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

DIPLOMA PROJECT

Anomaly Detection on Scientific Publications

David-Gabriel ION

Thesis advisor:

Prof. dr. ing. Ciprian DOBRE

BUCHAREST

2022

CONTENTS

SINOPSIS

Această lucrare propune o metodă de a determina în mod automat când un autor publică un articol științific într-un domeniu în afara ariei sale de specialitate. Sistemul poate rezolva situația când există doi autori cu același nume și o publicație a cărui autor este una din aceste două persoane, oferind un scor proporțional cu probabilitatea ca fiecare autor să fi scris respectiva publicație. Folosind algoritmi de învățare automată, se pot determina atât domeniile asociate unei publicații științifice, dar și ariile de specializate ale unui autor. Pe baza acestui profil, se poate determina cât de probabil este ca acel autor să fi scris publicația nou analizată. Abordarea curentă folosește modele preantrenate și metode de învațare automată nesupervizată si auto-supervizată, putând atinge performanțe mari chiar și atunci când e folosit un set de date mai mic, oferind utilitare de vizualizare menite să valideze performanțele modelului.

ABSTRACT

This work proposes a method for automatically determining when an author publishes a scientific paper in a field outside his expertise. The system can solve the situation when there are two authors with the same name and a publication whose author is one of those two people, offering a score proportional with the probability that each author wrote the aforementioned publication. Using machine learning algorithms, it is possible to determine both the topics or keywords associated with a publication and the fields an author is proficient in. Using this user profile, it is possible to estimate how likely it is that the author wrote the newly analysed publication. The current approach uses pretrained transformers and unsupervised and self-supervised learning techniques, which can achieve good performance even when using smaller datasets, while also providing visualisation tools meant for validating the model's performance.

1 INTRODUCTION

This project came to be when the need for an automated importing of publications mechanism was needed for the CRESCDI Platform. When a publication is imported, it is desirable to link it to the user who wrote it. In order to solve this issue, an anomaly detection algorithm was proposed, since having an author publish an article in a domain outside their area of expertise would clearly be flagged as an anomaly. While this project does not specifically focus on anomaly detection per se, since there are sufficient positive and negative examples in the dataset, it borrows the intuition from it.

One main assumption which was made is the fact that publications can be classified as belonging to a topic, and users generally publish articles in their topic or topics of expertise. While this is a fairly sane assumption, as the vast majority of authors only occasionally stray from their general field, the implementation may need to take into account several complex aspects, such as the relations between the topics and the appearance or disappearance of certain topics (such as block chain and outperformed algorithms).

1.1 The Problem

The main problem that this project aims to solve is the name ambiguity caused by the fact that there can be multiple different authors with the same name. When a new publication is added to a database, it is desirable to have its authors associated with the publication. This is sometimes difficult, since the publications can come from different sources and having different formats.

1.2 Objectives

The scope of the project is to provide a numerical score proportional to the probability that a given author wrote a given publication. Another objective is to update the model automatically as new publications are added, thus removing the need to periodically retrain the entire system.

1.3 Proposed Solution

The proposed solution is to extract the author's name from a new publication and test the publication against all authors in the database with the extracted name. If the model assigns

a score larger than a threshold, then the association between the publication and the author is marked as likely, otherwise it is marked as unlikely. All associations will pass through a manual review phase, where either the user or a site administrator can decide whether the association is correct or not, and they can see the model's predicted score. If the association is not reviewed in a given time, then the decision falls to the system. As a result, the system will not make any unsupervised decision without the opportunity for human intervention and will mostly aid in sorting the associations by likelihood and graphically augmenting the user interface.

In order to actually provide the aforementioned score, the system will be trained on the CRESCDI Research dataset. The publications' abstracts will be inputted into the system, which will determine the associated features for each publication. In other words, each publication will have a feature vector associated, where each entry in the vector intuitively represents how closely the publication matches each topic. With this, we can construct the user profile of each user, which indicates the topics they are proficient in. The score is given by the similarity between the newly added publication and the user profile.

1.4 Results

The problem was solved using two different methods: Latent Dirichlet Allocation and a proposed solution, which uses pretrained transformers. When using Latent Dirichlet Allocation, the accuracy of the system is approximately 84%. This result is while not using transfer learning or end-to-end training, meaning it can be drastically improved with those two enhancements alone. When using a pretrained transformer, the accuracy increases to 91%. This result does not benefit from end-to-end training, but illustrates how much transfer learning can increase the accuracy of a model.

1.5 Structure

The following sections present some of the background concepts needed to describe the solution, some related works in the field of topic modelling, the general proposed solution, which allows for various topic models or transformers to be used, implementation details and problems encountered while developing the solution, the evaluation on both the given and a standard dataset, and finally, the conclusion, along with potential future improvements.

2 REQUIREMENTS

The functional requirements are set by the client, the CRESCDI platform. During the client interviews, the main requirement that the project must meet is the ability to predict whether a given author has written a given publication. Another requirement is dynamic updating of the model. Since there is a live influx of publications being imported, it is important to have the model able to process new publications after the initial training period.

The provided dataset provided by the client contains a list of publications, a list of users and a list of correlations between users and publications, each correlation indicating if the user is confirmed to be the publication's author, if the user is confirmed to not be the publication's author or if their relation is not verified. Given the fact that the publications are provided without predetermined topics, the model must train without needing them, but also determine the topic or topics for each publication in order to manually validate it's correctness.

2.1 Functional Requirements

To summarize, the system should be able to:

- Train without the need for publications labeled with topics
- Describe an author's area of expertise based on their previous publications
- Predict whether an author wrote a newly added publication
- Dynamically update an author's profile based on newly added publications
- Generate human-friendly statistics which validate the model's correctness and performance

2.2 Nonfunctional Requirements

In terms of nonfunctional requirements, regarding the accuracy and performance of the system, there are several aspects to be discussed. Firstly, the prediction accuracy should be reasonably high such that the system is actually useful. Given the fact that supervised training is not possible, we can expect to not hit close to 100% accuracy, but this should be alleviated by the use of transfer learning.

Secondly, in terms of performance, this system is not a critical part to be integrated into the CRESCDI platform, since the system only serves as guidance for the already present human intervention, which will not be removed after the integration with our system. As a result,

it should not require exaggerated hardware resources. Not requiring a graphics card is a nice-to-have feature, but if that dramatically improves performance it may be acceptable. Also, the system must be able to scale well with the influx of publications. The number of publications has been exponentially increasing, exceeding 7 million publications per year [5]. That is about 14 new publications each minute. Even though not all of them will be inserted into the CRESCDI platform, this serves as an upper bound for the performance we should aim for.

To summarize, the system should:

- Be able to run on commodity hardware, without the need for a GPU, and achieve reasonable performance
- Achieve at least 90% prediction accuracy
- Process a new publication in less than 1 second
- Require at most 4GB of memory in order to make predictions

3 BACKGROUND

Natural Language Processing is a branch of artificial intelligence which focuses on understanding and extracting information from natural language, including text speech. Machine learning algorithms have shown incredible results when applied to classic NLP tasks, especially since the apparition of transformers and the attention mechanism [13] [4].

Topic modelling is the process of analysing a collection of documents, identifying the topics that appear in the corpus and assigning each document to one or more topics. This process is usually done through unsupervised machine learning algorithms and used in applications such as text mining. The general approach taken by topic models is to analyse either words or word meanings (word embeddings), grouping together the documents which contains similar keywords or which convey similar information.

3.1 Text Preprocessing

A machine learning's performance greatly depends on the quality of the provided dataset, in our case, a text corpus. In order to greatly boost the model's performance, the text must be modified to some degree, according to the model's needs and sensitivity to the input. For example, Bag of Words models greatly benefit from lower-casing all words, stemming and lemmatization, removing stop words, such as *the* and *from*, sometimes removing digits and numbers and so on. Other models are less sensitive and only need minimal cleanup, such as only removing hyperlinks.

Either way, dataset preprocessing is mandatory in all machine learning tasks. While some preprocessing steps can be easily carried out using regular expression substitution, such as removing hyperlinks, others require language knowledge, such as removing the stop words.

Stemming and lemmatization, while not necessary for some models, like transformers, are essential in other models, such as Bag of Words models, like LDA, discussed later. Both techniques have the goal of reducing a word to its base representation, such as removing the *-ed* or *-ing* suffixes from words, since the meaning of the words are the same as the base word, while reducing the vocabulary size. Stemming refers to hard-coded rules of text processing, such as removing the aforementioned suffixes, while lemmatization refers to using language knowledge to correctly reduce each word to its base form.

3.2 Bag of Words

What the concept refers to is the process of analysing words with no regard for the context they come from. Bag of Words algorithms treat text documents as a set of words (while retaining word frequency), not as a sequence of words. This technique makes document analysis much easier and may lead to reasonable results, however the fact that the surrounding words are ignored may drastically affect a model's performance. For example, the words *carbon* and *emissions* may be used individually in different contexts and have different meanings as to when they are used together to form *carbon emissions*, which may be used in a text about pollution. In a BOW approach, there is no clear way to tell, which is why workarounds were created.

One simple solution is using n-grams, which essentially imply using n adjacent words as a token. This increases the vocabulary size, but for our purposes, the main idea is that two or three words can convey much more information than a single word.

3.3 Transformers

Given the fact that transformers offer state of the art performance on popular Natural Language Processing tasks [12], some important questions are how do they work and how can they be used to solve our problem.

