EasyChair Preprint

№ 4530

Categorization of CORD-19 Articles Using Word Embeddings and Topic Models

José Antonio Espinosa-Melchor and Jerónimo Arenas-García

November 7, 2020

Categorization of CORD-19 articles using word embeddings and topic models

José Antonio Espinosa-Melchor
Universidad Carlos III de Madrid
Madrid, Spain
joseantem@gmail.com

Jerónimo Arenas-García
Universidad Carlos III de Madrid
Madrid, Spain
jarenas@ing.uc3m.es

Abstract

The outbreak of coronavirus disease 19 (COVID-19), the disease caused by the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2), has shaken the world causing a global crisis in a completely unexpected way not seen in years. The rapid spread and its severity have incited scientists all over the world to investigate its causes, symptoms, treatments and effects, resulting in a huge number of publications and articles in just a few months. This overwhelming amount of information complicates access to proper investigations and facilitates the inclusion of non-relevant studies that can delay critical activities. Our goal is to determine the best way to categorize documents, determining which are the ones most relevant to different groups, such as policy-makers or biomedical community, to advance in their investigations, overcoming information overload. We have proposed five classes for a predefined COVID-related corpus (CORD-19), demonstrating that some of the articles included have no connection with the subject, and that the relevance of each paper is highly dependent on the specific area of study. Promising results were obtained making use of a simple model that combines word embeddings, topic modeling, and a Support Vector Classifier.

1 Introduction

The outbreak of coronavirus disease 19 (COVID-19), the disease caused by the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2), has shaken the world causing a global crisis in a way not seen in years and completely unexpected. The rapid spread and its severity have incited scientists all over the world to investigate its causes, symptoms, treatments and effects, resulting in a huge number of publications and articles in just a few months. To support the investigation on COVID-19, Allen Institute for AI and some researching groups have created the Open Research Dataset (CORD-19) available on SemanticScholar, with approximately 200 000 scholarly articles, creating a challenge in Kaggle [1] in which some tasks are asked to be solved by means of Natural Language Processing (NLP). These tasks are a list of initial questions related to what is already known about COVID-19, for example making tables that summarize risk factors or clinical studies.

However, the overwhelming amount of information complicates access to proper investigations and facilitates appearance of non-relevant studies. For the creation of the dataset, papers with keywords such as *COVID* or *MERS-CoV* in their title, abstract or body are included; however this search retrieves a heterogeneous collection of documents. In fact, some completely unrelated work appears, and authors of [1] admit limitations, encouraging people to "curate and publish such datasets".

In the field of NLP, several attempts have been made to improve automatic text classification. Most of them include the representation of texts as bag of words (BoW), a simplified representation composed of tuples (word, number of appearances), or the representation of texts or words as embeddings, which generally works better, but require more computational power. Some of the most popular applications in this regard are word predictions and sentiment analysis used typically in short texts such as tweets or reviews where text is binary classified, either as positive or negative opinion. For this, recent works such as [2, 3] convert labels to vectors, making use of more information than just text. Others [4, 5] combine recurrent neural networks (RNN) with topic models to obtain semantic and syntactic features. A different approach is [6], modeling the entire corpus as a graph. However, these texts are generally short, even just a few sentences, and make general predictions like good-bad or differentiate between sports and history. Recent investigations aim for extreme multi-label text classification, to classify texts with thousands of

labels following a structured order [7–9]. These situations are completely different from our case, where scientific papers in the CORD-19 dataset may contain thousands of words and address subjects as different as sociology and physics, but it is also necessary to distinguish between viral and bacterial infections.

Our goal is to establish the best way to categorize selected documents, overcoming information overload, providing a classification by distinguishing multiple text classes and assessing performance of various classification methods. It is possible to select or filter texts according to their potential application in diverse areas. With this, it is expected to help biomedical researchers advance in their investigations, but also some other specialized groups, such as financial institutions to find groups that are worth funding or policy-makers to evaluate social effects and consequences of certain policies.

With the purpose of creating these tools for helping researchers and other potential users of the CORD-19 dataset, in this paper we propose a taxonomy with 5 different classes depending on the kind of relation to the COVID-19 disease, and manually label approximately 1000 documents so that the problem can be solved using machine learning. Different classification models will be tested, including different kinds of document representation and classification algorithms. Apart from classification performance, we will also discuss the potential benefits of this approach when using topic models and graph-based visualizations.

This paper is structured as follows. Section 2 explains the information found and used in the dataset employed. Section 3 presents the main tools that will be used in this work. The evaluation tasks are introduced in Section 4. Section 5 shows the experimental results and an analysis of them. Finally, conclusions are drawn in section 6.

2 The CORD-19 dataset

As stated in Section 1, the dataset used in this work is CORD-19, in particular, the version of the dataset is 22072020 (22 July 2020). The total number of articles in the entire dataset is 197 412, a significant number of COVID-related articles found in just seven months, and it keeps increasing for new versions.

