications based on author id as well as a function that retrieves the author id based on the author's full name.

A full implementation of these functions can be found in the GitHub Repository used for this project [1].

5.3 Stop-word Lists and Lemmatization Models

During the pre-processing section of the proposed solution description, we presented the use of lists of stop words used to filter out words that are very common and that have little meaning by themselves. We utilized the NLTK framework[1] for this task as it contains a multitude of tools for natural language processing. NLTK offers lists of stop words in multiple languages, as well as the ones that are of interest in the context of this project, Romanian and English. These lists contain the top 40 most common words in each language and can be extended to fit the needs of any project.

Another step in the pre-processing stage is that of text lemmatization. Lemmatization makes use of a vocabulary to make morphologically correct adjustments to words, in the process of obtaining the base word. For this, we used two lemmatization models, one for each language, which are pre-trained to perform this task. We obtained these models from the spaCy[2] framework, which is another framework that caters to natural language processing tasks. For the Romanian language, we used the "ro_core_news_lg"[3] model, which is the largest model offered for Romanian and is pre-trained on a large dataset of news articles. For the English

[1] https://nltk.org
[2] https://spacy.io
[3] https://spacy.io/models/ro

language, we used the "en_core_web_trf"[4] model, which is the most advanced model offered for English, based on the RoBERTa transformer model and pre-trained on a large corpus of written text available on the internet. We decided to use the most powerful models available as these ensured the best results for lemmatization.

5.4 LDA Hyper-parameters

In this section, we will be discussing the use of the various hyper-parameters available in the LDA model we used. The number of topics parameter will not be presented in this section as this is a parameter we tuned and it will be presented in a later section.

The "chunksize" was left as the default value due to this being sufficient for the size of our dataset and not impacting the performance of the module. The "passes" parameter was increased to 5 from the default of 1 to ensure better training of the model on the dataset. The "alpha" and "eta" parameters are presented in greater detail in the paper "Online Learning for LDA" by Hoffman et al. [13] but for this implementation, we used the default "symmetric" value for "alpha" and the "auto" value for "eta" as opposed to the default as this would allow the model to adjust this parameter based on the corpus. Using a similar scale to the number of passes, the "iterations" parameter has been increased from the default 100 to 500 to ensure better inference from the dataset. The "eval_every" parameter has been set to 1 to ensure that the weights of the keywords in the topics get recalculated after every update to the topic contents to ensure a more accurate topic representation, at the cost of an increased runtime.

To speed up the LDA model we used the LDAMulticore[5] model offered by the framework as this model could be parallelized by simply setting a "workers" parameter. We set this parameter to 4 as the machine we ran the project on had 4 cores and we were able to observe a performance improvement.

Finally, to generate topics, LDA starts by randomly assigning the initial keywords to each topic. To ensure reproducibility from one run of the project to another we set this parameter to a static, randomly generated integer to serve as the seed on which the random initial state is built.

5.5 Per-author Classifier and All-author Classifier

To mimic the way the classification system would be deployed in production we decided to scale this process to analyze more authors. To do this we attempted two different approaches, one which involved having one classifier trained and predicting for a single author, and one

[4] https://spacy.io/models/en

[5] https://radimrehurek.com/gensim/models/ldamulticore.html

classifier trained with data points from all authors. This all-author classifier approach is viable due to the data points containing essentially no links to which author they came from. From the classifier's perspective, the data is a series of topic difference matrices, in the case of LDA, or similarity scores in the case of KeyBERT, and a series of labels that indicate whether those values are from comparing a paper of the same author or not.

