Topic Modeling

Clustering and extracting topics from a set of documents

Yorick Estievenart

AI Center of Excellence – University Incubator
University of Mons
Sweden

November 8, 2023

Contents

1 Introduction

Machine learning models require more and more data to be trained, particularly in the field of natural language processing (NLP). Manually labeling data takes up a considerable amount of time that could be devoted to a more useful task. One way of solving this problem is to use topic modeling to help us divide the data into clusters, and then label the data using these clusters as classes [1]. Topic modeling is defined as a statistical and probabilistic technique used to automatically identify topics or themes in a collection of documents [3]. This document will be used to answer the following two questions:

- How can we correctly group a dataset of documents (i.e. a corpus of text) by topic?
- How can this help us to label a given dataset?

This report will be divided into two different scenarios, as shown in Figure 1. First, we will focus on the scenario in which the input data are documents and topics (represented by keywords) into which we want to classify the documents. This scenario is considered to be part of topic modeling, since we still need to identify the topic within each document in order to assign them to the desired topic as input. But it could be more precisely defined as unsupervised topic classification. For the sake of simplicity, we will consider this to be topic modeling. The second scenario is one in which the input data are documents only. We need to extract topics for each group of documents. It should be noted that in the second scenario, there is a version of the models, called the guided version, which also takes as input keywords for several topics in order to guide the extraction. Nevertheless, this version is still considered part of the second scenario, as it simply helps the model to extract topics, but its aim is not to classify documents into these topics.

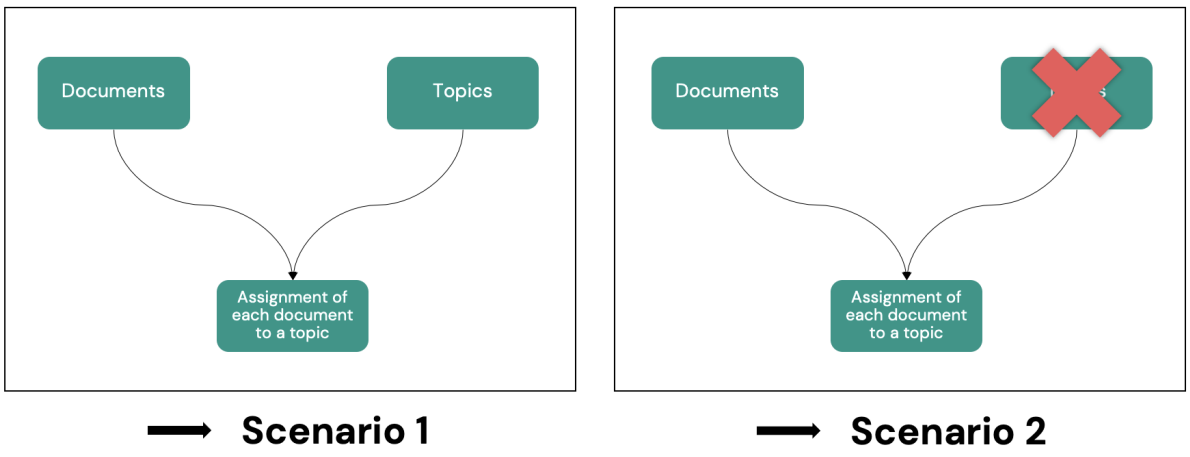

Figure 1: Difference between topic clustering and topic modeling.

It should be noted that the boundary between these two scenarios is very narrow. In fact, we can use the models of the second scenario to perform the task of the first if we use the extracted topics and compare them to those given as input in the first scenario. An example of this technique will be provided in section 4.7.2. Figure 1 shows an overview of the models used in the two scenarios. The models in gray are those that will be analyzed in greater detail in this report, obviously because it was not possible to compare all the models in the limited time available. Nevertheless, each scenario will contain a section explaining the current state of the art and a brief explanation of the models. We hope this will help the reader to develop ideas for future work.

[1] Code for this document is available at the following url. Each section of this report is represented by a separate folder.