In essence, transformers harness the attention mechanism, which quantifies how important a token is in the input sequence [13]. Given a learned word embedding, it is split into a set of query, key and value vectors through learnt parameters. The meanings of these vectors are: "What question does this token ask?", "What question does this token answer?" and "What is the answer to a question this token can answer?". For our use case, only self-attention is relevant, where each token asks each token in the input sequence, including itself, the learnt question.

Mathematically, this translates to performing a dot product between one token's question vector and another token's key vector, thus quantifying how capable each token is to answer the question. The dot products are then passed through a softmax function to form a distribution, which outputs a weight for each token, used to scale the value vector for each token. The scaled value vectors are summed over all asked tokens and finally returned as the answer to the question. There are several implementation details being omitted, such as multi-attention heads and scaling the dot product, but the key take-away is the same, namely that the transformer introduces a new layer in its architecture with the following mathematical formula:

$$Y(X) = \text{softmax}((XW_Q)(XW_K)^T)XW_V$$

The final result of a transformer is an embedding for each inputted token and usually also a sentence embedding, which encapsulates the meaning of the inputted sentence. This is extremely helpful, since we can now train a classification model which takes as input the sentence embedding, thus skipping the actual text processing. The idea is to have a pretrained transformer on a large corpus, which has a deep understanding of words and the meaning of text, and apply that knowledge on a concrete problem, such as topic modelling. The key property of a transformer's output is that sentences with closely related meanings or belonging to similar topics will have their associated sentence embeddings very similar.

3.4 Available Transformers

Most machine learning solutions which incorporate a transformer use an already pretrained transformer, since training one from scratch uses significant hardware resources. While there are several available choices, the most popular is BERT (Bidirectional Encoder Representations from Transformers), which in 2018 achieved new state of the art performance on several NLP tasks [4]. BERT has a standard transformer architecture, as presented in Attention is all you need, but uses 2 pretraining objectives: Masked Language Modelling and Next Sentence Prediction. Using MLM means that some tokens in the input sequence are hidden (about 15%) and the model is asked to predict what the token is. In NSP, the model is presented two sentences and is asked to predict whether the second sentence comes after the first one, or it comes from a different document.

ROBERTA (Robustly Optimized BERT Approach) improves BERT by improving the training objectives and increasing the training dataset size [8]. When training BERT, the authors used static masking when generating masked training examples, meaning that each sentence was duplicated 10 times and for each duplicate, a different subset of tokens was masked. The effect of this masking is that when training for several epochs, the model will keep seeing the same masks. ROBERTA uses dynamic masking, where a mask is dynamically generated right before passing it through the transformer. Another optimisation regards NSP, which was shown to hurt performance. One hypothesis for this is the inability to learn long-range dependencies, since only two sentences were inputted. The optimisation comes from filling the entire input with tokens from the corpus. Lastly, it was shown that larger batch sizes increase performance both in terms of accuracy and training speed. As a result, ROBERTA achieves new state of the art in 2019.

A transformer which outperforms ROBERTA is MPNet [12]. This yet again optimizes the training objective and addresses the main disadvantage of MLM, namely the fact that the hidden tokens are predicted independently of each other.

While the performance of the available pretrained transformers increases, so does their size, required storage and hardware requirements. As as result, distillation is a common technique to keep the model's accuracy high, while drastically lowering its size, which improves memory

usage and lowers run time. This technique implies taking an already trained transformer, called the teacher, and creating a smaller transformer, called the student. This has been explored by various models, such as DistilBERT [11], MiniLM [15] and MiniLM v2 [14], which take as teacher model BERT or ROBERTA.

3.5 Principal Component Analysis

PCA is a dimension reduction algorithm, which reduces the number of dimensions of the input dataset down to a chosen number of dimensions. This has the effect of stripping away unnecessary information and keeping only the essentials. Dimension reduction also has the side effect of reducing the memory footprint, since the original data can be discarded after the algorithm is applied, which results in the same number of examples, but with lower dimensionality.

In short, PCA computes the covariance matrix of the dataset ($X^T X$), which indicates how the different components of each feature vector interact. Next, the eigenvectors and eigenvalues for the covariance matrix are computed. The eigenvalues indicate how much information is encoded into its associated eigenvector. If the features are closely correlated, we should see a couple of high eigenvalues, which encode the majority of the information, and many more eigenvalues close to zero, indicating noise.

In order to compute new features, we compute the original dataset's features in terms of the eigenvectors, thus changing the coordinate system's basis. We can compute back the original dataset by multiplying the new features by the eigenvector matrix. The dimension reduction part comes from the observation that removing components whose associated eigenvalue is small leads to little loss of information. This means that we can simply drop some of the new features, which can be reconstructed to an approximation of the original dataset. As a result, highly correlated data will suffer little data loss while greatly reducing the number of components.

3.6 K-Means Clustering

This is a well-known clustering algorithm, which takes as input a set of points and clusters them based on euclidean distance into K clusters [16]. This is done by computing K cluster centroids, each point being assigned to the cluster associated with the closest centroid. The process of computing the clusters is split into two phases: cluster assignment and centroid relocation. Initially, there are K centroids chosen at random in the feature space. Afterwards, until convergence, assign each data point to one of the clusters. Next, update the clusters' locations to the mean of all points assigned to it. This process is repeated until the centroids converge. While the algorithm is well-studied and performs reasonably well, the mean is not a good statistic. As a result, to achieve better accuracy, it is usually better to use a clustering

		Predicted Value		
		n	**p**	**total**
Actual value	**n′**	True Negative	False Positive	N′
	p′	False Negative	True Positive	P′
	total	N	P	

Table 1: Confusion Matrix

algorithm more well-suited for the problem it is applied to.

3.7 Model Evaluation Metrics

There are several metrics which can be used to evaluate a model's performance. This process is more difficult when using unsupervised learning, since there may not be a clear indicator of the model's performance. There are also metrics which make it difficult to evaluate models trained on imbalanced datasets. We will look over some of those, examining their advantages and disadvantages:

Accuracy is usually defined in a supervised learning problem as the percentage of correctly predicted examples in the test set. The advantage of this metric is that it is simple to compute and understand, however its main disadvantage is that it may not evaluate a model tested on an imbalanced dataset well.

Usually the confusion matrix is computed and, based on it, the model is evaluated. Since it is more convenient to have a single-numbered evaluation metric, as opposed to having four different numbers, there are several metrics created, which solve this issue. Precision and Recall are among them. They optimize the false positive and false negative rates, based on the number of true positives. Since having two numbers to compare is still less than ideal, the F1 Score was introduced, which is the harmonic mean of Precision and Recall. The F1 Score is very widespread and usually a good metric, since it is better than accuracy when it comes to imbalanced datasets.

$$Precision = \frac{TP}{TP + FP}$$
$$Recall = \frac{TP}{TP + FN}$$
$$F1 = \frac{2(Precision \cdot Recall)}{Precision + Recall}$$

The main disadvantage of the F1 Score is the fact that it ignores the true negatives from the confusion matrix. This may not be an issue with some datasets, but with others it can cause optimization issues. This is why the Matthews Correlation Coefficient (MCC or Phi Coefficient), which considers all elements of the confusion matrix, is sometimes preferred. This score has shown better results in practice [3].

$$\phi = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TN + FN)(TN + FP)(TP + FN)(TP + FP)}}$$

The Receiver Operating Characteristic Curve (ROC curve) is a tool designed to illustrate the false positive rate of a model in terms of the true positive rate of a model, which aids in picking a threshold for systems trained in classification tasks and output a numerical score value for each prediction. This shows the trade-off employed by the model, which must compromise at a certain false positive rate in exchange for another true positive rate. Numerically, the ROC curve can be evaluated using the area under the curve (AUC), which in ideal models is 1 and in models which chose the prediction at random in binary classification is 0.5. The higher the AUC is, the better the model performs.

Finally, since our model can be evaluated as a topic model, is can be evaluated using topic coherence. This metric presents several variations, which have been previously analysed [10], but among them, a good metric to use is Normalized Pointwise Mutual Information (NPMI) coherence. What coherence in general measures is the quality of generated topic keywords. Measuring this result was initially done using human evaluation, where a list of topic keywords, along with a random intruding word, was presented to a human. If the human can identify the intruding word, the topic keywords are coherent, and thus, the models outputs high-quality topic keywords. Otherwise, the topic keywords are not coherent and this indicates poor performance.

While this process is accurate, it is also costly, since it requires manual human evaluation. This is why topic coherency was created. It measures a metric similar to the process described earlier and correlates with the human evaluation, which means it may be able to replace the manual process. This process has been criticized since it may not apply to modern topic models, because of the change in their architecture [7].

4 RELATED WORK

4.1 Latent Dirichlet Allocation

The traditional go-to algorithm when it comes to topic modelling is LDA (Latent Dirichlet Allocation). This algorithm has been recognized as the standard approach when it comes to grouping text documents by their determined topics in an unsupervised manner. Several other approaches have been created, which improve upon LDA, but as-is, the algorithm achieves reasonable performance in most use-cases.

LDA is a generative model. The basic principle is creating a model based on the correct topic allocation. After the model is defined, its parameters are optimized, which ultimately serve as the output of the model. This is the case because the model is defined in such a way that topics are required for document generation.