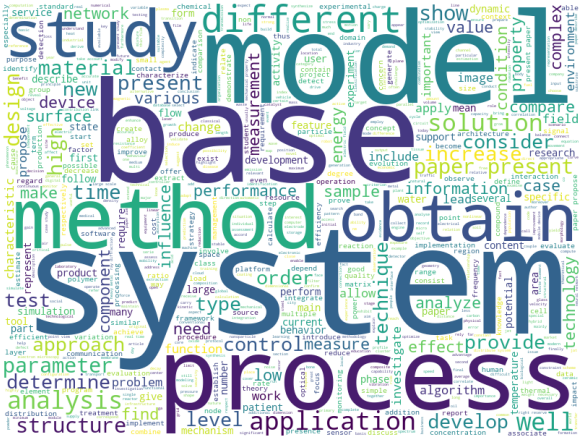
To test the two approaches we needed to build a larger dataset for the classifier containing data from multiple authors. This process turned out to be extremely computationally intensive and very slow in terms of runtime. Due to this, we were required to lower the number of authors and publications per author we used to create the dataset for the classifier. We decided to only use papers from the authors in the CRESCDI dataset that had more than 20 published papers as this criterion sufficiently reduced the number of authors to 88 and ensured that the authors had enough papers so that the topics generated would be diverse enough to cover a multitude of cases. For each of the selected authors we sampled 10 papers from their corpus and as described previously, also selected 10 random papers from other authors. As such, the dataset consisted of 20 data points per author.

6 EVALUATION

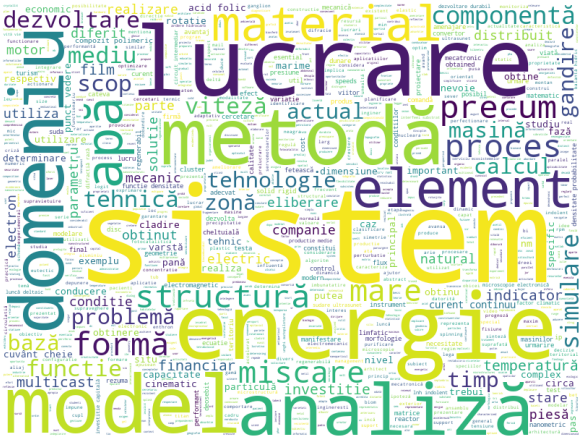
The Evaluation chapter has the role of presenting the various methods used for evaluating our proposed solutions using the provided dataset. This chapter will present the results of these evaluations as well as some visualizations that will showcase the contents of the dataset used when implementing our solutions. We will also mention some of the model tuning techniques used to improve the performance of both LDA and KeyBERT.

6.1 Keyword Visualizations

To ensure a better understanding of the contents of the dataset that were most important in implementing our solution, the abstract texts of publications, it is useful to visualize them from the perspective of the models that are using them. These models view these texts as a multitude of words that need to be grouped into topics, in the case of LDA, or that need to be analyzed in the context of the text containing them, in the case of KeyBERT. As such, it is useful to visualize these words both at the dataset level and at the author level to get a better understanding of their diversity or the topics and contexts these words could be grouped into. The texts these words are extracted from have already been pre-processed so that the remaining words are the most meaningful ones.



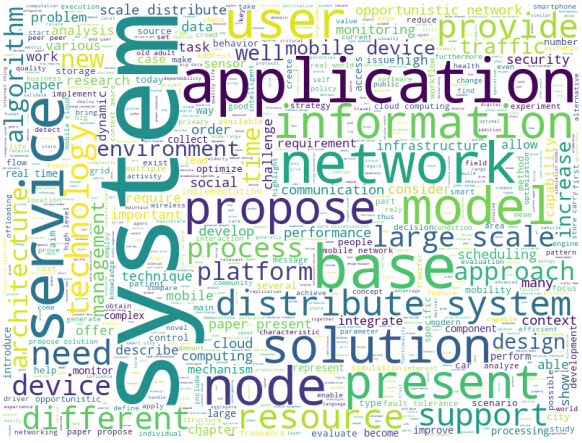
(a) English words from the entire dataset



(b) Romanian words from the entire dataset

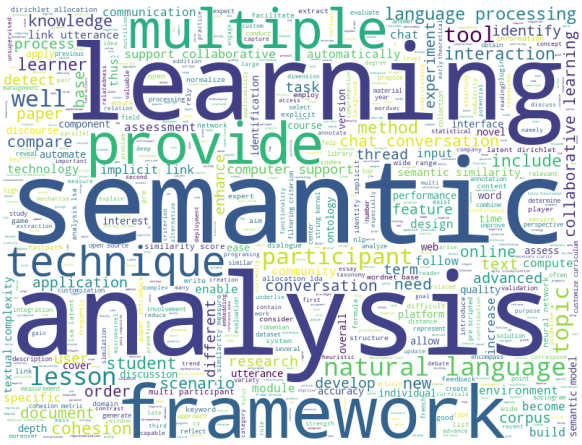
Figure 5: Words from the entire dataset

In Figure 5 we can see a visualization of the words in the entire dataset, with the size of the word representing how frequently that word is used throughout all of the abstracts. This gives us an insight into the types of topics we can expect to obtain from the various publications available.