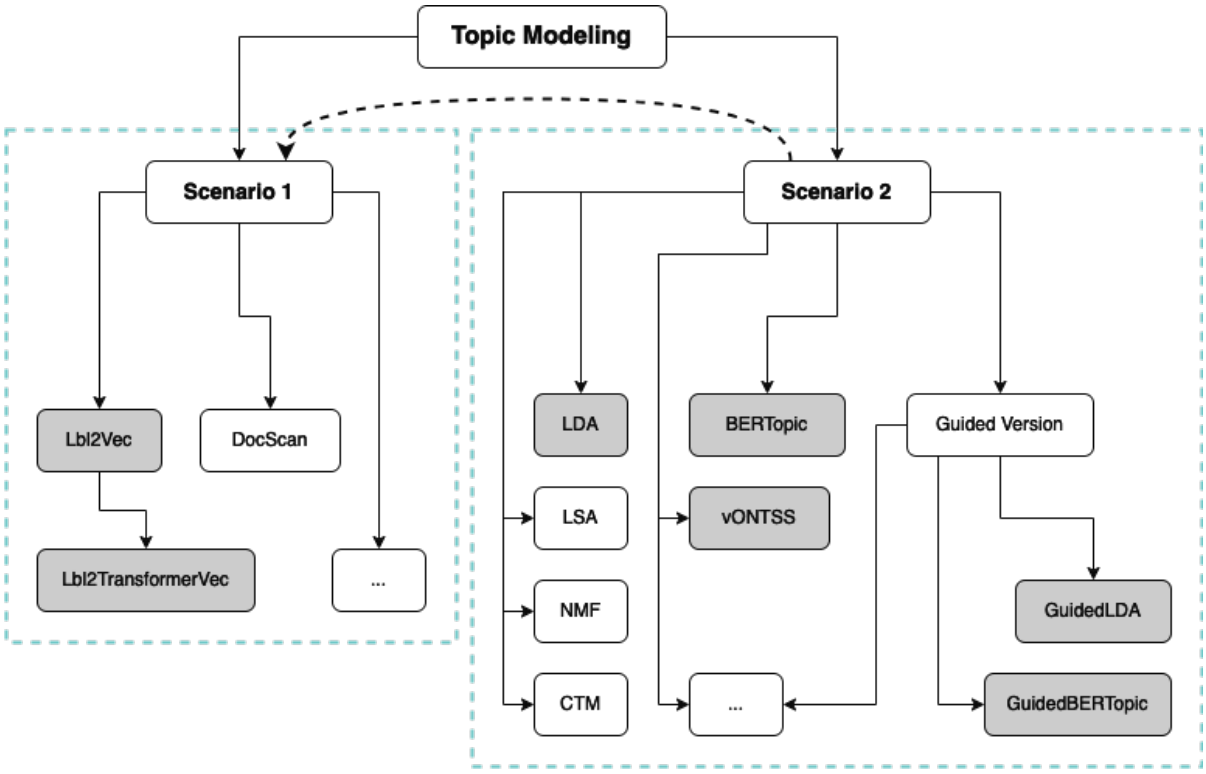

Figure 2: Overview of topic modeling. The gray boxes are the models that will be analyzed in greater detail.

2 Datasets

Before researching the two scenarios and trying out various models in an attempt to solve the data labeling problems, we need to have some benchmark datasets, popular in the architecture, to help us compare our results. Table 1 shows the following baseline datasets used throughout the project. These datasets were extracted using the OCTIS library and pre-processed by manually rewriting each ambiguous class name with an easier-to-interpret name. The predefined keywords needed to represent the classes/topics are presented in the next section.

Name	# Docs	# Train/Test Docs	# Classes
BBC_News	2225	1780 / 445	5
20NewsGroup	16309	13047 / 3262	20
DBLP	54595	43676 / 10919	4
M10	8355	6684 / 1671	10

Table 1: Dataset extracted from the OCTIS library and used throughout the project.

2.1 Keywords

The keywords used for each class in each dataset are shown in Tables 2 and 3. The reason for replacing the original heading name with something more legible is that some models use these keywords to find other similar words. Word abbreviations distort these similarity measures. For example, ”sci.space” is not perfectly similar to ”space”. A more easily interpretable name was therefore needed for these models to work properly. These predefined keywords were defined solely on the basis of the class name, or by consulting the various dataset class descriptions on the Internet (e.g. for dataset M10). Nevertheless, other more in-depth techniques such as document content searches could be used. This will be done for some of the experiments in this report.

Topic	Topic Index	Keywords
soc.religion.christian	0	religion christianism
rec.sport.hockey	1	sport hockey
sci.space	2	science space
comp.os.ms-windows.misc	3	computer operating systems microsoft windows
comp.graphics	4	computer graphics
comp.sys.mac.hardware	5	computer systems mac hardware
rec.motorcycles	6	motorcycles
comp.sys.ibm.pc.hardware	7	computer systems ibm pc hardware
talk.politics.guns	8	politics guns
sci.electronics	9	science electronics
talk.politics.misc	10	politics miscellaneous
rec.sport.baseball	11	sport baseball
talk.religion.misc	12	religion miscellaneous
sci.crypt	13	science cryptography
alt.atheism	14	atheism
talk.politics.mideast	15	politics middle east
sci.med	16	science medicine
rec.autos	17	automobiles
comp.windows.x	18	computer windows x
misc.forsale	19	miscellaneous for sale

Topic	Topic Index	Keywords
2	0	archaeology
3	1	computer science
4	2	financial economics
0	3	agriculture
7	4	petroleum chemistry
9	5	social science
1	6	biology
8	7	physics
6	8	material science
5	9	industrial engineering

Table 2: Keywords for the 20NewsGroup dataset (left) and the M10 dataset (right).

Topic	Topic Index	Keywords
business	0	business
sport	1	sport
entertainment	2	entertainment
tech	3	technology
politics	4	politics

Topic	Topic Index	Keywords
2	0	computer vision
0	1	database
3	2	data mining
1	3	artificial intelligence

Table 3: Keywords for the BBC News dataset (left) and the DBLP dataset (right).

2.2 Straightforward data analysis

Figure 2.2 shows the class distributions for the datasets. It can be seen that the DBLP and M10 datasets have more consistent document lengths than the 20NewsGroup and BBC News datasets. In addition, the BBC News and DBLP datasets may be more suitable for testing, as they have fewer classes. As the BBC News dataset has many advantages, we will focus mainly on it for our primary experiments and then compare the results obtained with the other datasets.

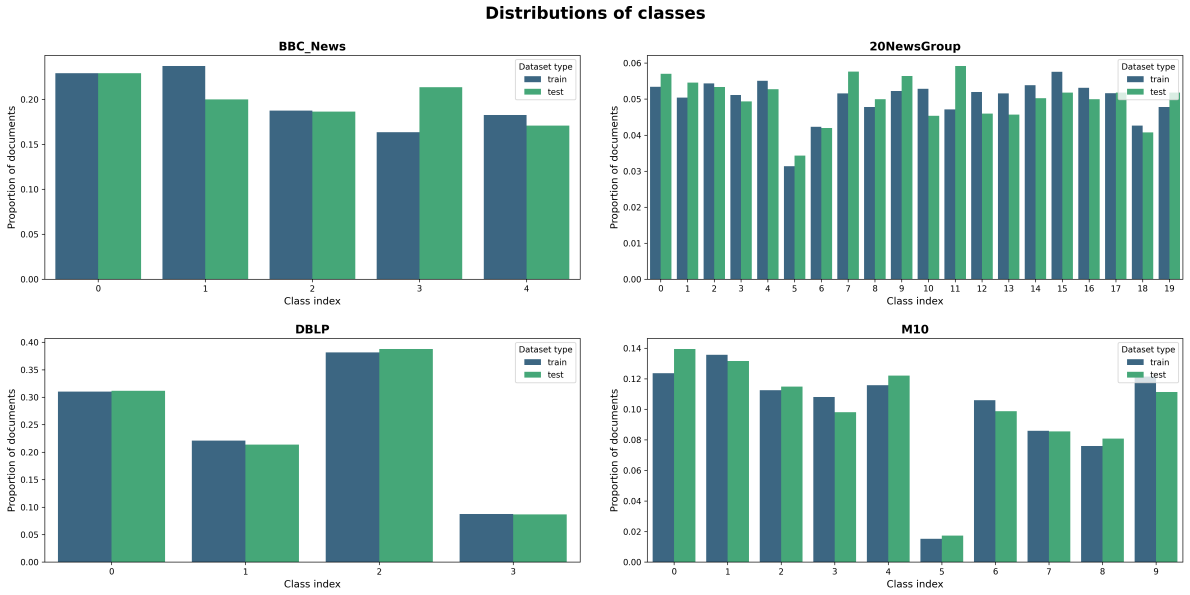

Figure 3: Class distribution for all datasets.