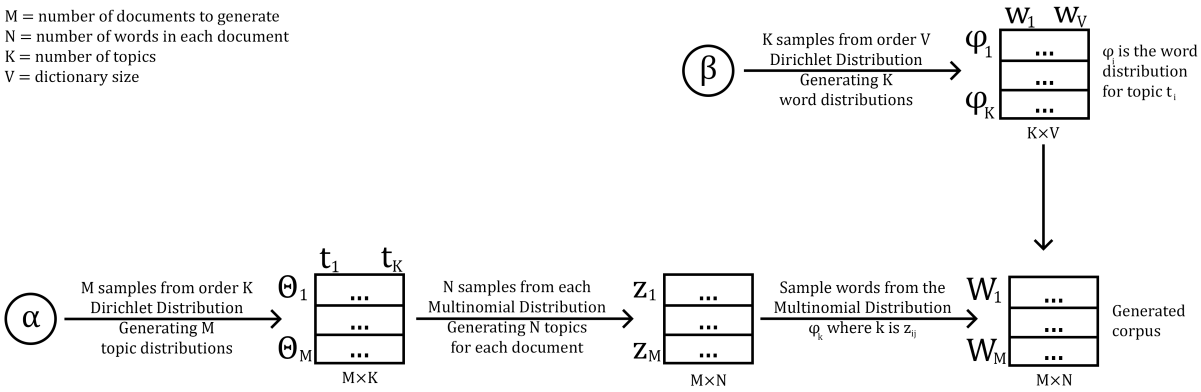


Figure 1: LDA Corpus Generator Architecture

The LDA model is trained by optimizing the two variables α and β. α represents the Dirichlet distribution from which topic distributions are chosen. When generating a corpus, this distribution is sampled M times, where M is the number of documents that we want to generate in the corpus. Since we are sampling from a K-order Dirichlet distribution, the result is an $M \times K$, where K is the number of topics the model uses. A Dirichlet distribution was used here because of its properties to chose vectors that have one dominant component. This matches the intuition that generally a document mostly belongs to a single topic. This means that the resulted matrix θ is mostly sparse, or, intuitively, we chose a single topic for each document.

Once the topic distributions are chosen, each distribution is sampled N times, where N is the number of words in each document. In practice, separate document lengths N_i chosen from Poisson distributions are used, but the implementation does not greatly differ. The

$M \cdot N$ samples result in an $M \times N$ matrix Z of topics, where each value represents the topic associated with each word in the final corpus.

Independently, we can compute the word distributions for each topic, which will be used to sample the words of the generated corpus. These distributions are vectors sampled from the V-order Dirichlet distribution β, where V is the size of the vocabulary. Again, a Dirichlet distribution is used here for its properties to produce sparse results. In this case, each topic will have associated only a few words, so we can intuitively think that this selects the top few words for each topic. The word distributions are defined as ϕ

After both Z and ϕ are computed, we can iterate over all the elements of Z, z_{ij} and sample the multinomial distribution $\phi_{z_{ij}}$ for the word w_{ij}. Once the entire matrix Z is sampled, the resulted matrix W represents the generated corpus. A model has a higher accuracy if the probability of generating a corpus equal to the inputted corpus is as high as possible. Analytically, the function that we want to optimize is:

$$P(W, Z, \theta, \phi; \alpha, \beta) = \prod_{i=1}^{K} P(\phi_i; \beta) \prod_{j=1}^{M} P(\theta_i; \alpha) \prod_{t=1}^{N} P(Z_{jt}|\theta_j) P(W_{jt}|\phi_{Z_{jt}})$$

There are several ways to optimize this function. The original paper used a Monte Carlo simulation and Gibbs sampling was later suggested. The method used by this work follows the implementation described in *Online Learning for Latent Dirichlet Allocation*, which allows for online learning the model.

The main shortcoming of this topic model is the fact that there are no pretrained models already available, so in order to successfully differentiate between common words and technical terms, a large and diverse dataset is needed. Another shortcoming is the fact that the inputted text must be heavily preprocessed:

- Characters must be lower-cased
- Stop words must be removed (the, a etc.) and sometimes common words
- Words must be stemmed and lemmatized

One more disadvantage is the fact that LDA uses a Bag of Words approach. This means that the analysed words lose their context. This also means that multilingual LDA is very difficult to achieve, since the tokens that identify the same word, with the same meaning, but in different languages, are different.

4.2 Topic Modeling in Embedding Spaces

This approach attempts to modify the traditional LDA model by incorporating word embeddings into the model.

Word embeddings are essentially lower-dimension representations of words, with the property that words closely related are also close in the embeddings space. These are usually obtained through unsupervised or self-supervised learning, algorithms which analyse large corpora of text and determine the best relations words should have. They are the predecessors of modern transformers, which generate different embeddings based on the context.

While words can have embeddings, it can be useful for sentences or documents to have embeddings as well, to summarize that the text is about, which is a widely explored idea. This approach assigns each topic an embedding, thus exploiting the property of embeddings of being close for words with similar meanings. This way, a topic embedding will be close to its keywords.

As a result, in order to determine the similarity between a word and a topic, this approach uses the dot product. This maximizes the similarity score if the topic is closely related to the analysed word embedding and also has nice mathematical properties, mainly the fact that doing multiple dot products in parallel with the same one vector is equivalent to matrix multiplication.

To formalize, each word in our vocabulary gets assigned a word embedding $\rho_v \in \mathbb{R}^L$, either pretrained or trained with the topic model. If we assume V words in our vocabulary, the matrix of word embeddings is $\rho_v \in \mathbb{R}^{L \times V}$. We mentioned that each topic gets assigned an embedding α_k for the k^{th} topic. If we compute $\rho_v^T \cdot \alpha_k$, we get the similarity score, and similarly, $\rho^T \cdot \alpha_k$ computes the similarity score for each word in the dictionary for topic k. From here, we can compute a word distribution by applying a softmax function over the resulting dot products, which plays the role ϕ_k in the previous related work. By applying the same procedure for each topic, we get the matrix ϕ, from where the algorithm remains unchanged. As a result, the Dirichlet distribution β is replaced with the topic embeddings α.

There are some other modifications to LDA, such as the shift from the Dirichlet distribution α to a logistic-normal distribution in order to simplify the training process. Also, the training algorithm also needs to be adapted to accommodate for these changes.

The main advantage of this approach is the ability to use pretrained word embeddings, thus making use of transfer learning. As a consequence, it is possible to train on a smaller dataset, but with word embeddings trained on a much larger corpus and on a much larger vocabulary, which benefits from the fact that words in the training dataset will get similarly treated to synonyms present only in the larger pretraining dataset. This somewhat addresses the BOW limitations in LDA through the word embeddings, but the context is still lost.

When analysing the results, this model comes on top of LDA and other topic modelling models. validating the improvement and importance of word embeddings in solving NLP tasks.

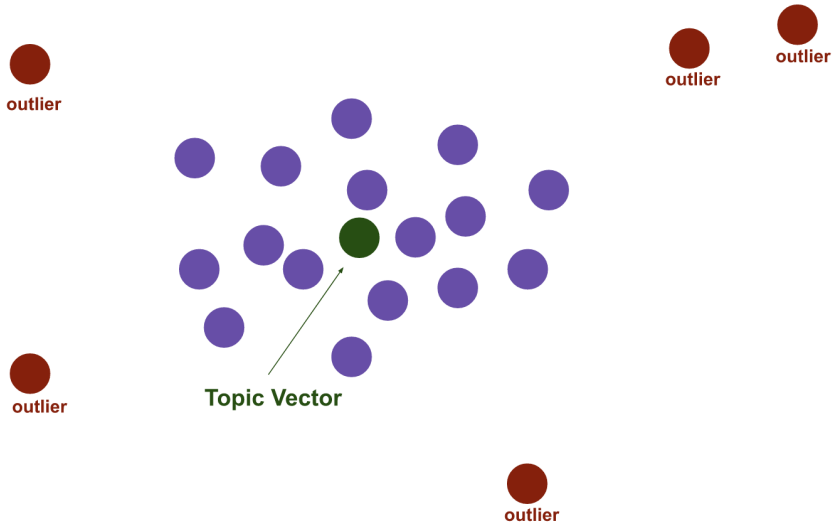


Figure 2: Top2Vec topic vector assignment. Red and purple dots represent document embeddings [2]

4.3 Top2Vec

One approach which takes advantage of pretrained transformers is Top2Vec [2]. This approach tries to mimic traditional topic modelling algorithms, while applying modern machine learning algorithms to the problem. The algorithm begins by passing each document through a document-to-vector model, which outputs the embeddings associated with that document. The intuition behind this is that document embeddings from similar topics are similar themselves. The model which converts the document to vector form, or generates the document embeddings, is usually a transformer, but it also allows any other model with similar properties, such as Doc2Vec. Ultimately, a clustering algorithm will be applied to determine which topic a document belongs to.

One common issue with clustering algorithms is that they do not work well for highly dimensional feature vectors, so for this reason, a dimension reduction algorithm is used. In this case, the algorithm is UMAP (Uniform Manifold Approximation and Projection). This algorithm tries to approximate the manifold on which the data points are located in order reduce their dimensionality. This was preferred as it offered better accuracy than other algorithms, such as PCA, which is more susceptible to outliers.

After the embeddings are shrunk down, HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) is used to assign them into clusters. The choice for this algorithm is its ability to filter outliers and not force each document to be included in a cluster. As a result, the formed clusters offer stronger confidence in the model's prediction, preferring not to make a prediction about the outliers. In order to make a prediction, the model looks at which cluster each document was assigned to, each cluster signifying one topic.