Dataset information This dataset contains some metadata, including title, publication date, authors, several IDs, DOI and journal. Also, some text is provided for the articles, the abstract is available for 140 894 of them and full body only for 63 023. It does not contain any type of label or keyword to filter by themes. In our experiments, the information kept is: CORD id, title, authors, publication year, abstracts and full body text. Together with paper data, the dataset includes pre-computed 768-dimensional document embeddings. These embeddings are computed using the paper titles and abstracts of the entire corpus; however, to the author's knowledge, the model used to create them is not available and may change with each version of the dataset. This data will be used as a reference to assess other text representations, as it has been obtained using the complete dataset and it is assumed to provide a better representation.

Article classification One of the problems of this dataset is the generic search made to obtain it where, in order not to miss any relevant article, almost any paper containing certain terms is retrieved. This produces a collection of documents with diverse topics, such as medicine, veterinary, economics or even optics. Some of the documents whose main topics are these may actually be relevant, but not all of them, as they can also be separated in many subtopics. For this reason, a set of documents, called gold dataset, has been created by manually labeling[1] the corresponding documents. Five classes 1 to 5 has been defined and is assigned to documents based on the main themes addressed in those documents, creating and employing the following matrix:

Table 1: Class assignment table

Class	Description	Term examples	Number documents
1	Directly related to COVID-19 disease and other coronaviruses	covid, sars	265
2	Other respiratory diseases and effects, virology, genetics	influenza, respiratory disease, virus, vaccine	265
3	Economic effects and policies application effects from COVID-19	project, public health	207
4	Psychological investigation on COVID-19	depression, confinement	62
5	Non-COVID-related articles	cybersecurity, bone fracture	206

The total number of articles that have been manually classified is 1005. The column *term examples* on Table 1 shows some words expected to appear on articles belonging to that class. Note some of the terms are characteristic of one class but might be shared among different ones. Also, some articles classified as belonging to class 5 may deal with other medical conditions with no direct relation to COVID-19; however, they are not included in any other classes as their main themes are, for example, parasites in plants. Finally, the number of documents assigned with class 4 is lower, but they do not fit in any of the others and appear in a sufficient quantity that they define a class by themselves. This complicates the original manual classification and reduces performance on the experimental execution.

A graph of the manually classified articles is shown in Figure 1[2]. This graph is computed using the document cosine similarity on a 300 dimension representation obtained with

[1] The assignment is done by just one person with no extensive knowledge in the field.

[2] An interactive representation is available at `https://grarck.github.io/gephi-graph/network/`

model explained in Section 3.4. Each node is a document and the size depends on its grade, the number of nodes that are close; links between each pair of documents are weighted according to their similarity. Finally, each color is assigned to a class. In this figure, it can be seen that all classes are distinguishable and form big clusters, with more similarity between classes 1-2 and 3-4.

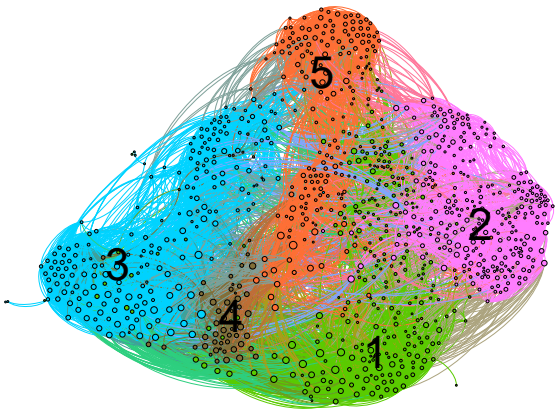


Figure 1: Document graph.

3 Natural Language Processing

The goal of NLP is getting machines to *understand* human language. It is not possible to assign directly some meaning to a word, instead the semantic assignment is given by a set of rules and context, based on the appearance of words hinging upon others. To make such computations it is necessary to represent text into some kind of mathematical form, which usually includes at some point the use of bag of words. This allows obtaining information such as the relevance of a word in a corpus; however, it just considers the number of appearances, order is lost.

This section introduces some of the most common text representations currently used that are present in this project, some of them based on bag of words.

3.1 Term Frequency-Inverse Document Frequency

One of the central tools in NLP is Term Frequency-Inverse Document Frequency (TFIDF) presented in [10], which is a statistic that shows the importance of a word in a particular document belonging to a set of documents. For this reason, it has been used as a base in some search engines.