Figure 2.2 shows the document length distribution for the datasets. The DBLP and M10 datasets contain shorter documents, raising a question. Is it better to have a large number of shorter documents, or a few larger ones? After some research, the choice depends on a number of factors, and there is no single answer. Both approaches have their advantages and limitations. For example, a large number of shorter documents might improve diversity, but they would take longer to process. On the other hand, a small number of larger documents could solve this problem of computational complexity, but one limitation could be that the topics extracted are too general and more specific topics are avoided. Some research also indicates that certain models such as LDA, for example, only work on large collections of documents [5]. The study of the number of documents and their length is left for future work.

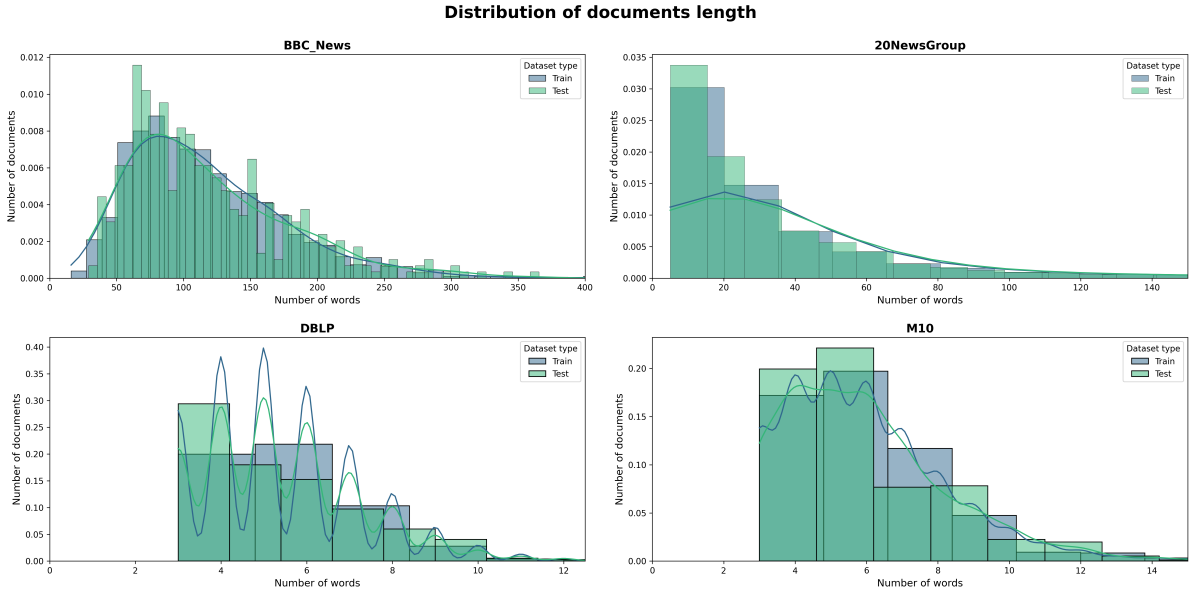

Figure 4: Document length distribution for all datasets.

The 10 most frequent words for each dataset are shown in Figure 2.2. It might be even more useful if a visualization of the most frequent words were available directly by class. This will be done as part of some experiments. Nevertheless, this graph is still useful for understanding the main content of the datasets.

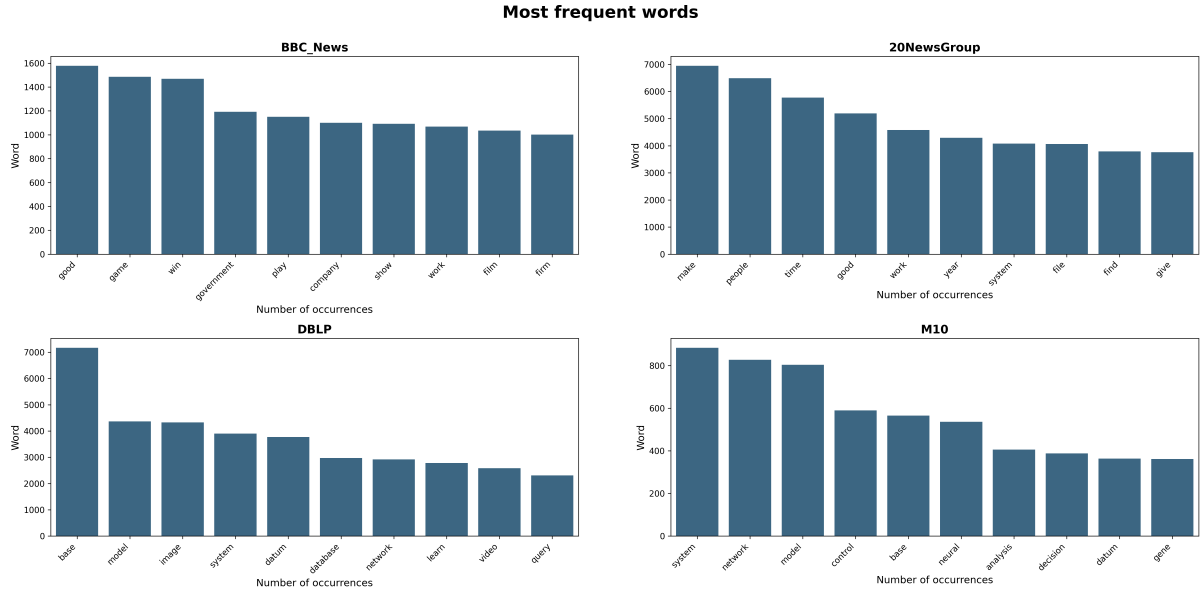

Figure 5: Most frequent words used in each dataset.