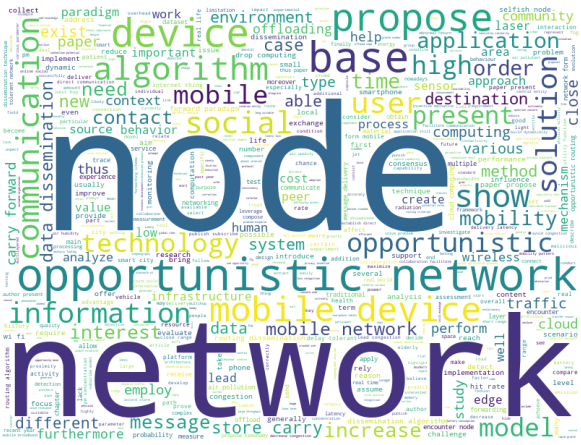


(a) English words from the corpus of author Ciprian Dobre

(b) Romanian words from the corpus of author Ciprian Dobre, Professor



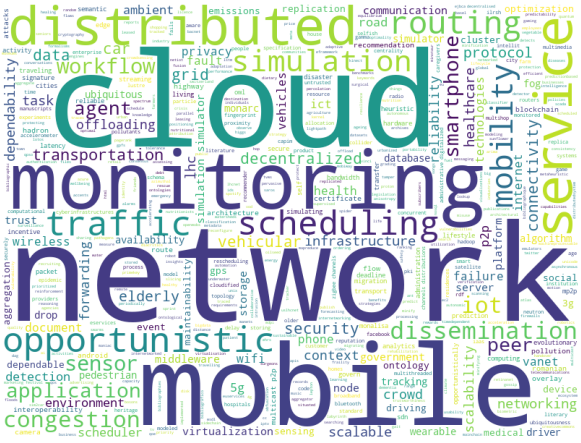
(c) English words from the corpus of author Gabriel Guțu-Robu



(d) English words from the corpus of author Radu-Ioan Ciobanu

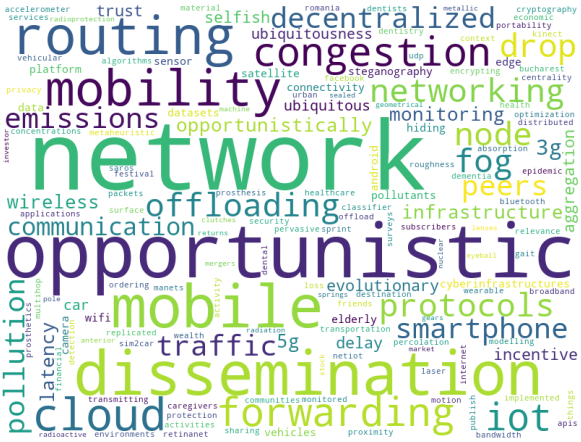
Figure 6: Words from the corpora of various authors analyzed while implementing the project

In Figure 6 we can view the words used most often by some of the authors we analyzed during the research and implementation of this project. This allows us to get an initial idea of the subject matters tackled by these authors throughout their publications. Further, we can make use of KeyBERT to extract the most representative words for these same authors. These words can be used to determine the interests and the area of expertise of the authors which could be used in the CRESCDI platform. These words can be viewed in Figure 7 displayed below. This figure shows us the aggregated top 10 most relevant keywords per publication in each author's corpus.



(a) Top English keywords extracted by BERT from author Ciprian Dobre

(b) Top English keywords extracted by BERT from author Gabriel Guțu-Robu



(c) Top English keywords extracted by BERT from author Radu-Ioan Ciobanu

Figure 7: Top keywords extracted by BERT from various authors analyzed while implementing the project

6.2 Classification of Similarity using SVM

As mentioned in previous sections, interpreting the results of the two methods used to determine the similarity has been performed using an SVM classifier. Since we needed to determine whether or not the new paper could be considered part of the corpus, it made sense to interpret this task as a binary classification task. For the classifier, a label of 1 means that the paper belongs to the author, and a label of 0 indicates the opposite.