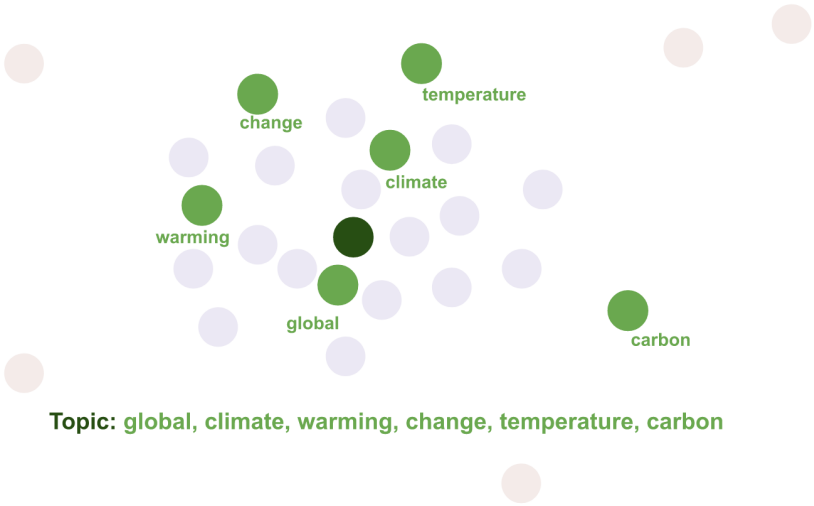


Figure 3: Top2Vec top words selection [2]

To extract meaning out of these arbitrary points in embedding space, the closest k word embeddings for the words in the corpus are selected to be the top topic words. This gives the ability to manually validate that the formed topics are correct.

The disadvantage to this method is that most words, by themselves, do not convey the same information as when they are surrounded by context. For example, in the figure the author illustrates, words such as *global*, *warming* and *climate* will probably be very far apart, since, by themselves, they have different meanings and are only distantly related, when compared to, for example, *temperature* and *warming*.

There are several advantages to this approach compared to the previously presented works, mainly the fact that transformers can be more seamlessly incorporated. They can be used to analyse documents as a whole, which captures their entire meaning, not just individual words. Another advantage is the fact that the number of topics is automatically determined by HDBSCAN, thus removing a hyperparameter. Also, since the data passes through a pretrained transformer, less text processing is required, because the transformer can capture the meaning of documents even in the presence of stop words and performs word stemming when converting the words to tokens.

4.4 BERTopic

Lastly, BERTopic improves on the previous solution by shrinking the number of topics to a desired number. As the clustering algorithm can produce too many clusters, it is usually desired to shrink down the number of clusters. This is done by merging similar topics into a single one in a greedy manner.

Similarity between topics is determined using c-TF-IDF (Class-Based Term Frequency - Inverse

Document Frequency). Traditionally, TF-IDF is used to determine how important a word is in a text corpus. The intuition behind it is that an important word appears several times in the document, but does not appear in the majority of documents. Therefore, the product between the term frequency within a document and the inverse document frequency is usually a good indicator to how important a word is. The modification made to the formula is a generalization, such that each document belonging to a cluster is defined to belong to the same class, and the inverse document frequency now computes the inverse class frequency. In other words, the new formula divides the term frequency in a class, which is the concatenation of documents in the same cluster, by the inverse class frequency, which indicates in how many classes a word is.

Using this new formula, topic distributions can be computed for each word, or inversely, word distributions for each topic. Thus the decision to merge two topics is done by measuring the similarity between their word distributions. The secondary use for the word distributions is that they offer human insight into what topic each cluster represents. This is one of the visualisation tools offered by this model: viewing the top words for each topic which best describe it.

The main advantage of this modification is the fact that we can manually tune the number of generated topics if their number is too great. The main disadvantage of this approach is that merging the clusters is done by a bag of words approach. This diminishes some of the performance given by the transformer, since perhaps c-TF-IDF may disagree with the transformer's output and result in erroneous merges.

5 PROPOSED SOLUTIONS

5.1 General Solution

The general proposed solution follows the following steps:

1. Preprocess the dataset
2. Train the system on the training dataset
3. Compute the feature vector for each publication in the training dataset
4. Compute the user profile for each author using the feature vectors associated with publications the authors wrote
5. Compute the feature vector for each publication in the cross-validation dataset
6. Evaluate the model's performance on the cross-validation dataset
7. Assess the results and if improvements are believed to be possible, go back to step 2.
8. Compute the feature vector for each publication in the test dataset
9. Evaluate the model's performance on the test dataset

The motivation for this general process is the easy integration with existing transformers. While alternatives, such as directly computing the user profile, would have required more complex and custom solutions, this approach relies on the accuracy and properties of the generated feature vectors.

While this architecture was built for a form of author identification, but is more similar to a topic model. The reason for this is the fact that most author identification algorithms focus more on the style of writing for each author, as opposed to the areas of expertise of one

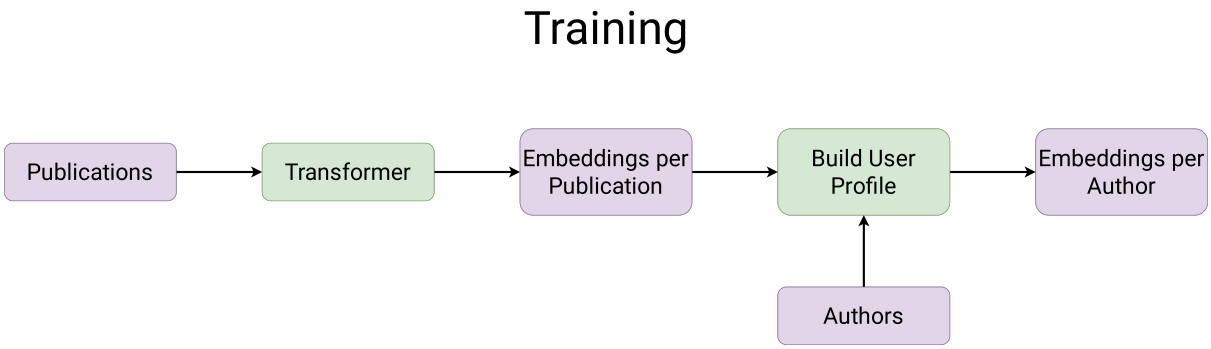


Figure 4: General Solution Training diagram. This describes the initial training step, which generates the topics' feature vectors. The topic model can be validated by the words associated with each topic.

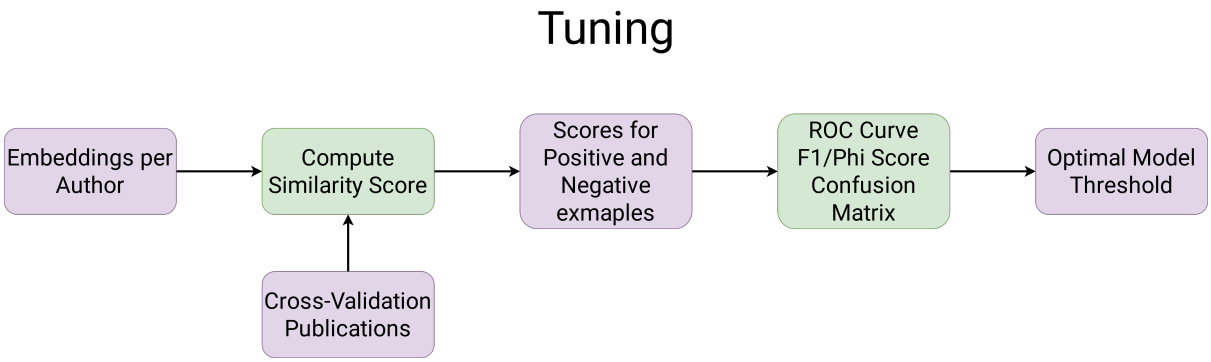


Figure 5: General Solution Tuning diagram. The purpose of this step is to find the optimal threshold, which separates the positive examples from the negative examples.

author. This solution does not look to invalidate an example based on the particularities of one's style of writing, but rather based on the topics of their previous publications.

5.2 User Profiles

When discussing the implementation details, if the transformer computes feature vectors closely related for publications in similar fields, then it is easy to compare the similarity between two publications. However, in our case, we need to compare the similarity between a user profile and a publication. As a result, there are multiple ways to represent the user profile. A simple proposed solution is to compute the mean feature vector of an author's publications. This way of creating the user profile has its disadvantages, since embeddings are not meant to be averaged in such a way, but for our use case it works reasonably well.

In order to validate that the model performs well, we can take advantage of text summarization techniques. Transformers have the nice property of generating document embeddings similar to the embeddings of words which summarize the document. Knowing this, we can extract all words in a publication, compute their embeddings, and compare them with the document embeddings. This technique achieves great results in extracting a document's keywords. When single words do not convey as much information as we would like, we can use n-grams from the text document, giving as more freedom in representing the text's meaning.

Extending this algorithm comes naturally to any embedding. We can use the mean embedding of an author's publications to extract the keywords which best characterize that author. Manual inspection algorithms, such as this, are extremely important in debugging and verifying the model's performance.