The distribution of word occurrences for the datasets is shown in Figure 2.2. The general rule is that most words occur rarely. Furthermore, with the exception of dataset M10, no dataset contains words that occur only 2 or 3 times. This can be explained in part by the fact that the datasets have already been processed and therefore contain no stop words or punctuation. It may also be explained by the fact that pre-processing has removed words that appear only a few times because they contain only a small amount of information.

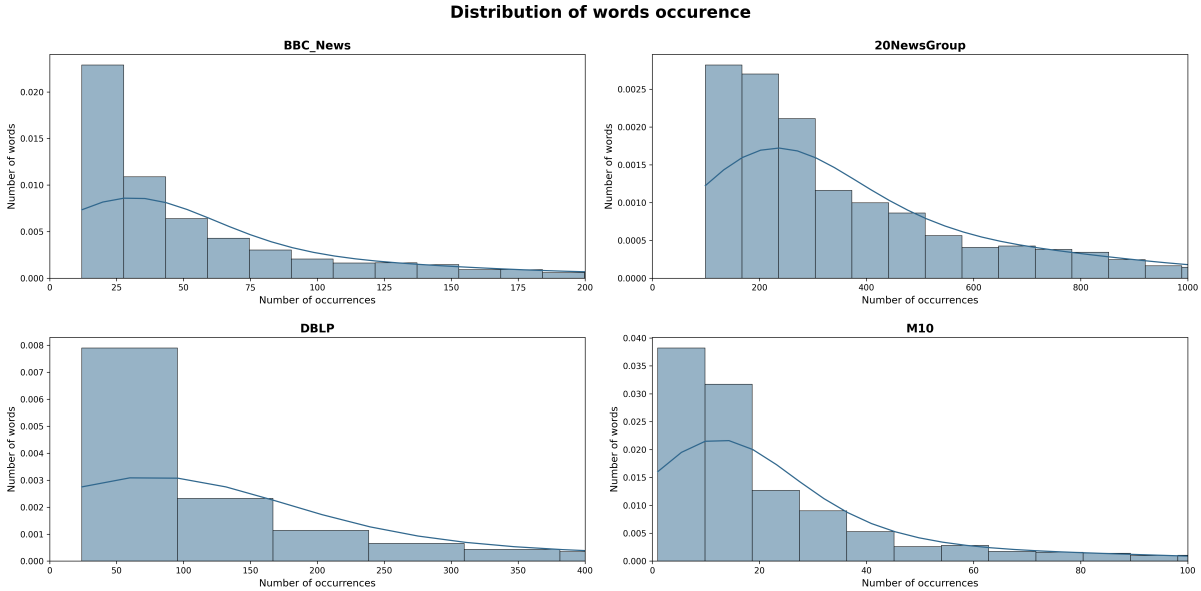

Figure 6: Distribution of word occurrences for each dataset.

3 Scenario 1

This section deals with the scenario in which we receive not only input documents, but also the classes into which we want to classify the documents. Each class is represented by the keywords defined in section 2.1 but, depending on the situation, these can be modified and adapted for analysis. In this section, we will look at the Lbl2Vec model and its transformer-based variants.

3.1 Background

Several methods can be associated with this scenario. The first is DocSCAN [15], an unsupervised text classification approach based on the Semantic Clustering by Adopting Nearest-Neighbors (SCAN) algorithm. With this model, it is sufficient to provide one document per class, and it assumes that neighboring documents in the embedding space must have the same labels. Other methods are based on the models used in the second scenario, such as LDA [2], LSA [9] (also known as LSI) or more advanced methods like BERTopic [4]. These methods use the extracted topics to calculate cosine similarity with the desired classes. These models often differ from this scenario in that they have to deal with documents belonging to uninteresting topics, i.e. potential outliers in the dataset. Finally, the method closest to the definition of this scenario is Lbl2Vec. It takes no account of uninteresting topics and uses predefined keywords to calculate the distance to embedded documents. It is explained in more detail in the next section.

3.2 Lbl2Vec and Lbl2TransformerVec

3.2.1 Theory

Lbl2Vec [14] is an unsupervised algorithm that classifies documents into classes defined on the basis of predefined knowledge, namely the keywords associated with each class. At a high level, this algorithm performs the following tasks:

1. Use manually defined keywords for each category of interest.
2. Create jointly embedded word and document vectors. The idea behind embedding vectors is that similar words or text documents will have similar vectors. Therefore, after creating jointly embedded vectors, documents are located close to other similar documents and close to the most distinctive words.
3. Find the document vectors that are similar to the keyword vectors for each classification category. We calculate cosine similarities between documents and manually defined keywords for each category. Documents that are similar to the category keywords are assigned to a dataset of candidate documents in the respective category.
4. Clean outliers for each classification category using an algorithm called Local Outlier Factor (LOF).
5. Calculate the centroid of outlier vectors cleaned as a label vector for each classification category.
6. The algorithm computes label vector $\Leftrightarrow$ document vector similarities for each label vector and document vector in the dataset. Finally, text documents are classified into the category with the highest label vector $\Leftrightarrow$ document vector similarity.

Figure 3.2.1 shows a visual explanation of the above steps. This algorithm raises a number of questions: How do you choose the right keywords? How do you create embeddings? And is LOF really useful for model performance? The first two questions will be answered in the following sections, while the third is left for future work.