The term-frequency of a term ***t*** in a document ***d*** is represented in (1), where the numerator is the number of appearances of the word in a document, while the denominator is the total number of words in that document.

$$tf(t,d) = \frac{f_{t,d}}{\sum\limits_{t\prime \in d} f_{t\prime,d}} \tag{1}$$

Inverse document frequency is a measure of the importance of a term ***t*** in the set of documents ***D***. It is computed as the inverse fraction of documents ***D*** that contain the term ***t***, where $|\cdot|$ represents the cardinality of a set, over the cardinality of documents, this is the total number of documents, logarithmically scaled:

$$idf(t,D) = \log \frac{|D|}{|\{d \in D : t \in D\}|} \tag{2}$$

Finally, the TFIDF value assigned to each word in each document is computed in (3) as the product of (1) and (2), obtaining a $D \times N$ matrix.

$$\begin{aligned} tfidf(t,d,D) &= tf(t,d) \cdot idf(t,D) \\ &= \frac{f_{t,d}}{\sum\limits_{t\prime \in d} f_{t\prime,d}} \cdot \log \frac{|D|}{|\{d \in D : t \in D\}|} \end{aligned} \tag{3}$$

This representation weights each word in a document as a function of its relevance in the entire corpus and the document itself: if a word is very rare in the corpus, but common in just a few documents, its value will increase, however if it is common to all of them, it will be reduced.

3.2 Topic modeling

Latent Dirichlet Allocation (LDA) [11] is an unsupervised generative statistical topic model used for extracting underlying topics of document collections. From the assumption that a document is a sequence of N words drawn from a multinomial distribution θ over topics or themes, and topics are drawn from a multinomial distribution over a vocabulary, it explains a set of latent variables given some observations. For simplicity and some properties, these distributions are assumed to be Dirichlet distributions, with priors α and β. The generative process is the following:

1. Choose $N \sim \text{Pois}(\xi)$
2. Choose $\theta \sim \text{Dir}(\alpha)$
3. For each of the *N* words w_n:
   (a) Choose a topic $z_n \sim \text{Multinomial}(\theta)$
   (b) Choose a word w_n from $p(w_n \mid z_n, \beta)$

The Poisson distribution is not critical and some other may be used, it is also independent of all other generating variables.

By just employing words, the only available information, the objective of LDA optimization is finding the composing topics, latent variables and soft topic-document relation. However, LDA entails some problems as the number of topics used must be selected prior to any calculation, and the optimal number is not known so a general search must be done, typically using perplexity or a likelihood method

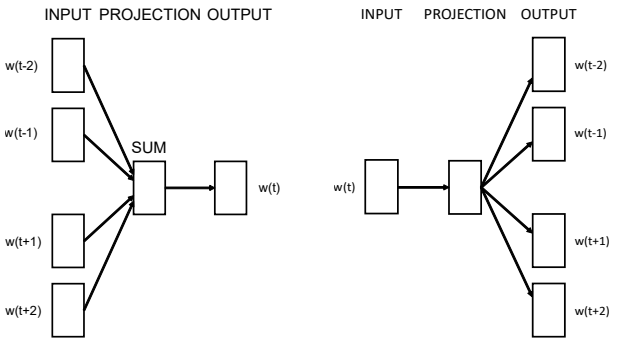
called coherence. Also, the structure of the words in the text is lost. Some improvements have been made on this subject integrating word embeddings to capture semantic properties with Dirichlet distributions [12] or some other distributions [13, 14].

3.3 Word embeddings

One of the techniques to fix the word-order loss and get semantic properties of vocabulary is word embedding. Word embedding is an NLP technique that consists on representing words as D-dimensional vectors. A popular model that generates word embeddings from a corpus is *word2vec*, which was developed in [15, 16] and revolutionized the field. Unlike the other two methods, it takes semantics into account as it works in a local scope by means of a sliding window, in other words, word co-occurrence is important within a given context of reduced terms, not globally. This produces similar representations for words that appear normally in similar contexts, which allows computing similarity between words. There are two architectures represented in Figure 2:

1. Continuous Bag of Words (CBOW): target word is predicted from context. It works better in small datasets.

2. Skip-gram: surrounding context is predicted from target word. Usually works better in larger datasets and infrequent words.



(a) CBOW (b) Skip-gram

Figure 2: Word2Vec model architectures.

The system implemented to obtain this representation consists on a dense neural network of one hidden layer, with number of neurons equal to the number of desired features, so the output of the model are the weights of this layer for each of the words in vocabulary. Some available pretrained models are available online, such as GloVe[3].

[3] `https://nlp.stanford.edu/projects/glove/`

3.4 Mixed model combining word embeddings and topic models

Following the idea of [12], where authors transform LDA algorithm to obtain a vector representation of documents based on their topics. In this section we propose a variant with the same purpose. As in [12], the objective is representing documents in the word feature space, integrating both local features with word embeddings and global features with topic models. The original method has some problems: it is outdated, only works in Python2 and requires much more computation than simple LDA. On the other side, our model does not require LDA modification, and provides a fast document representation.

The method for document transformation into word feature space is shown in Figure 3. First it is required to compute LDA topic models of the articles. In LDA all topics are a distribution of words, so the embeddings of these words are computed. With this, a dot product of topic distribution of words and features of words produces a topic feature representation. Finally, another dot product of document topics and this matrix produces a document feature representation.