From the user's perspective, the result of this classification can be interpreted as the paper's content seemingly matching the author's usual subjects and as such requiring little to no manual verification from the user in the case of a positive result from the classifier. In the case of a negative result from the classifier, the user should receive visual feedback which serves as an indication of the fact that the paper should be manually reviewed to determine whether the classifier's decision is correct and the paper does not belong to the author or

the classifier's decision is incorrect and the paper does belong to the author but tackles a new subject from the ordinary. These types of exceptions should be flagged and the classifier should be retrained through online learning to cover similar new papers in the future.

To train the classifier, a dataset containing the values obtained from the previous two methods needed to be created. As the two methods have different formats for their outputs, it was unfeasible to use a single classifier or a single dataset for both. In the case of LDA, the dataset would be represented as a list containing the flattened topic difference matrices for all papers currently available in the CRESCDI dataset presented earlier for an author. By flattening the matrices, we essentially represent each difference of a pair of topics as a set of features the SVM classifier needs to look at. To ensure the topic differences are accurate and represent the real case of a new paper being compared to the existing corpus, we take the entire corpus, eliminate the current paper we want to compare from it, and compute the difference between the topics generated by an LDA model for the paper and the topics generated by an LDA model for the other papers in the corpus. To this set of values, we added an equal-sized set of topic differences from papers that did not belong to the author to ensure that the dataset will be balanced between positive and negative examples. These negative values are generated by computing the difference between the topics generated by an LDA model for a random paper from a different author and the topics generated by an LDA model for the author's corpus of papers. This process will give us our input data, and alongside it, we generated a list of labels, 0 or 1, for each data point, that indicated whether that data was generated from a paper from the same author or not.

We tested out the classifier by creating a dataset from all publications available from author Gabriel Guțu-Robu and excluding the two papers we saw analyzed earlier in the LDA section in Sub-figures 3b and 4a, one paper which covered similar subjects to the author's usual area of interest and one paper which contained a completely different subject. After building the dataset for SVM using the process described above we performed an 80/20 train-test-split so we could evaluate the classifier as well. The training data was first normalized and then used to fit the SVM classifier to it. After fitting, we normalized the test data and used the classifier to predict the labels for it. The resulting labels were then compared to the labels set while the dataset was being built and we used both Accuracy and F1 score as metrics to determine the performance of the classifier. For this author, we obtained an F1 score of 0.86 and an Accuracy score of 0.89.

We then ran the classifier on the two papers we excluded from the training and testing sets and a random paper from a different author. As expected, the paper from the same author that covered a usual subject for the author was predicted as belonging to the author. Similarly, the paper that was from a different author was predicted as not belonging to the same author. Finally, the paper from the same author which covered a completely different subject was flagged as not belonging to the author. This made sense due to it bearing little to no resemblance to the rest of the author's corpus and served as a good example of how user feedback will tie into the project when deployed, as this paper will need to be flagged by

the user as belonging to the author and introduced into the classifier's dataset for retraining.

For the KeyBERT approach, we built our dataset and used an SVM classifier in a similar way to what is described above. The only difference is that in this case we only have the similarity score to work with. In terms of the dataset, this will be represented as a list of single-feature data points, i.e. the similarity scores. We trained and tested the classifier using the same process for the same author and obtained an F1 score of 0.89 and an Accuracy score of 0.91, a small improvement from using LDA to determine similarity. Despite this, the classification of the three other papers remains the same.

The results presented above have been obtained using the tuned LDA and KeyBERT models. The tuning process is described in detail in the following section.