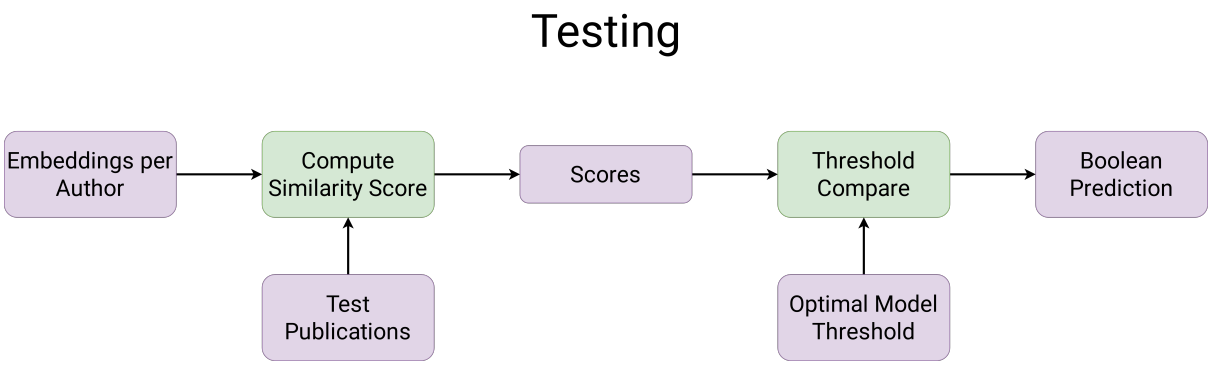


Figure 6: General Solution Testing diagram. Yields the expected performance on new data.

5.3 Fine Tuning

In order to train the system, the publications dataset was split in the training set (60% of the data), the cross-validation set (20% of the data) and the test set (20% of the data). After training on the training set, the performance was analysed on the cross-validation set. Because the scores outputted by the system were arbitrary, a threshold value had to be chosen to separate the negative and positive predictions.

To do this, the ROC curve was computed, which plots the true and false positive rates based on the set threshold. The curve was sampled uniformly and each point was considered as a set threshold. This was done to reduce the number of candidate values. Afterwards, for each sample, the confusion matrix was computed and scored. The threshold with the highest score was chosen.

Scoring the confusion matrix proved difficult because of the imbalanced dataset. The ratio between positive and negative examples is about 7:1. Ultimately, the Matthews Correlation Coefficient (MCC) or Phi Coefficient was chosen. This proved useful because it considers the true negatives as well, whereas the more popular F1 score ignores it. This was also more useful than the FBeta score, since using it introduces another hyperparameter.

TODO graph with phi score and f1 score

5.4 Variations

One possible variation from the presented architecture is the possibility to replace the transformer with a topic model. We can simply compute the feature vectors as arrays of topic probabilities. The workflow in this case remains almost the same. Because of this, we can think of a topic model as a document-to-embedding model. We do not need to generate word embeddings any more, since the topic models already come with validation and visualisation tools.

Another variation of the current architecture is turning it into a topic model. There are

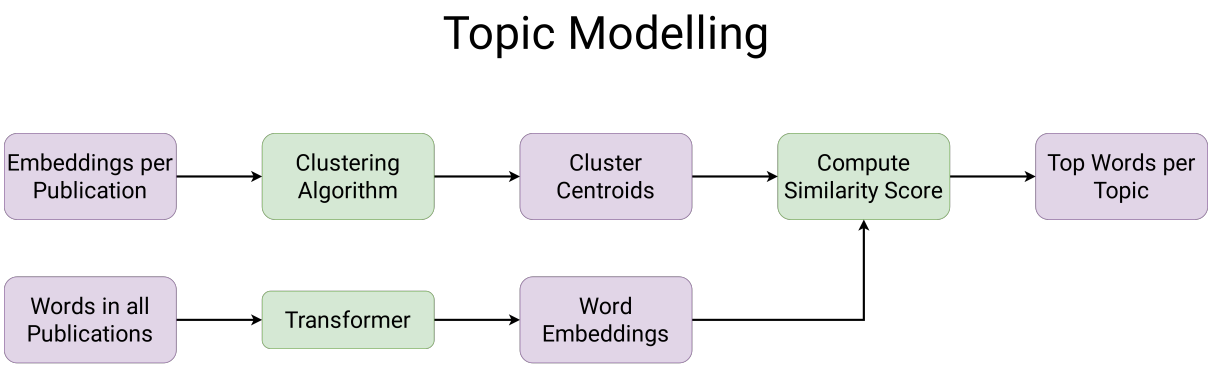


Figure 7: General Solution Topic Modelling diagram. Modifies the current architecture to generate topics based on the provided text corpus.

embedding vectors generated for each publication in the dataset, and intuitively, publications of similar topics get placed close to each other in the embedding space. Knowing this, we can apply a clustering algorithm, such as K-Means, in order to determine the various topics. Extending the visualisation algorithm described earlier when talking about user profiles, we can take the computed cluster centroids and find the related words related. This gives a similar output to that of a traditional topic model.

Finally, this approach allows for online learning the system, as long as the topic modelling algorithm supports it as well. Transformers are already suited for this use case, since we do not require transformer fine tuning to be done. The general strategy would be to update the topic modelling algorithm and periodically update the user profiles as well. It is possible to update the mean of a user profile by removing the old feature vector and adding the new feature vector, without iterating over all the authored publications. As long as the topic modelling algorithm does not output drastically different data, this method allows for a rolling update over the entire dataset. The same applies for a transformer. There are some issues which appear when doing dimension reduction on the embeddings, but this detail will be discussed later.

6 IMPLEMENTATION DETAILS

6.1 User Profile and Similarity Metrics

There are several ways to compute the user profile. The chosen approach is tightly tied with the similarity metrics chosen to compare the profile with the publications. Three approaches were tested:

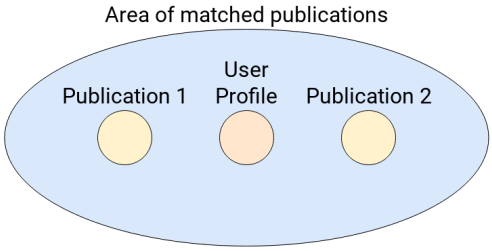
- Computing the average feature vector of the publications
- Computing the average feature vector and standard deviation of the publications
- Keep the list of the authored publications' feature vectors

The first approach is the simplest. The intuition is that the average feature vector encapsulates the average publication information. In other words, given enough publications, the average should converge to the author's area of expertise. Another advantage is the fact that there are several similarity metrics between two vectors, such as cosine similarity and euclidean distance, which are thoroughly tested and known to achieve good results. The disadvantage is that diversity is lost through this approach, as the set threshold may not include the original publications, as exemplified in Figure 8(b)

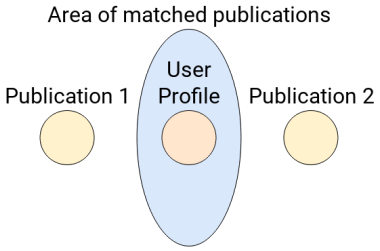
The second approach attempts to fix the diversity problem mentioned earlier by also computing the standard deviation for the feature vectors. One issue that appears is the fact that this requires at least two publications per author, so for authors with one publication, there needs to be a placeholder value for the standard deviation. The similarity metrics in this case have changed and we chose to use the normal probability density function multiplied for all publications, which treats the authored publications as independent samples. In practice, this achieves results worse than simply using the mean.

The last considered approach is keeping the list of authored publications, computing the similarity between each of them and a new publication using similarity metrics mentioned in the first approach, then returning the maximum score. The problem this tries to solve is dynamic expansion of the area of matched publications. If an author publishes several less related publications, writing another one in a similar field will not get triggered as an anomaly, regardless of the number of publications. The disadvantage would be an increased false positives rate, since the area of accepted publications is expanded with the number of publications. In practice, this is slightly worse than the first approach.

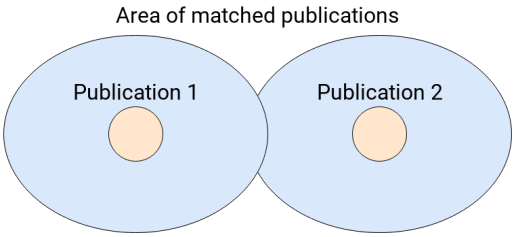
One theoretical approach, which was not tested, is computing a custom shape, such as a convex hull, based on the authored publications. This way, computing the distance to this



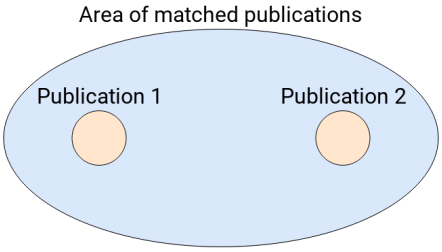
(a) User Profile computed as the mean of authored publications. The threshold is sufficiently large and correctly classifies publications in the training set.



(b) User Profile computed as the mean of authored publications. The threshold is too small and publications in the training set are misclassified.



(c) User Profile computed as the list of authored publications. A publication is predicted positive if it is similar enough to only one authored publication.



(d) User Profile computed as custom shape depending on all authored publications. The score is the distance to the custom shape.

Figure 8: Various ways to compute the user profile

shape would count as the score. This would be a more elegant variation of the third approach, but it remains as a theoretical idea.

Overall, the best approach was the first one, while using cosine similarity. An interesting observation is the fact that the false negative rate decreases for authors with more publications, which may enforce the idea that authors have well-defined area of expertise to which the user profile converges.

6.2 Dimension Reduction

Some related woks applied dimension reduction algorithms, particularly UMAP [6] [2], before applying a clustering algorithm. This was attempted as well in this implementation. The dimension reduction algorithms considered are PCA and UMAP. The number of components and other parameters related to UMAP were considered using a randomized grid search, which maximized the model's performance.