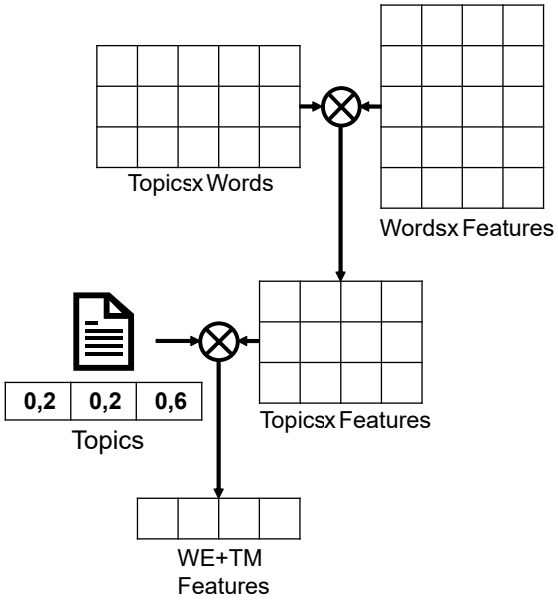


Figure 3: Document vector transformation.

4 Experimental setup

To train each of text representation models and obtain a proper vocabulary, the body of 25 000 non-classified ar-

ticles were used, 40% of the total 63 023 as indicated in section 2.

Preprocessing Prior to any approach, text is processed, dropping all articles whose main language is not recognized as English. Additionally, non-ascii characters and stopwords from other common languages are removed. This filtering is done because most articles are written in English but there are some exceptions. These exceptions add new vocabulary that, with the techniques used, can add noise to the models, deteriorating them. Next, text inflection is removed by lemmatization (words are reduced to lemmas by removing verb tenses or plural forms). Finally, bigrams and trigrams are used to get better contextual information. The tools used to do all this processing can be found in the Python library Gensim[4].

Cross validation In order to get a fair assessment of the different tested approaches, and because of the reduced number of documents manually classified, corpus is split in train, test and validation using 10 fold StratifiedShuffleSplit (75% train, 15% test, 15% validation) proportional to each class.

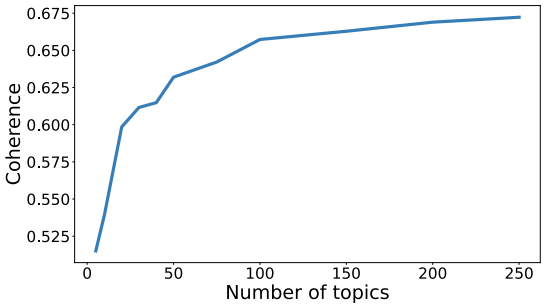


Figure 4: Coherence evolution with number of topics.

Text representations Text is represented using three different techniques:

- **TFIDF**: It is a basic model that just considers top 30 000 terms ordered by frequency across the whole corpus, previously removing very frequent and rare words.
- **Topic models**: Computed with LDA. The values selected for hyperparameters are the default ones in the Mallet[5] implementation. The optimal number of topics is selected by models' C_v coherence from Figure 4. The value increases rapidly up to 50 topics, but it stabilizes around 100-125. Based on coherence behavior, we decided to use 150 topics, as using a larger number of topics does not improve coherence significantly.
- **Word embeddings and topic models**: For the word embeddings part, it is decided to compute own vectors because of the specific and widespread vocabulary. For this, Word2Vec is used on the vocabulary obtained from topic models with algorithm CBOW, creating vectors of 300 features, as this is a widely used selection.

Classification models In our experiments, each of the text representations is combined with the three following automatic classification techniques:

- **KNN**: Two different distance measures are used depending on the input features. For TFIDF, cosine similarity, a measure of similarity between two vectors is used. For topic model distributions and word embeddings it is Jensen-Shannon divergence, a method to measure the similarity between two probability distributions. The number of neighbors selected is 25, selected by cross validation exploring values from 5 to 60 and steps of 5.
- **DNN**: Simple 3-layer dense neural network with ReLU activation and dropout in intermediate layers and softmax activation in last layer. To prevent overfitting, L2 regularization is also included. The number of cells selected for each layer has been selected by cross validation, exploring 2^n values, with n from 4 to 8, finally selecting 64 cells for the first layer and 32 for the second.
- **SVM**: OneVsRest classifier with linear SVM as it has been found [17] to be more efficient in text classification, where number of features and correlation are high. The value of hyperparameter C has been selected by cross validation, exploring 10^n values, with n from -3 to 3, finally selecting $C = 0, 1$.