6.3 Tuning the Models

To ensure that we get the most out of the LDA and KeyBERT models, we attempted to tune the models by adjusting certain parameters. In the case of LDA, the main parameter we looked at was the number of topics generated. To tune the number of topics we used models using different topic counts in the range [1, 5, 10, 15, 20, 25]. To measure the performance difference caused by using different numbers of topics we used the coherence score to determine the best-performing model. We ran the model on the corpus of publications from the author Gabriel Guțu-Robu and calculated the coherence score for each model using a coherence model from the same framework we used for LDA. We plotted out the values and analyzed the graphs to determine the most optimal number of topics. We wanted to select the lowest number of topics with the highest coherence score as further usage and testing of the model showed that an increase in the number of topics would result in lower performance for the LDA model when computing topic differences and lower accuracy and F1 score for the classifier trained on topic differences. As previously stated, a higher number of topics does not always result in better performance due to topics starting to overlap and have less meaning.

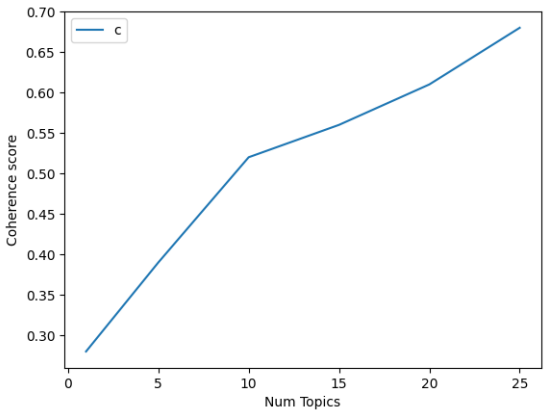


Figure 8: Evolution of coherence score correlated with the increase of topics

In Figure 8 we can observe the most common evolution of the coherence score, correlated

with the increase in the number of topics used for the model. In this case, the coherence score peaks when using 25 topics. Due to the random state that LDA uses to initially assign keywords to topics, the evolution can sometimes vary with the coherence score peaking at a lower number of topics. This can be observed in Figure 9 below, with the coherence score peaking at 15 topics.

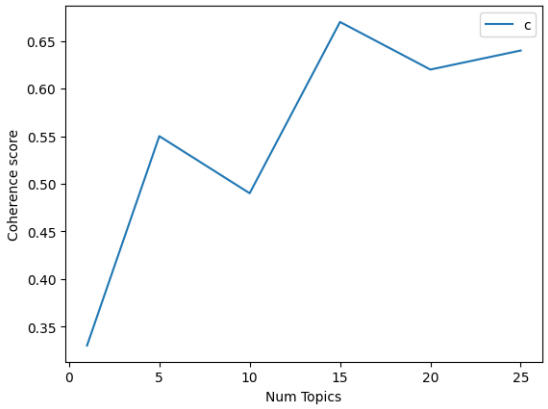


Figure 9: Evolution of coherence score with the peak at a lower topic count

We also tested with a higher range of topics, going up to 50, but observed the coherence score starting plateau also around 25 topics. This behavior can be observed in Figure 10 displayed below. As such, we decided to keep the range as initially stated and run the optimization using that.

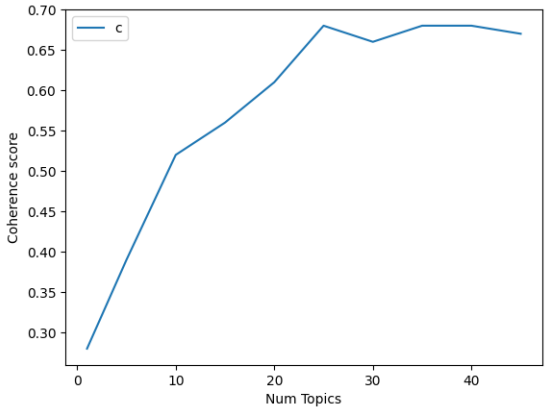
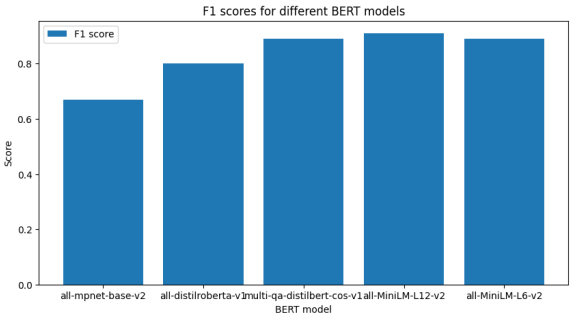