PCA achieved the best results, but not by a large margin, the overall improvement being of 1% in accuracy. While it may seem insignificant, this indicates that it is possible to extract more performance out of the model with other similarity formulas. The most promising direction would be training a neural network which takes as input a document embeddings and a user profile, and outputs whether the publications belongs to the user.

During testing, PCA's number of components parameter did not greatly affect the performance, as long as it was a reasonably small number (i.e. 5 to 20). UMAP however seemed to have varying performance based on the parameters. Since UMAP has many more parameters than PCA, tuning it took much longer, but always seemed to perform worse than not using a dimension reduction algorithm at all. For this reason, while both are supported, as well as not using any dimension reduction at all, the decision was to keep using PCA.

While it improves performance, using a dimension reduction algorithm makes implementing an online learning model more difficult, since it requires this algorithm to allow online learning as well. There are solutions for this problem, such as using IncrementalPCA.

6.3 Synthetic Negative Examples

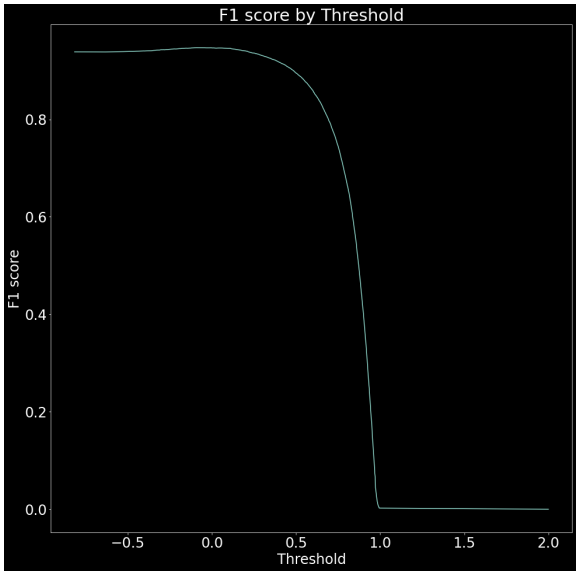
A technique whose purpose is to prevent overfitting is creating a synthetic dataset for negative examples. There is a possibility to have negative examples in the original dataset closely related to the positive examples. This would throw off the fine tuning phase, since it would result in determining an optimal threshold too high, while our target is to train a model which rejects publications more distant to an author's area of expertise. To overcome this, we can use a synthetic dataset for the negative examples.

Generating this is fairly easy, since we can pick a random publication and a random user and, most probably, the user won't be the publication's author. This alloes us to generate many negative examples, which can match the positive examples in the dataset. This technique is not necessary for the positive examples, since it actually benefits the model's performance if they are closer to the user's area of expertise, and there are sufficient positive examples.

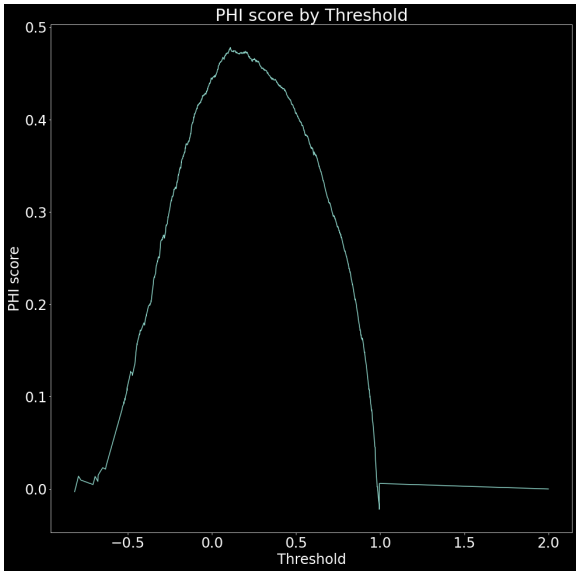
After training the model with both the real and synthetic negative examples, we can observe that in all cases, the performance is worse on the synthetic dataset, which is expected. Also, the optimal threshold is lower, which, again, is a desired effect. This technique was used to validate if a model overfits the training data, and in some cases it signaled this issue. We can observe that training the model with the optimally determined parameters on both datasets yields similar performance (2% accuracy difference). which leads us to believe that those optimal parameters are not overfitting the model.

6.4 F1 Score vs Phi Coefficient

An issue arose when evaluation the model: which evaluation metric to use? This was essential for choosing a good threshold for the model, since the score outputted is an arbitrary value, which only correlates with the prediction we want the model to make. The selection process was as follows: choose a number of candidate thresholds, which are sampled uniformly from the ROC curve; then for each threshold, compute the model's performance using a numerical



(a) F1 Score computed for each considered threshold



(b) Phi Coefficient computed for each considered threshold

score and keep the threshold which yields the highest score.

Naturally, the more popular F1 score was tried first. The performance was not bad when using this metric, however its behaviour would sometimes be inexplicable. It would sometimes generate thresholds which are extremely low, orders of magnitude sometimes, compared to the expected values. The root cause for this was the imbalance of the dataset. Applying the F1 score on the synthetic dataset worked much better, since we can generate as many negative examples as we have positive examples, but for evaluating on the original cross-validation set, we need another metric.

This is how the Phi Coefficient was considered. This also considered the true negatives, which the F1 score did not. Overall, tuning the model using this metric yielded much better results. Plotting the score over the candidate thresholds should intuitively peak somewhere in the middle, where the balance of false positives and false negatives is in balance, but that did not happen for the F1 score, and it did for the Phi Coefficient. This is the reason why this metric was used to tune the model.

7 EVALUATION

There are two main methods for evaluating a model: quantitative and qualitative evaluation. The quantitative evaluation is done through the F1 score, Phi score, AUC, accuracy and topic coherency. For qualitative evaluation, we inspect the keywords generated based on a publication, user profile or topic.

While the proposed solution is not a topic model, we will evaluate its performance as one, since this allows us to compare its performance with other similar models. One observation is the fact that quantitatively evaluating a topic model is problematic, as stated in a recent paper, because a topic mode's performance is best determined by a human, not a score. This is why quantitative evaluation is preferred when evaluating the model's performance on the CRESCDI dataset, but qualitative evaluation is better when evaluating it as a topic model.

7.1 CRESCDI Dataset

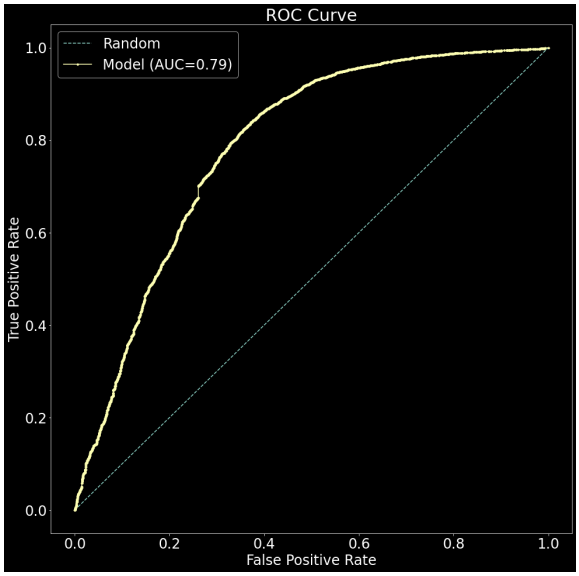
This is the dataset for which the model was optimized. For a fair comparison, both the proposed solution was evaluated on this dataset, but also LDA. The evaluation metrics used are NPMI Coherency, AUC, Phi Coefficient, F1 score and accuracy. Since the dataset contains both positive and negative user-publication associations, meaning we have both examples of users who know wrote specific publications and examples of users who did not write specific publications, we can compute the accuracy as the percentage of correct predictions of our model.

We can observe that the proposed solution outperforms LDA in all quantitative metrics, except topic coherency. This may be due to the fact that this metric was not optimized, but qualitative evaluation hints towards the model having a good performance.

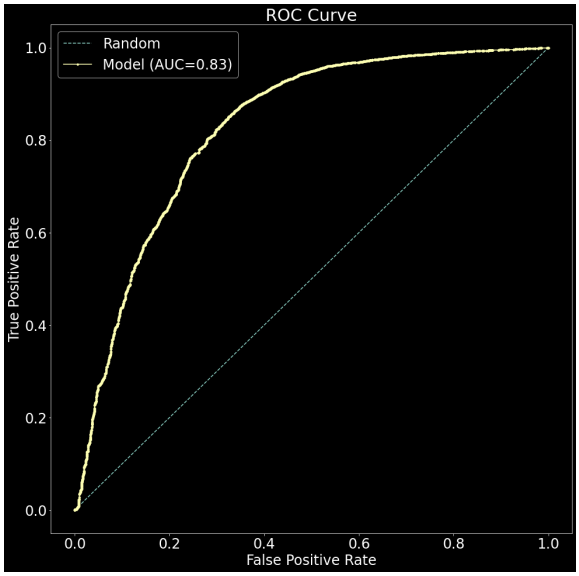
The qualitative evaluation is represented by the topics generated by each model. We can see that they are both coherent, but there are some differences. Since LDA is a token-based

Model	NPMI Coherency	AUC	CV Phi Coefficient	CV F1 Score	CV Accuracy	Test Accuracy
LDA	0.0436	0.7856	0.4140	0.9469	85.18%	83.86%
Proposed Solution	-0.2041	0.8256	0.4780	0.9473	90.26%	**91.12%**