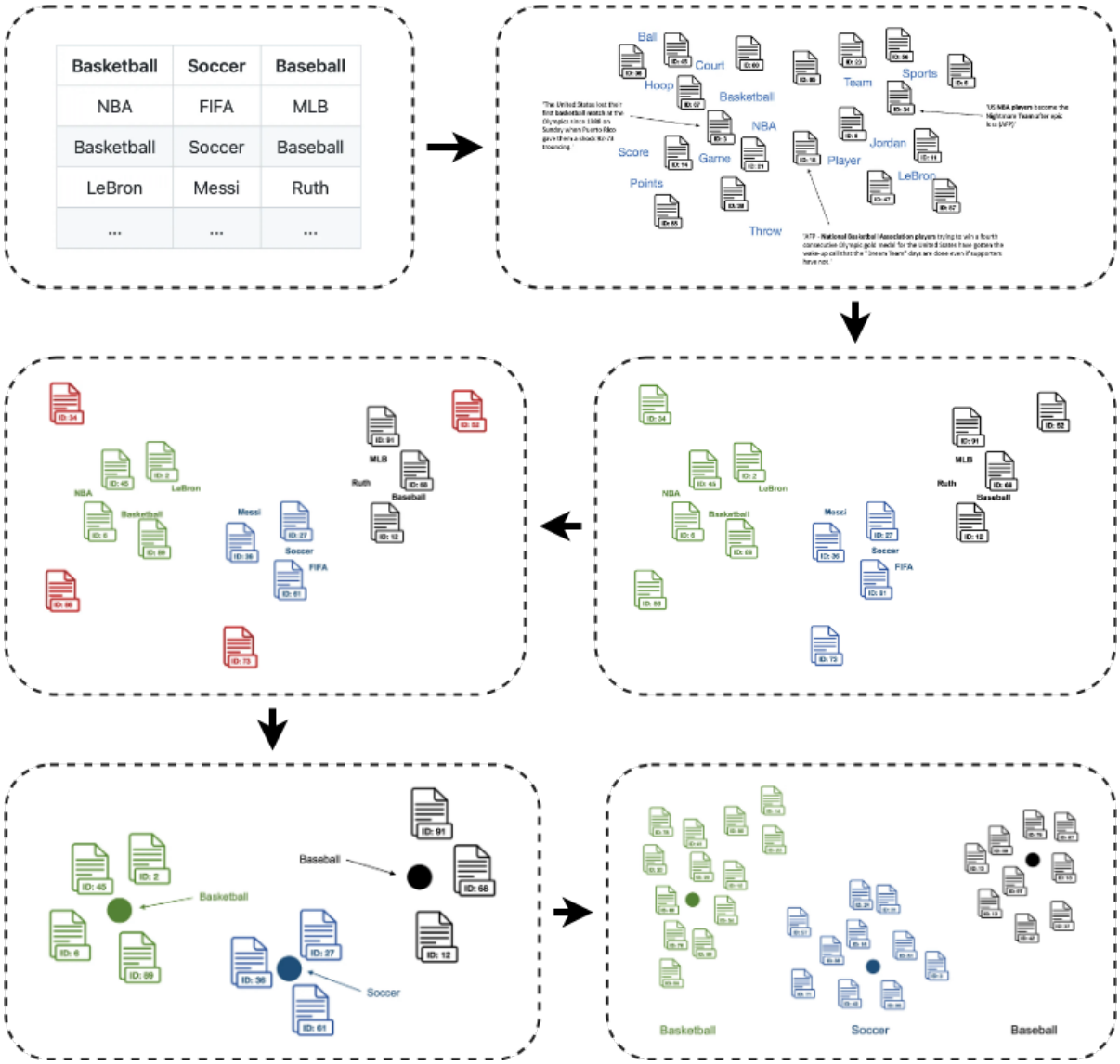

Figure 7: Visual explanation of how this model works.

Note that the original article provides a library for the efficient use of this model. But there are two problems. Firstly, we have no freedom over the internal model. Indeed, it is impossible to extract the embeddings created during inference. Secondly, it sometimes produces strange results. For example, it has a high F1 score on the BBC News dataset with Doc2Vec as embeddings, whereas it has a very low F1 score with a transformer embedding. To solve these problems, we decided to rewrite the model from scratch (you can find it on the Github repository under the name DaCluDeK (Data Clustering with Defined Keywords)).

3.2.2 Embeddings models

Two types of embedding model can be used. One uses Doc2Vec and the other a transformer-based model. The advantages and disadvantages of these two architectures are presented in Table 4. Both models have been used and compared in this section, generally in a 2D setting for visualization purposes. Doc2Vec can directly generate 2D embeddings, but we decided to use another dimensionality reduction method called *t-Distributed Stochastic Neighbor Embedding (t-SNE)*. Automatic generation of 2D embeddings is not possible with transformer-based

models. To solve this problem, we use a dimensionality reduction method called *Principal Component Analysis (PCA)*. The transformer-based model used is *all-MiniLM-L6-v2*. The type of embedding model (Doc2Vec or transformer) defines the name of this model, i.e. Lbl2Vec or Lbl2TransformerVec.

Doc2Vec		**Transformer-based**	
Advantages	**Disadvantages**	**Advantages**	**Disadvantages**
Fast to calculate	Can only use keywords seen during training Poor performance	Can use any keywords Better embeddings leading to better performance	Costly in terms of computing

Table 4: Advantages and disadvantages of Doc2Vec architecture compared with transformer-based models.

A major problem with using Doc2Vec-based models is that they can only use keywords seen during training. This problem can be solved in an unsupervised way by using the similarity between two words. Knowing that, when using Doc2Vec-based models, instead of using the keywords from the datasets, we will calculate the similarity between the keywords and all the words contained in each document. The keywords used will be those that are most similar to the predefined keywords. To do this, the *spaCy* package will be used. Here's how similarity is calculated in spaCy:

1. Word vectors: Each word in the input text is represented by a high-dimensional vector in a continuous vector space.
2. Sentence or document vectors: spaCy combines the individual word vectors in the sentence or document to create a vector representation of the entire sentence or document dataset. This is usually done by taking the average or weighted average of the word vectors.
3. Cosine similarity: Once the sentence or document vectors have been obtained, spaCy calculates the similarity between them using cosine similarity. Cosine similarity measures the cosine of the angle between two vectors and provides a value between -1 (completely dissimilar) and 1 (perfectly similar).