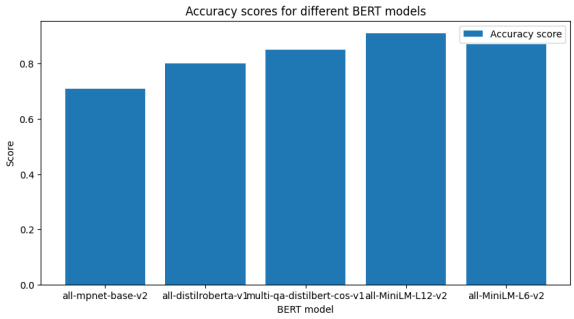
Figure 10: Evolution of coherence score with a higher range of topic numbers

When attempting to tune KeyBERT we looked at using different embedding models from the ones offered by Sentence Transformers. We selected 5 of the top performing models according to the model listing [3] for this test. One of these models, specifically "all-MiniLM-L6-v2", is the default one used by KeyBERT. The other ones we selected are "all-mpnet-base-v2", "all-distilroberta-v1", "multi-qa-distilbert-cos-v1", and "all-MiniLM-L12-v2". Apart from "multi-qa-distilbert-cos-v1", all the other models are pre-trained and tuned for general purpose, with the aforementioned one being pre-trained and tuned for semantic search [3]. To analyze the difference in performance between the models we used the F1 and accuracy scores of a

classifier trained and tested using similarity values computed from the same corpus used when tuning LDA.



(a) F1 scores for various embedding models

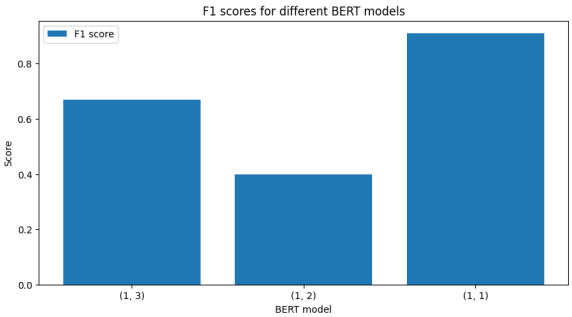


(b) Accuracy scores for various embedding models

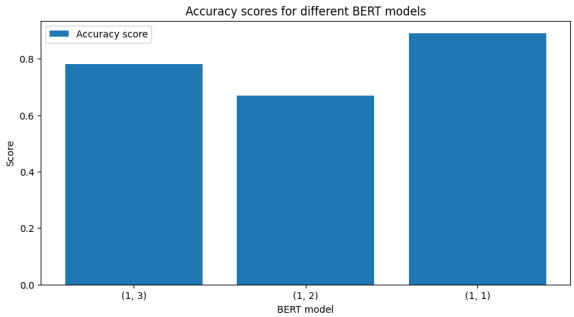
Figure 11: Visualizing F1 and accuracy scores when using different embedding models

In Figure 11 we can observe that when classifying similarities of keywords using the "all-MiniLM-L12-v2" embedding model, we get the best results in terms of F1 and accuracy scores. Following this analysis we continued using this model for the next steps in our evaluation process.

Another parameter we tuned for KeyBERT was the size of the N-grams extracted by the model. We tested using monograms, bigrams, and trigrams and performed the same type of analysis described above.



(a) F1 scores for various N-gram sizes



(b) Accuracy scores for various N-gram sizes

Figure 12: Visualizing F1 and accuracy scores when using different N-gram sizes

In figure 12 we can observe the results obtained when testing out different N-gram sizes. The (1,1) indicator describes the fact that in that case, we are using only monograms, the (1,2) range indicates that we are using both monograms and bigrams, and the (1,3) range indicates that we are using monograms, bigrams, and trigrams for our extracted keywords/key-phrases. Following this analysis we can observe the fact that using only monograms offers us the best performance, and because of this we continued using only monograms for the rest of the evaluation process.