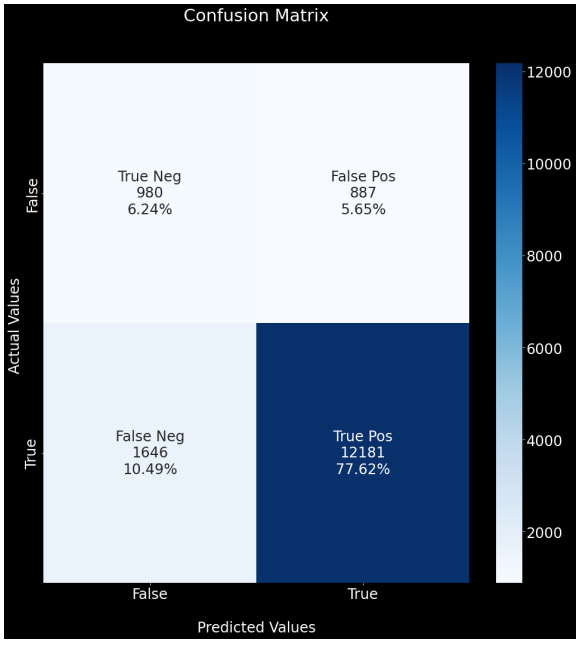
Table 2: Performance comparison between the two solutions.



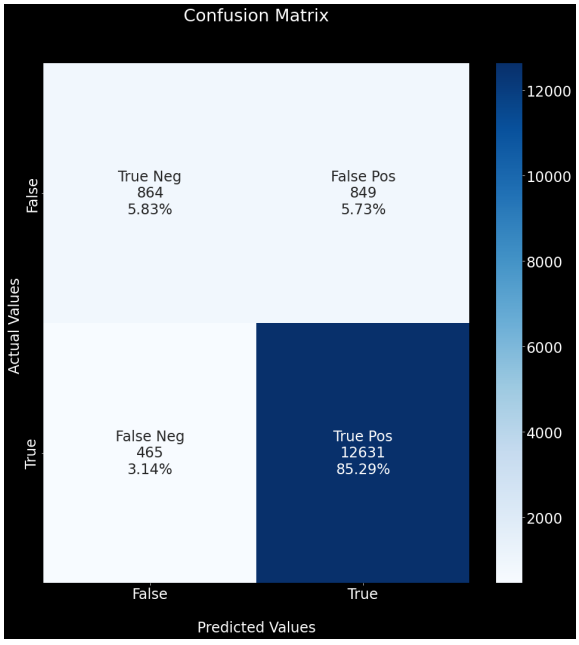
(a) ROC curve obtained by the LDA model on the cross-validation set.



(b) ROC curve obtained by the proposed solution on the cross-validation set.

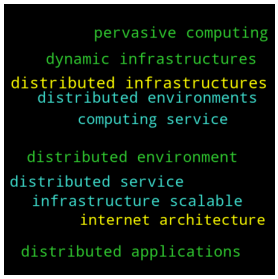


(c) Confusion matrix obtained by LDA on the test set.

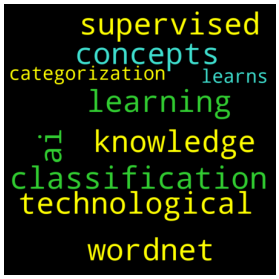


(d) Confusion matrix obtained by the proposed solution on the test set.

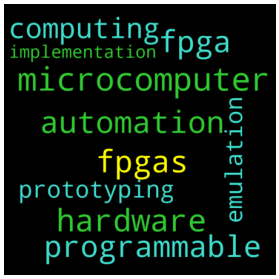
Figure 10: Metrics obtained by the two implemented models



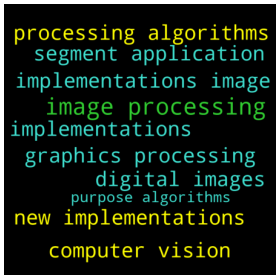
(a) Keywords for Ciprian Dobre



(b) Keywords for Traian Rebedea

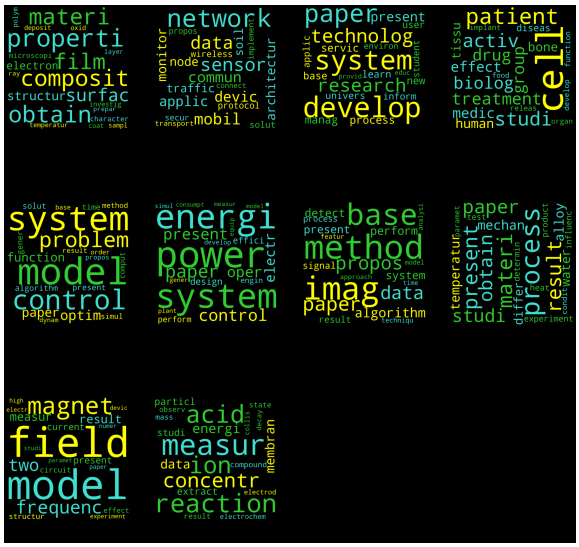


(c) Keywords for Cornel Popescu

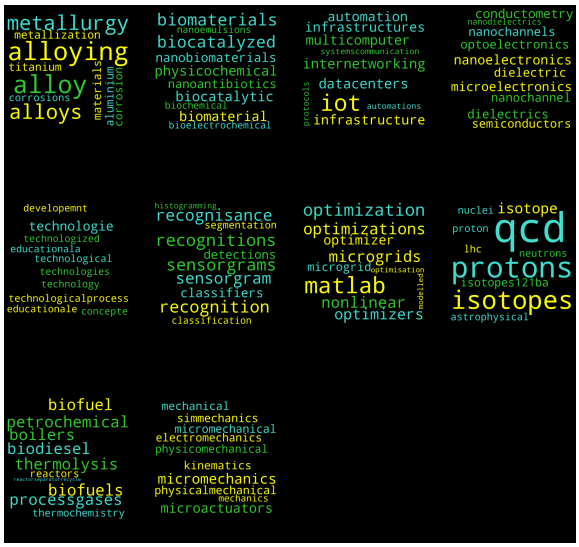


(d) Keywords for Victor Asavei

Figure 11: Top keywords generated for various authors from University Politehnica of Bucharest



(a) Keywords generated by LDA

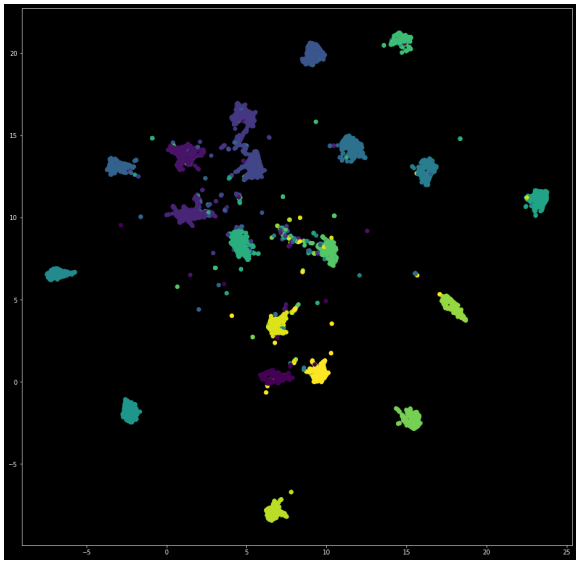


(b) Keywords generated by the proposed solution

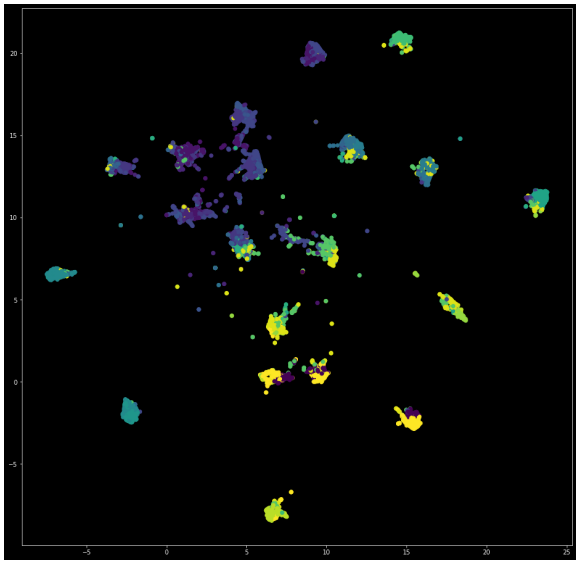
Figure 12: Topic keywords generated by the two models

model, it can not easily generate n-grams for topic keywords. Because of this, the model must convey the topic information through single words. We can see common words misclassified as topic keywords, such as *result* and *two*. Going to our proposed solution, we can observe the same words appearing, which is unusual for a traditional topic model. We can also observe that the topics are slightly different when generating words, compared to n-grams, which are more coherent. This unusual behavior for a topic model may be the cause for the poor topic coherency score, even though the performance is not bad.

Looking at the keywords generated for each author, a feature non-existent when using LDA, we can observe that those match the keywords for their publications. Again, generating n-grams yields more coherent results and the same words get generated.