Metrics Accuracy is the first metric used, computed as the percentage of the total test set that has been classified correctly, although this does not provide a complete characterization of the classification models. Additionally, as the objective is finding and classifying relevant documents, the metrics used to assess the experiments are those typically used in information retrieval: precision, recall and F_1-score. These classifications are binary for each class, this is, for each class either the document does or does not belong to that class. Precision is the fraction of documents whose class has been correctly predicted out of the total number of documents predicted as belonging to that class. Recall is the fraction of documents of the target class that have been retrieved. F_1-score is the harmonic mean of precision and recall.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5}$$

[4] https://radimrehurek.com/gensim/

[5] http://mallet.cs.umass.edu/

$$F_1 = 2\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (6)$$

Equation (4) represents the percentage of retrieved documents that are relevant, and (5) indicates the percentage of relevant documents that are retrieved, where TP, FP, and FN represent respectively, the number of true positives, false positives and false negatives. Both results averaged in (6), penalizing low values of precision and recall.

Precision and recall metrics are commonly plotted in a graph that shows the tradeoff between them for different thresholds. These measures depend on the threshold as it indicates the number of documents that will be retrieved. If the threshold is set very low, more articles will be retrieved, therefore the probability of finding all correct articles will be high (high recall), but many of the articles retrieved will be of different class (low precision). The results obtained for a high threshold value will be very few, but probably correct (high precision, low recall). High precision and recall values produce a high AUC and, therefore, accurate results

In our case, these curves are binary computed for each class, if a document is correctly classified as the given class, it will be a 1, else it is a 0, ordering outputs by probability. This will produce for each class a vector of same length as number of documents classified, where first few values are likely to be the correct one, and last ones are likely to be different. Precision-recall values are computed over these vectors, gradually increasing the number of documents retrieved. Therefore, values of precision will be high at the beginning and lower at the end, while values of recall will be low at the beginning and higher at the end.

5 Results

A summary of the results obtained is shown in the following figures: Table 2: performance breakdown by class, text representation and classification method; Table 3: accuracy performance by representation and classification method; Table 4: composition of most relevant topics by class; Table 5: confusion matrix; Figure 6: precision-recall curves for most accurate methods in each text representation.

In addition to the enumerated text representations, document embeddings provided in the dataset are included as reference. In this section the following nomenclature to refer to the different text representations is used: TM as topic models; WE as word embeddings, following the schema from Figure 3; DE as document embeddings, the ones provided in the dataset.

All the classifiers have been optimized to obtain the minimum error. In the named tables, results are shown for hard decision, the most probable output is the selected one, it is computed as the number of correct predictions over the total 1510 test samples (151 test samples for each of the 10 folds).

Models Analyzing the results from Table 2 by columns, and Table 3, it can be observed that KNN appears to be the least accurate model, being significant the case of DE, where class 2 is selected for almost all cases, thus obtaining an accuracy roughly equal to the presence of that class. Performance of DNN and SVM varies with each text representation, where DNN is preferred with TFIDF and DE, and SVM works better with vectors based on topics.

Table 3: Percentage of correctly classified documents.

	KNN	DNN	SVM
TFIDF	70,27%	78,74%	77,35%
TM	72,19%	72,85%	72,98%
WE	59,60%	70,60%	75,10%
DE	26,49%	80,80%	77,95%

Text representation Tables 2 and 3 also provide some information regarding the text representations used. The representation with TFIDF generates a 30 000-dimension vector for each text, producing robust representations, hence showing best results in all models and classes on average, being the closest to DE. TM and the proposed representation combining topic models and word embeddings produces much lower dimensionality vectors, only 150 and 300 respectively; nevertheless, this type of representations takes up less space and can be computed relatively fast. TM maintains similar performance across all classifiers, far from being the best in both DNN and SVM. In contrast, WE presents worse results for KNN and DNN, but improves significantly when employing SVM, being closer to TFIDF. These results for WE indicate that these features capture the discriminative information in the dataset, though, at the same time, the underlying models can also be interpreted for a better understanding of the dataset and each particular class.

Classes Figure 5 represents document vectors in a 2D space, similar to Figure 1, but with regions according to k closest neighbors. The 2D map has been computed on TM, as it provides best results with KNN classification, with t-SNE[6]. Each symbol represents one of the classes defined in Table 1.

[6] `https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html`

Table 2: Summary of precision, recall and F_1 results in all approaches.