One perplexed user might ask: What is the point of using Doc2Vec if you always have to use a transformer like spaCy to find keywords? It should be noted that spaCy is only used as a pre-processor to retrieve the keywords most similar to the predefined keywords, and is not used directly in the model. This step could be avoided by directly selecting the predefined keywords as words contained in the documents.

3.2.3 Embeddings

Figure 3.2.3 shows an example of 2D embeddings created for Doc2Vec and the transformer-based model. It can be seen that the transformer-based model has a better ability to find embeddings because they are more clustered. Whereas most of the clusters found by Doc2Vec with t-SNE overlap. But we have to bear in mind that we are in a 2D configuration and it is possible that Doc2Vec also creates good embeddings in a higher dimension. At least, both embeddings respect the property that similar words will have closer embeddings. Without this assumption, calculating the distance between keywords and document embeddings is irrelevant, because if the distance is small, it does not mean that the embeddings are similar.

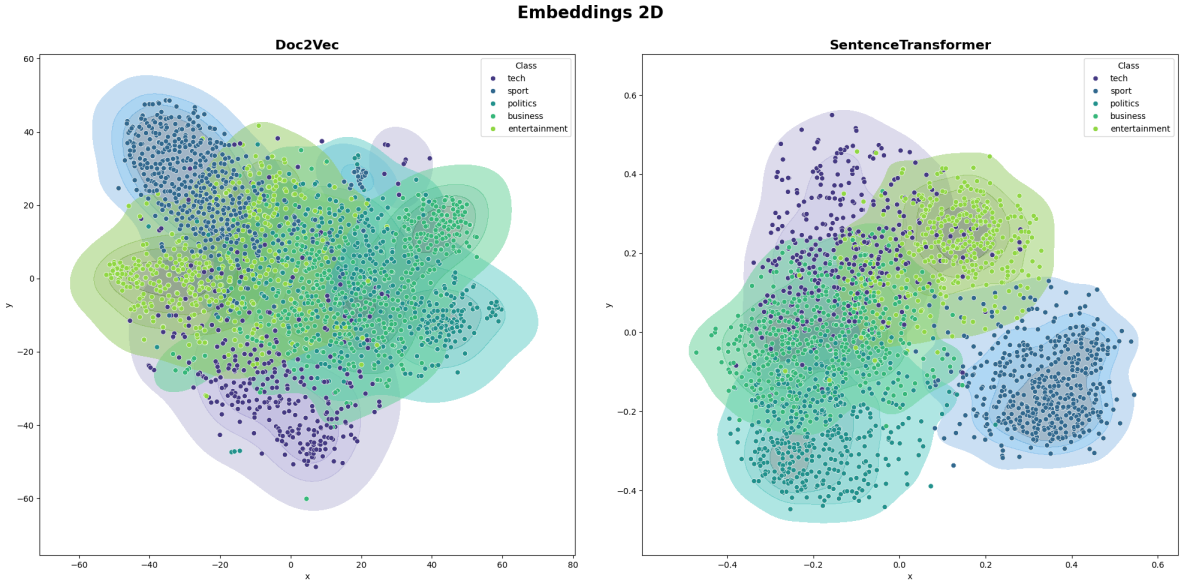

Figure 8: 2D embeddings generated by Doc2Vec and the transformer-based model on the BBC News dataset.

3.2.4 Performances

This section will attempt to answer the following questions: How does the model's performance vary according to the keywords used? And how can we find these keywords, bearing in mind that we are in an unsupervised framework? We will extract three types of keywords: the most frequent (supervised), the most similar (unsupervised) and the most important (supervised). As we were aiming for unsupervised learning, only the most similar will be used, but the other two types of keywords present interesting results that could give the reader ideas for future work.

3.2.4.1 Performance using the most frequent words

The keywords used in this section are the five most frequent words per class and have been extracted in a supervised way. They are listed in Table 5. Figure 3.2.4.1 shows the performance of the two models on the BBC News dataset using these keywords. The measure used is the F1 score. It can be seen that the model based on transformers outperforms Doc2Vec, but remains very unbalanced. The score obtained for the "tech" class is much lower than that obtained for the "politics" class. Nevertheless, the training and test scores are fairly well balanced, which is a positive result, since it could mean that our model is not over-fitted to our training data, and is in fact generalizing to the true distribution of the data.

business	**entertainment**	**politics**	**sport**	**tech**
company	film	government	win	game
firm	good	election	game	technology
market	award	party	play	phone
rise	music	labour	player	mobile
sale	show	plan	good	service

Table 5: The 5 most frequent keywords by class for the BBC News dataset.

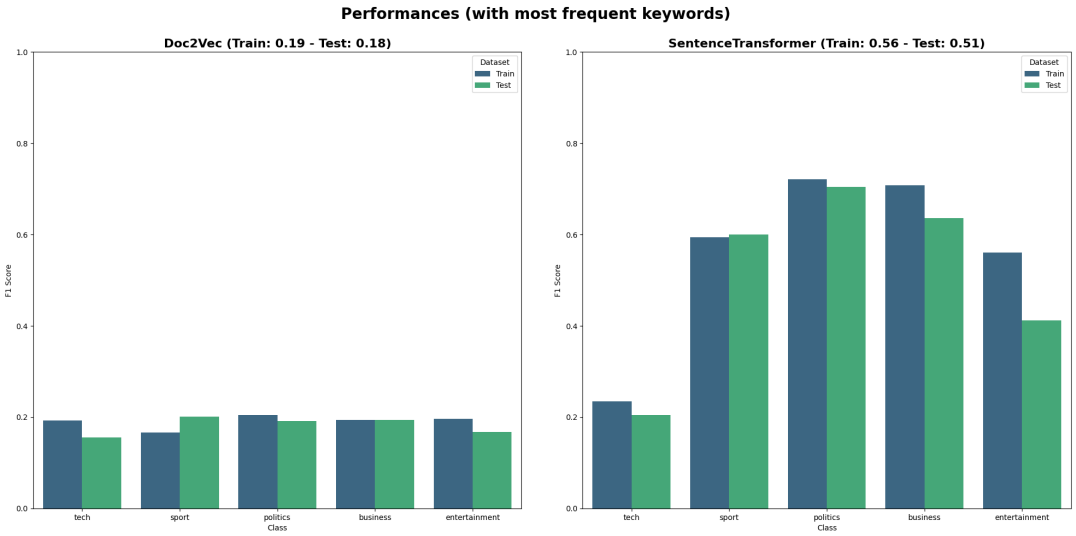

Figure 9: F1 scores obtained after applying Doc2Vec and the transformer-based model to the BBC News dataset using the most frequent words in each class.