(a) Correct topic allocation from the training set



(b) Allocation discovered by the proposed solution

Figure 13: 20 Newsgroup document embeddings projected in 2D using UMAP, while knowing the target topics

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

Table 3: Target topics for the 20 Newsgroup dataset [9]

7.2 20 Newsgroup Dataset

This model was also evaluated on a standard dataset, the 20 Newsgroup Dataset, which contains around 18.000 training text examples belonging to 20 topics [1]. As observed on the previous dataset, the qualitative evaluation showed better results than the topic coherency. This pattern holds on this dataset as well, the topic coherency being very low compared to other models. The dataset offers the correct topic allocation to the text examples, however we can't automatically correlate this allocation to the one generated by our model. For this reason, this mapping had to be manually done by observing the topic keywords and matching them with each target topic.

The number of topics is known in advance, but setting the number of topics to 20 leads to merging and splitting of topics. For example, the *auto* and *motorcycle* topics have been merged, while the *mid-east* topic has been split in two. As a result, we will set the number

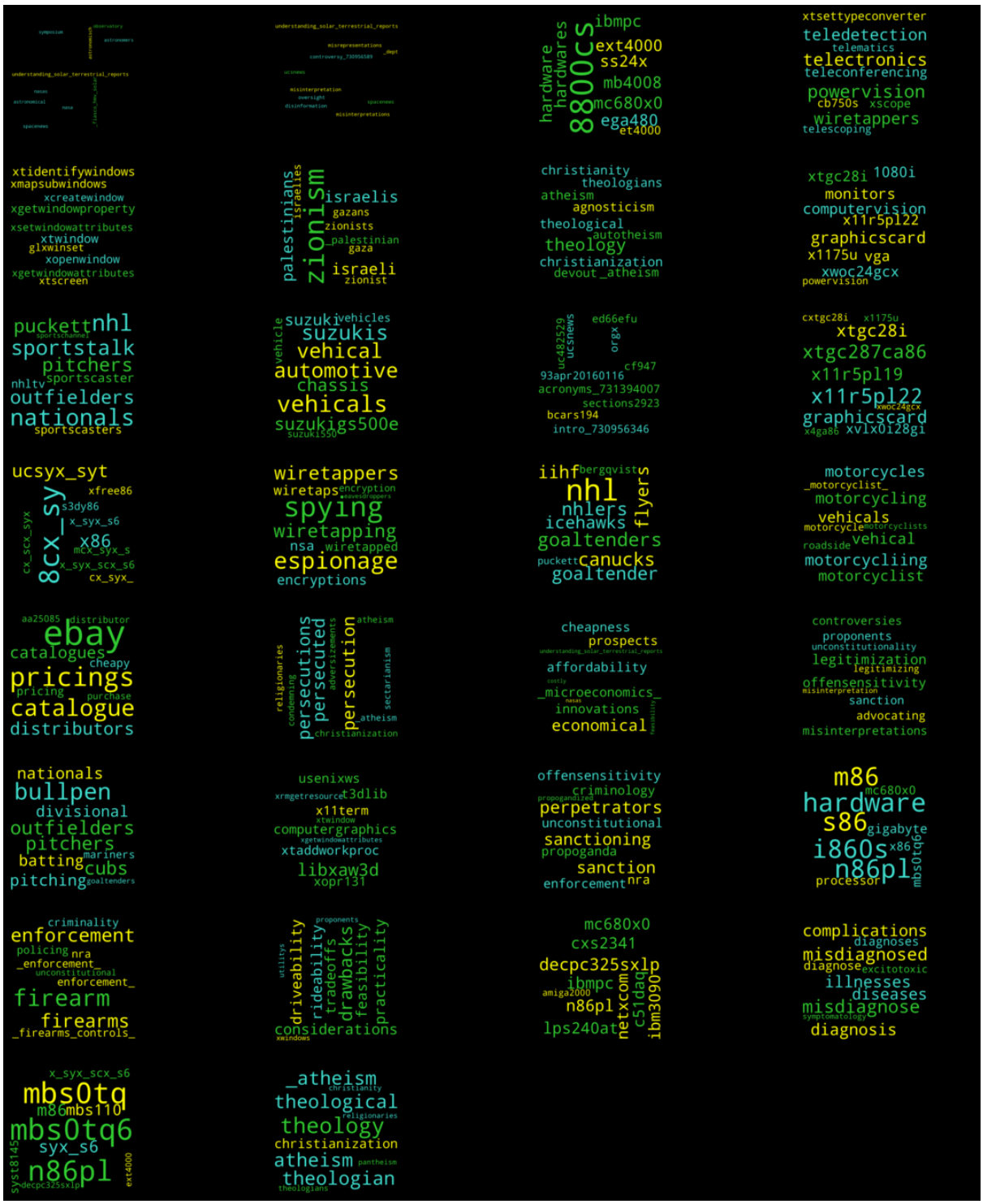


Figure 14: Topic keywords generated by the proposed solution on the 20 Newsgroup dataset

of topics to 30, since this value is used in other works as well. Computing the keywords for each topic, we observe that all target topics are present, but some have been split into several detected topics. This evaluation is done by computing the top words for each topic, which we can compare with other models.

We can visualize the results by applying a dimension reduction algorithm, which reduces the input to two dimensions. We can use UMAP for this, since it can also account for the target label. Looking at the correct and identified topic allocations, we can observe that the model has difficulty correctly identifying some clusters. Since the dimension reduction is done from the document embeddings in both cases, we can assume that this poor performance is due to the clustering algorithm, namely K-Means.

7.3 Resource Consumption

We will analyse the resource consumption both when using a GPU and when only using a CPU. The model was tested on a laptop with the following configuration:

The model development was done on a dataset of about 40000 publications, 13000 positive author-publication associations (i.e. a user is a publication's author) and 1600 negative author-publication associations (i.e. a user isn't a publication's author). The dataset was split into a training set (60% of the publications), cross-validation set (20% of the publications) and a test set (the last 20% of the publications). As a result, the training was done on about 32000 publications. The transformer's batch size was set to 32, since any higher number resulted in possible out-of-memory issues on the GPU.

The benchmark was done on a machine with Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz, 6 cores, 12 threads, 24GB RAM, GP107M [GeForce GTX 1050 Ti Mobile] 4GB, Ubuntu 22.04 LTS, Linux 5.15.0-40-generic.

	CPU	CPU + GPU
Training Time	14min 30s	3min
Prediction Time per batch (32 publications)	1s	0.2s
Memory Usage	1.6GB	2.3GB (+1GB Graphics Memory)

Table 4: Resource consumption

8 CONCLUSIONS

This proposed solution solves the author identification problem applied on the CRESCDI publications dataset. The architecture uses pretrained transformers, which allows the model to benefit from transfer learning. It allows for generating a user profile for each author, as well as topics associated with each topic. For each publication, user profile and topic, it allows for generating keywords, which validate the model's performance, as well as identifies a user's area of expertise and a topic's meaning.

The architecture allows for online learning, but improvements, such as using dimension reduction algorithms, may make the process more difficult, while still entirely possible. Since the transformer used is very lightweight, a graphics card is not required to train or use the model, and the architecture allows for the data to be located in an external database. The accuracy on the test set for the model was about 91%, which satisfies the performance requirements, given the fact that the model does not use supervised learning.

8.1 Future Work

Certain improvements need to be made in order to make the system production-ready, mainly incorporating it in a Docker container and exposing it using a REST API. This would allow the system to be deployed and communicate with the CRESCDI Platform, while running in its isolated environment. The hardware requirements, as stated before, do not pose an issue.

Another improvement is exploring more options of computing the user profile. As previously stated, the mean does not have good statistical properties, so a different algorithm may benefit the performance. Also, the similarity metric used between the user profile and publications may not be optimal, as indicated by the fact that applying dimension reduction resulted in a better performance. These issues may be solved by applying supervised learning techniques, such as creating a model which takes as input a list of the most recent publications of an author and a new publication, outputting the required similarity score.

BIBLIOGRAPHY

[1] 20 newsgroups text dataset,. https://scikit-learn.org/stable/datasets/real_world.html#the-20-newsgroups-text-dataset. Last accessed: 25 June 2022.

[2] Dimo Angelov. Top2vec: Distributed representations of topics, 2020.

[3] Davide Chicco and Giuseppe Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21, 01 2020.

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

[5] Michael Fire and Carlos Guestrin. Over-optimization of academic publishing metrics: Observing goodhart's law in action. *GigaScience*, 8, 06 2019.

[6] Maarten Grootendorst. Bertopic: Neural topic modeling with a class-based tf-idf procedure, 2022.

[7] Alexander Hoyle, Pranav Goel, Denis Peskov, Andrew Hian-Cheong, Jordan Boyd-Graber, and Philip Resnik. Is automated topic model evaluation broken?: The incoherence of coherence, 2021.

[8] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[9] Jason Rennie. 20 newsgroups,. http://qwone.com/~jason/20Newsgroups/. Last accessed: 25 June 2022.

[10] Michael Röder, Andreas Both, and Alexander Hinneburg. Exploring the space of topic coherence measures. *WSDM 2015 - Proceedings of the 8th ACM International Conference on Web Search and Data Mining*, pages 399–408, 02 2015.

[11] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2019.

[12] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding, 2020.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[14] Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. Minilmv2: Multi-head self-attention relation distillation for compressing pretrained transformers, 2020.

[15] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.

[16] Xindong Wu, Vipin Kumar, Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, G. Mclachlan, Shu Kay Angus Ng, Bing Liu, Philip Yu, Zhi-Hua Zhou, Michael Steinbach, David Hand, and Dan Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14, 12 2007.