		Precision			Recall			F1		
	Model	KNN	DNN	SVM	KNN	DNN	SVM	KNN	DNN	SVM
TFIDF	Class									
	1	0,658	0,858	0,844	0,84	0,74	0,757	0,738	0,795	0,798
	2	0,727	0,78	0,733	0,805	0,877	0,932	0,764	0,826	0,821
	3	0,671	0,735	0,728	0,723	0,823	0,81	0,696	0,776	0,766
	4	0,677	0,831	1	0,489	0,544	0,367	0,568	0,658	0,537
	5	0,86	0,77	0,788	0,435	0,768	0,671	0,578	0,769	0,725
		Precision			Recall			F1		
	Model	KNN	DNN	SVM	KNN	DNN	SVM	KNN	DNN	SVM
TM	Class									
	1	0,743	0,801	0,809	0,688	0,723	0,688	0,714	0,76	0,743
	2	0,721	0,731	0,678	0,762	0,838	0,873	0,741	0,781	0,763
	3	0,677	0,641	0,685	0,871	0,816	0,855	0,762	0,718	0,76
	4	0,667	1	0,833	0,644	0,022	0,556	0,655	0,043	0,667
	5	0,788	0,752	0,784	0,587	0,713	0,526	0,673	0,732	0,629
		Precision			Recall			F1		
	Model	KNN	DNN	SVM	KNN	DNN	SVM	KNN	DNN	SVM
WE	Class									
	1	0,559	0,725	0,781	0,56	0,632	0,705	0,559	0,676	0,741
	2	0,57	0,674	0,78	0,613	0,853	0,79	0,59	0,753	0,785
	3	0,651	0,689	0,738	0,655	0,794	0,781	0,653	0,738	0,759
	4	0,606	0,875	0,634	0,667	0,311	0,656	0,635	0,459	0,645
	5	0,627	0,744	0,728	0,542	0,639	0,758	0,581	0,688	0,742
		Precision			Recall			F1		
	Model	KNN	DNN	SVM	KNN	DNN	SVM	KNN	DNN	SVM
DE	Class									
	1	0,265	0,834	0,785	0,1	0,815	0,823	0,145	0,824	0,803
	2	0,265	0,835	0,817	0,9	0,845	0,825	0,409	0,84	0,821
	3	0	0,733	0,73	0	0,787	0,697	0	0,759	0,713
	4	0	0,699	0,677	0	0,644	0,7	0	0,671	0,689
	5	0	0,852	0,802	0	0,819	0,771	0	0,836	0,786

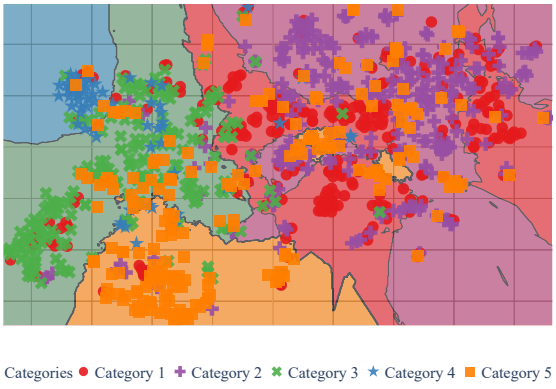


Figure 5: 2D scatter plot representation of TM+KNN classification with $k = 25$.

Table 5 displays the confusion matrix of WE+SVM, showing the distribution of predicted classes compared with the real classes. Figure 5 along with Table 5 allow us to arrive at equivalent conclusions. Classes 1 and 2 are the most similar ones and significantly overlap, being clearly distinguishable from class 4, which is the one with fewer documents. From Table 2 it can be observed that the recall values for class 4 are very low, this is, number of retrieved documents of this class is very low, but most of them will be correct, as denotes the high precision. Class 5 does not overlap significantly with any other class.

This difference in classification performance is better understood with Table 4[7]. This table shows the top 5 terms for the most important topics of each of the classes, the ones that are most present and define them.

The defining terms for class 1 seem to be related to the disease itself and clinical environment: coronavirus, sarscov, patient, etc. Top vocabulary of class 2 is closely related to class 1, and many of the terms are expected to appear in both classes, hence the overlapping. As the context of class 3 is also the disease, some of the terms are similar to class 1, but includes more *social* vocabulary. With class 4 occurs

[7] An interactive representation is available at `https://grarck.github.io/gephi-graph/terms.html`

Table 4: Top 5 representative terms for most relevant topics of each class.

Class	Top topic					
1	1	sarscov	coronavirus	human	sars	coronaviruses
	2	patient	covid	severe	sarscov	symptom
	3	patient	day	included	admission	outcome
	4	patient	treatment	clinical	risk	management
2	1	rsv	respiratory_virus	detected	viral	respiratory
	2	increased	potential	difference	response	impact
	3	viral	host	protein	cell	infected
	4	human	process	approach	development	complex
3	1	country	covid	day	lockdown	pandemic
	2	development	process	approach	technology	system
	3	population	model	infected	rate	parameter
	4	covid	pandemic	covid_pandemic	care	resource
4	1	mental_health	stress	anxiety	individual	symptom
	2	participant	survey	respondent	response	knowledge
	3	work	family	experience	worker	people
	4	epidemic	outbreak	china	day	wuhan
5	1	algorithm	problem	set	method	step
	2	human	process	approach	development	complex
	3	feature	training	model	dataset	class
	4	development	process	approach	technology	system

something similar, but related to mental health instead of society. Last, characteristic vocabulary of class 5 is vaguer. There is no defining key term, which is expected as any non-covid related article should be included here.

Table 5: Class assignment distribution of WE+SVM.