3.2.4.2 Performance using the most similar words

The main problem with the previous result is that we retrieve the most frequent words per class in a supervised way. However, our problem is related to unsupervised learning and it requires another method. One solution is to use the keywords defined in Section 2.1 and apply the spaCy algorithm as explained in Section 3.2.2. This approach will enable us to extract keywords that are already contained in documents. A possible improvement would be to use this algorithm with a dictionary of the given language (e.g. English). This would enable it to find more general keywords than those contained in the documents. This possibility is left open for future work. The algorithm uses the five keywords most similar to the predefined keywords, which are listed in Table 6.

business	**entertainment**	**politics**	**sport**	**tech**
business	entertainment	political	sport	technology
corporate	television	democracy	tennis	technological
company	theatre	democratic	rugby	innovation
marketing	gaming	debate	athletic	innovative
industry	multimedia	liberal	football	engineering

Table 6: The 5 most similar keywords by class for the BBC News dataset.

The results are shown in Figure 3.2.4.2. The results were roughly the same, with the difference between training and test scores being smaller for the transformer-based model. The fact that the score did not change for Doc2Vec could be due to the fact that the model makes random predictions as the embeddings overlap. And as we iterate several times to account for the randomness of the model, the scores remain close to the same values. As shown in Section 3.2.5, this intuition that Doc2Vec makes random predictions seems feasible. However, one question remains: why is the F1 score so low in the ”tech” class, and how can this problem be solved? To solve this problem, we will go back to a supervised solution to see what the results are with the ”best” method.

3.2.4.3 Performance using the most important words

To determine which might be the best keywords to use to improve the score, we use a supervised algorithm to extract the most important words. We have applied one *TfidfVectorizer* to the documents before running a *Logistic*

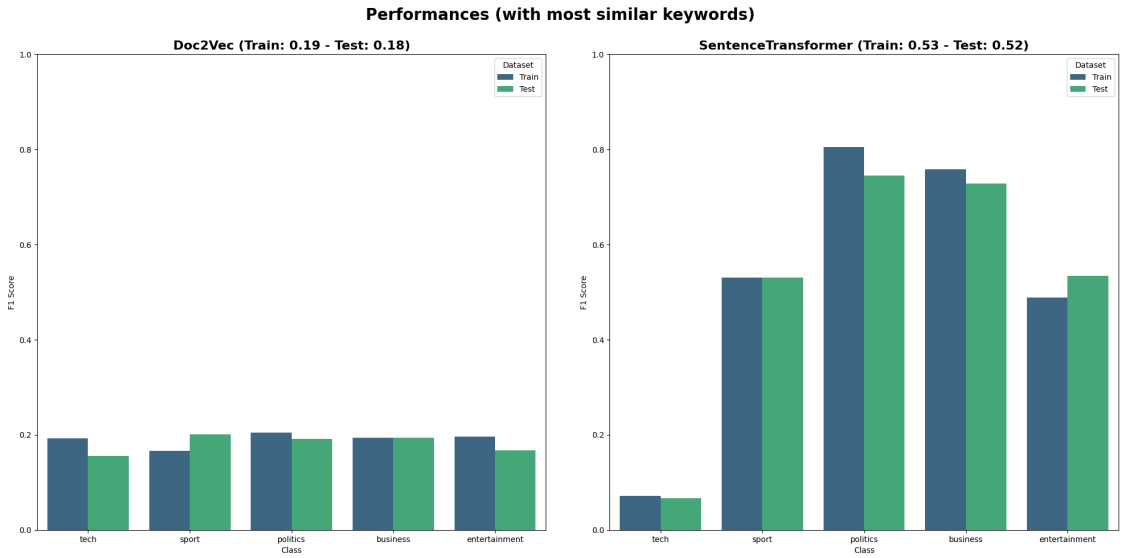

Figure 10: F1 score obtained after applying Doc2Vec and a transformer-based model to the BBC News dataset using the most similar words in each class.

Regression model. This model predicts the classes with an F1 score of 0.99, meaning that the model is able to extract the important words accurately. The 100 most important keywords are extracted and then used to predict classes using our model. The choice of 100 is due to the fact that other numbers were tried and nothing was as good as 100. The results are shown in Figure 3.2.4.3. It can be seen that the scores are more balanced (the ”tech” class is not as bad as with the other methods), but that the average score remains the same. Doc2Vec’s scores have not improved. This can mean that our unsupervised keyword retrieval method is as good as a supervised one.