6.4 Per-author Classifier Results

One of the approaches tested out when trying to scale the classification process to multiple authors was using one classifier trained for each author. Currently, the main issue with using this approach is that with the relatively low number of publications some of the authors have on the platform, the dataset used to train the classifiers for each of those authors will be small in size. This will result in those classifiers not learning properly due to not being able to analyze a wide variety of potential values and as such, new publications for those authors will have a high likelihood of being classified incorrectly. To test out the viability of this option, we used a small sample of authors and a partial set of their publications as described in the Per-author Classifier and All-author Classifier Section. A part of the results of each of those classifiers can be viewed in the tables in Appendices A for LDA and B for KeyBERT. The complete tables have been truncated due to size.

After analyzing the results, we can observe diverse scores across all authors. We can also observe a fairly high number of scores of 1.0 which can be considered an indication of a dataset being too small. In these cases, the classifier may not generalize well and may struggle to correctly identify new cases. With the number of publications in the platform increasing over time, the classifier datasets may reach a sufficient enough size to ensure that after training the classifiers can generalize correctly for each author and have a high accuracy when classifying new papers. As such, the disadvantages of this method are the fact that while accuracy may increase over time it may take a prolonged amount of time for this to happen and the initial accuracy may lead to a high amount of mislabeling for authors with few publications. For these authors, the retraining will need to occur after every new publication is uploaded to the platform which will be more computationally expensive.

6.5 Single Classifier Results

After observing the results of the per-author classifier approach, we can discuss the improvements that having one classifier for all authors can bring. In this case, the classifier would be trained using the topic difference values from all authors or a large sample of authors. This is possible because the data used by the classifier does not contain any actual links to the original author or paper which it came from. With the classification being binary, the classifier only tells us whether the paper belongs to the author or not, regardless of the context in which the prediction is made. Additionally, due to having access to a larger dataset initially, the classifier will be able to generalize better and offer more accurate predictions for new papers. Another benefit stems from the fact that, with the dataset being diverse enough, the classifier could be retrained at specified intervals of new papers with just a sample of the new papers uploaded since the last training cycle, instead of retraining after every new paper is uploaded. This will reduce the computational load while still ensuring that the training dataset is diverse enough.

Using the same datasets used in the per-author approach and merging them into a singular dataset resulted in the classifier having an F1 score of 0.85 and accuracy score of 0.86 in the case of LDA and a F1 score of 0.89 and an accuracy score of 0.9 in the case of KeyBERT. This shows us that the classifier still retains a fairly high accuracy when trained over a much larger dataset from diverse authors and publications. We can also observe here that the usage of KeyBERT in the case of this task offers us a higher accuracy as opposed to using LDA.

7 CONCLUSIONS

In this final chapter of the project, we will be summarising what we set out to achieve, how we attempted to solve the task at hand, and what results we obtained from this project. In the end, we will also present what improvements and further work can be done to this project to prepare it for integration in the CRESCDI platform, as well as scale it and improve the performance or the result.

7.1 Key Takeaways

The proposed solutions offer two different approaches that solve the issue of automatically determining whether a new paper that has been assigned to a user on the CRESCDI platform for review has been assigned correctly, with it having a high similarity to the author's pre-existing corpus of publications, or incorrectly, with the paper tackling vastly different subjects and potentially indicating a case of author mismatch. These solutions intend to reduce the need for manual review from the user for a newly registered publication on the platform. The two approaches investigate two different techniques for extracting the contents of a publication as well as the interests of an author and for comparing new papers to pre-existing corpora.

Upon classification of similarity values and evaluation we can determine that, despite the difference being small, the keyword extraction approach offers better results with an F1 score of 89%. Additionally, due to the keyword extraction approach making use of pre-trained transformer-based models, it eliminates the need for training on a personal dataset and instead makes use of the knowledge gained during training for general-purpose tasks through transfer learning. As such, this approach should be preferred over the LDA approach.

7.2 Further Work

While this project is not ready for production deployment, the research and results presented throughout this document serve as a good base for further expansion. Getting this project ready for deployment would involve a slight restructuring of the code base to be able to be integrated into a full-stack application, improvements on scalability, and the implementation of support for using a GPU for all training as this would offer better performance. The implementation of a flagging and online learning system would also benefit the application as the system's accuracy in detecting outliers will be able to improve as more publications are added to the platform. Additionally, by using the keywords from the dominant topic

extracted by LDA for each document in a user's corpus of publications or by using the lists of keywords extracted by BERT alongside their contextual meaning, i.e. their word embeddings, the system on the platform that automatically extracts an author's interests via keywords could be improved.