Real class \ Class assigned	1	2	3	4	5
1	282	40	43	10	25
2	37	316	11	0	36
3	30	10	242	12	16
4	7	0	13	59	11
5	5	39	19	12	235

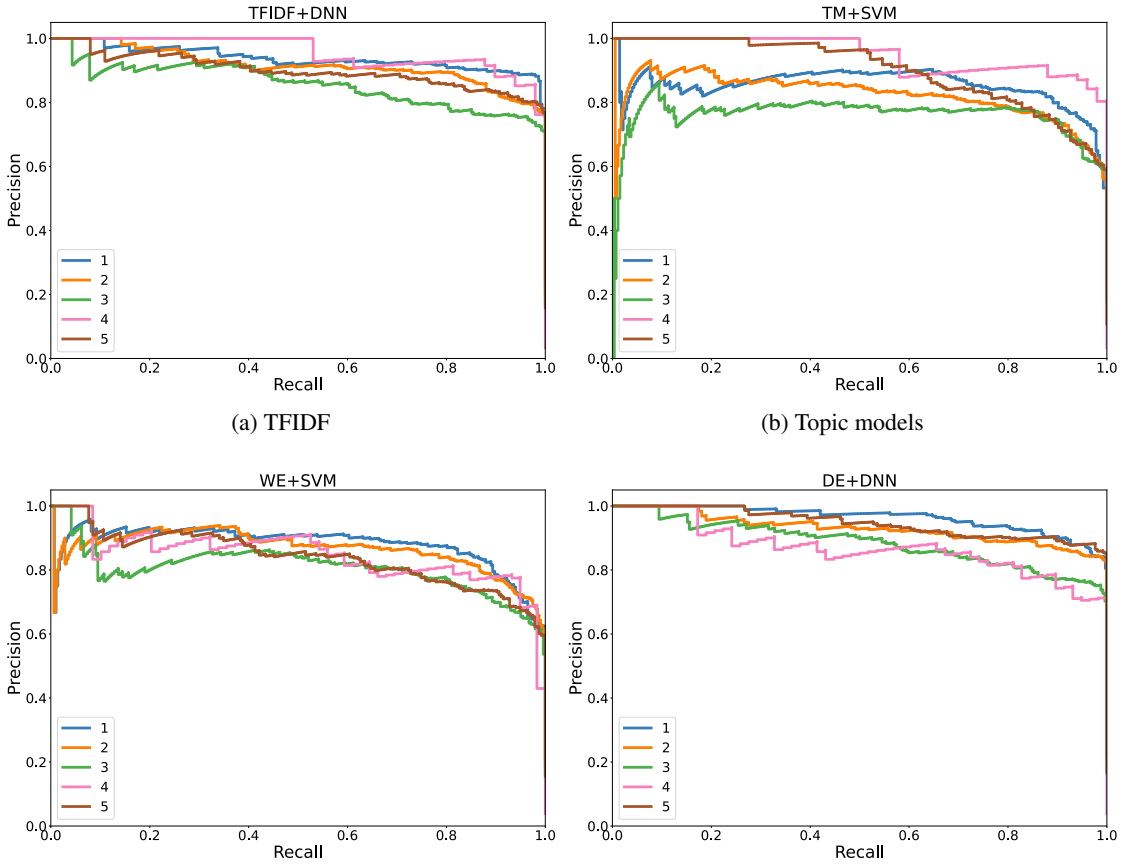
Precision-recall Precision-recall curves for best classification models of all text representations are shown in Figure 6. This image shows that, in most classes, precision values start close to 1 for low recall values, which indicates that top documents retrieved are very likely to be correct, and the more documents retrieved, the more errors will appear, decreasing the precision to around 60 to 90% (depending on the class, and for the best of the approaches) as the recall gets close to 100%. It is observed that WE is, on average, better than TM, although for some particular classes and ranges it can be worse.

6 Conclusion

In this paper, different approaches for COVID-19 paper classification have been explored. It has been shown that the corpus can be split into several meaningful categories, as the tested classification schemes were able to discriminate between the predefined classes quite successfully. Thus, the proposed 5 classes could be included as an additional metadata to enrich paper description and help users of the CORD-19 dataset find the information they are looking for.

We have proposed a method that combines word embeddings and topic models. This model has been proven to achieve good results with the corpus used, improving the performance of plain topic models when employing SVM as classifier, but also preserving the topic features that allow model interpretability. The topic information is kept and expanded, allowing, for example, mapping document vectors to vocabulary by similarity.

The results obtained are consistent in all experiments and show clear differences among papers previously considered as similar, where some conclusions can be drawn. First, performance of classifiers is highly dependent on the models, with the exception of TM, and being TFIDF the one that yields the most robust results across all of them. From the models computed, TFIDF+DNN is more confident in general. Finally, the mix model proposed achieves better results than plain topic model when employing SVM as classifier, however it does not perform that well on the



(a) TFIDF

(b) Topic models

(c) Word embeddings

(d) Document embeddings

Figure 6: Precision-recall curves for best models of each text representation.

others. In this sense, more investigation is required, as it can be clearly improved.

The initial five classes proposed are a suggestion; however, results show overlap between some of the classes, making it difficult to distinguish them, mostly with classes 1 and 2. Besides, during manual classification a few documents were excluded due to the lack of knowledge on the subject, generating a bias in the results.

It is expected that assigning an adequate category to documents will help researchers from various fields to overcome the problems that arise from the information overload. Those interested can create more suitable categories or even use the ones proposed, expanding the information and labels. The creation of a semantic graph is also helpful to identify relevant connected documents by similarity.

This paper illustrates the feasability of using automatic classification and topic modeling as part of a methodology to enrich the CORD-19 dataset, further research could improve these results in a real case scenario, for example, by having several experts make the class definition and assignment, oriented to specific problems, or combining the proposals from several fields into a global better-defined classification table. Last, this proposal has been applied to scientific papers, but can be expanded to other types of project where it is also important to distinguish between classes.

References

[1] Lucy Lu Wang, et al. CORD-19: The COVID-19 Open Research Dataset. *arXiv:2004.10706 [cs]*. URL `http://arxiv.org/abs/2004.10706`.

[2] Honglun Zhang, Liqiang Xiao, Wenqing Chen, Yongkun Wang, and Yaohui Jin. Multi-Task Label Embedding for Text Classification. *arXiv:1710.07210 [cs]*, October 2017. URL `http://arxiv.org/abs/1710.07210`. arXiv: 1710.07210.

[3] Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. Joint Embedding of Words and Labels for Text Classification. *arXiv:1805.04174 [cs]*, May 2018. URL `http://arxiv.org/abs/1805.04174`. arXiv: 1805.04174.

[4] Adji B. Dieng, Chong Wang, Jianfeng Gao, and John Paisley. TopicRNN: A Recurrent Neural Network with

Long-Range Semantic Dependency. *arXiv:1611.01702 [cs, stat]*, February 2017. URL http://arxiv.org/abs/1611.01702. arXiv: 1611.01702.

[5] Wenlin Wang, Zhe Gan, Wenqi Wang, Dinghan Shen, Jiaji Huang, Wei Ping, Sanjeev Satheesh, and Lawrence Carin. Topic Compositional Neural Language Model. *arXiv:1712.09783 [cs]*, February 2018. URL http://arxiv.org/abs/1712.09783. arXiv: 1712.09783.

[6] Liang Yao, Chengsheng Mao, and Yuan Luo. Graph Convolutional Networks for Text Classification. *arXiv:1809.05679 [cs]*, November 2018. URL http://arxiv.org/abs/1809.05679. arXiv: 1809.05679.

[7] Francesco Gargiulo, Stefano Silvestri, Mario Ciampi, and Giuseppe De Pietro. Deep neural network for hierarchical extreme multi-label text classification. *Applied Soft Computing*, 79:125–138, June 2019. ISSN 15684946. doi: 10.1016/j.asoc.2019.03.041. URL https://linkinghub.elsevier.com/retrieve/pii/S156849461930167X.

[8] Ronghui You, Zihan Zhang, Suyang Dai, and Shanfeng Zhu. HAXMLNet: Hierarchical Attention Network for Extreme Multi-Label Text Classification. *arXiv:1904.12578 [cs, stat]*, March 2019. URL http://arxiv.org/abs/1904.12578. arXiv: 1904.12578.

[9] Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. AttentionXML: Label Tree-based Attention-Aware Deep Model for High-Performance Extreme Multi-Label Text Classification. *arXiv:1811.01727 [cs]*, November 2019. URL http://arxiv.org/abs/1811.01727. arXiv: 1811.01727.

[10] Gerard Salton and Michael J. McGill. *Introduction to modern information retrieval*. McGraw-Hill computer science series. McGraw-Hill, New York, 1983. ISBN 978-0-07-054484-0.

[11] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *2003*, page 30, 2003.

[12] Christopher E. Moody. Mixing Dirichlet Topic Models and Word Embeddings to Make lda2vec. *arXiv:1605.02019 [cs]*, May 2016. URL http://arxiv.org/abs/1605.02019. arXiv: 1605.02019.

[13] Rajarshi Das, Manzil Zaheer, and Chris Dyer. Gaussian LDA for Topic Models with Word Embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 795–804, Beijing, China, 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1077. URL http://aclweb.org/anthology/P15-1077.

[14] Ximing Li, Jinjin Chi, Changchun Li, Jihong Ouyang, and Bo Fu. Integrating Topic Modeling with Word Embeddings by Mixtures of vMFs. page 10, 2016.

[15] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]*, September 2013. URL http://arxiv.org/abs/1301.3781. arXiv: 1301.3781.

[16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and their Compositionality. page 9, October 2013.

[17] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A Practical Guide to Support Vector Classification. *2003*, page 16, May 2016.