In terms of research, this could be taken further to determine better methods and algorithms for comparing topics in the case of LDA, other formulas for determining the similarity of keyword sets in the case of the BERT-based keyword extraction, and other ways of classifying similarities that may offer improved performance. Additionally, other technologies described in the State of the Art section could be trialed to determine whether there would be better-suited alternatives to the current ones.

BIBLIOGRAPHY

[1] Analysis of scientific competencies code repository. `https://github.com/TudorPescaru/AnalysisOfScientificCompetencies`. Last Edited: 22.06.2023.

[2] Keybert documentation and api reference. `https://maartengr.github.io/KeyBERT/index.html`. Last Accessed: 20.06.2023.

[3] Pre-trained models offered by sentence transformers. `https://www.sbert.net/docs/pretrained_models.html`. Last Accessed: 22.06.2023.

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

[5] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes, and Adam Jatowt. Yake! collection-independent automatic keyword extractor. In *Advances in Information Retrieval: 40th European Conference on IR Research, ECIR 2018, Grenoble, France, March 26-29, 2018, Proceedings 40*, pages 806–810. Springer, 2018.

[8] Courtney Corley and Rada Mihalcea. Measuring the semantic similarity of texts. In *EMSEE@ACL*, 2005.

[9] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20:273–297, 1995.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[11] Maarten Grootendorst. Bertopic: Neural topic modeling with a class-based tf-idf procedure. *arXiv preprint arXiv:2203.05794*, 2022.

[12] Maarten Grootendorst, Abhay mishra, Art Matsak, Priyanshul Govil, Yuki Ogura, vincent d warmerdam, and yusuke1997. Maartengr/keybert: v0.7.0, November 2022.

[13] Matthew Hoffman, Francis Bach, and David Blei. Online learning for latent dirichlet allocation. *advances in neural information processing systems*, 23, 2010.

[14] Muhammad Omar, Byung-Won On, Ingyu Lee, and Gyu Sang Choi. Lda topics: Representation and evaluation. *Journal of Information Science*, 41(5):662–675, 2015.

[15] OpenAI. Gpt-4 technical report, 2023.

[16] 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.

[17] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. Automatic keyword extraction from individual documents. *Text mining: applications and theory*, pages 1–20, 2010.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

A PER-AUTHOR CLASSIFIER RESULTS FOR LDA

author_id	f1_score	accuracy_score
1455	0.5	0.5
562	0.8	0.75
1672	0.5	0.5
841	0.66	0.75
534	0.8	0.75
829	0.8	0.75
1171	0.5	0.5
1225	0.66	0.75
1246	1.0	1.0
872	0.66	0.75
712	1.0	1.0
1284	0.5	0.5
584	0.5	0.5
1151	0.8	0.75
1127	1.0	1.0
601	1.0	1.0
4	0.8	0.75
1612	1.0	1.0
946	0.5	0.5
844	0.5	0.5
1244	0.66	0.75
558	1.0	1.0
1700	0.66	0.5
859	1.0	1.0
2218	1.0	1.0
360	0.5	0.5
1849	0.66	0.75
1297	0.66	0.75
1228	0.5	0.5
2252	1.0	1.0
855	1.0	1.0

B PER-AUTHOR CLASSIFIER RESULTS FOR KEYBERT

author_id	f1_score	accuracy_score
1455	0.5	0.5
562	0.4	0.25
1672	0.67	0.5
841	0.5	0.5
534	1.0	1.0
829	0.67	0.75
1171	0.4	0.25
1225	0.67	0.75
1246	1.0	1.0
872	0.67	0.75
712	0.4	0.25
1284	0.67	0.75
584	0.8	0.75
1151	0.8	0.75
1127	0.67	0.75
601	0.86	0.75
4	0.67	0.5
1612	0.4	0.25
946	0.67	0.5
844	0.67	0.75
1244	0.67	0.75
558	0.67	0.75
1700	1.0	1.0
859	0.4	0.25
2218	1.0	1.0
360	0.8	0.75
1849	1.0	1.0
1297	0.8	0.75
1228	0.5	0.5
2252	1.0	1.0
855	0.4	0.25