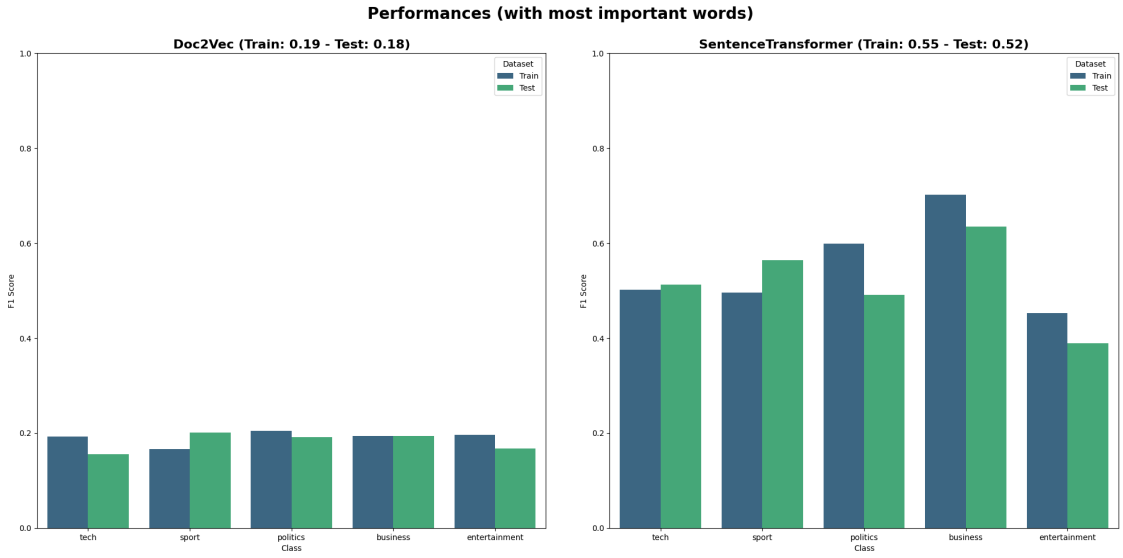

Figure 11: F1 scores obtained after applying Doc2Vec and the transformer-based model to the BBC News dataset, using the 100 most important words in each class.

3.2.5 General performances

As suggested by the previous observation, i.e. that our unsupervised method provided as good an average score as a supervised method, we will see how the models perform on each dataset using the most similar keywords method, and the number of most similar keywords will be set to 10. The results are shown in Figure 3.2.5. In addition, the F1 score obtained using a random method, i.e. by selecting a class at random, is shown. It can be seen that Doc2Vec is often worse than this random classifier, which can be explained by its poor embedding performance. Interestingly, the transformer-based model outperforms the random classifier. Overall, we can say that the results are quite good, since we are in an unsupervised framework and the number of classes is quite high. But compared with a supervised method, the results are still rather mediocre. The results are just as bad when we use the library provided by the author of the scientific paper, with the exception of the BBC News dataset where the F1 score obtained with Doc2Vec is strangely good. What could be different in the paper library that could cause this behavior? And how did they choose the keywords for their model to be declared state-of-the-art in unsupervised text classification?

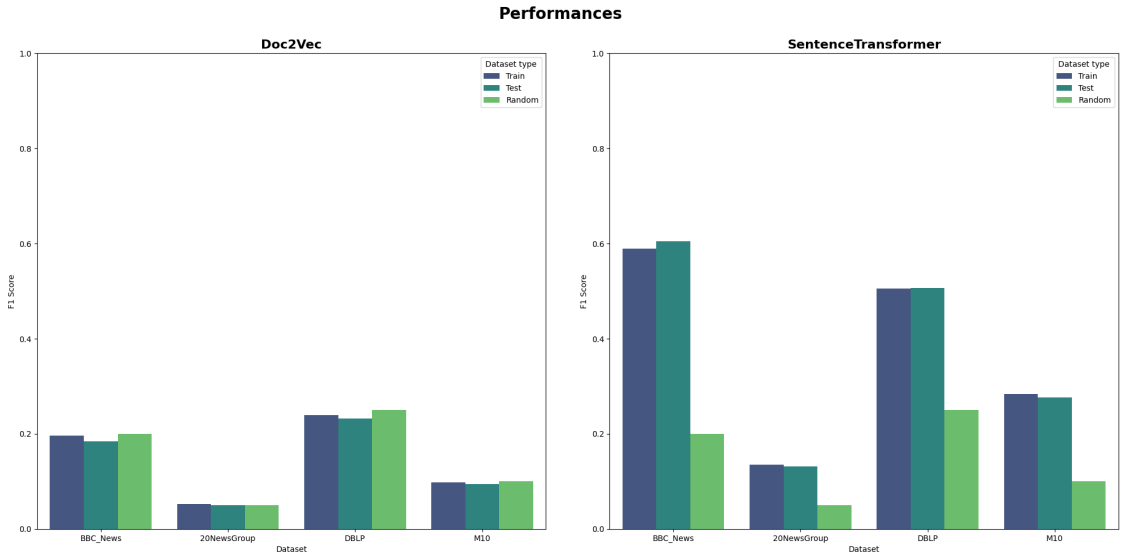

Figure 12: F1 scores obtained after applying Doc2Vec and the transformer-based model to all datasets, using the 10 most similar words in each class.

Based on these results and those of the embedding, we believe there may be a problem with the Doc2Vec implementation in the library. The implementation is similar to the one in the article, and many ideas have been tried without result. Future work could involve a second reading of the code to ensure that the results are good. It is worth noting that we also tried using Doc2Vec with t-SNE to see if the results improved, but this had no impact and performance did not change. If a second reader confirms that there are no errors, the possible explanation is that there is too much overlap between clusters, therefore all the centroids found by Lbl2Vec are almost the same, forcing the model to make random predictions. Future work could display the centroid of keywords defined with the two embedding models to see if they are close to each other. If they are, this would confirm our intuition.

In the article, they claim that their method is state-of-the-art in this scenario, with an F1 score of 0.75 on the 21NewsGroup dataset. We were therefore interested in how they found the keywords, as this is the only difference between their results and ours. In the article, they state that they defined the keywords as follows:

> *We emulate human experts ourselves, that define some initial keywords based on the class descriptions only. Then, we randomly select some documents from each class to further derive some salient keywords. In the case of a strict unsupervised setting with completely unlabeled datasets, human experts*

might describe a topic with keywords based on their specific domain knowledge alone and without necessarily being familiar with the document contents.

Therefore, it is likely that they obtained the result by trial and error. They tried a few keywords, calculated the F1 score and tried again until they got the best score. This keyword definition problem remains the biggest challenge of this model, and is left for future work.

3.2.6 Assignment confidence

In the previous section, we observed that the results are not particularly impressive. But we wanted to make the link with confidence in the assignment. How confident is the model in the event of a bad prediction? Can we detect problematic data points? To answer these questions, we began by plotting the confidence score, i.e. the probability emitted by the model when making predictions, of the two models applied to the BBC News dataset. Figure 3.2.6 illustrates this result. It can be seen that Doc2Vec is not at all confident. It is as if he is making random guesses, which is in line with the intuition we had in the previous section. Fortunately, the transformer-based model gives better results.

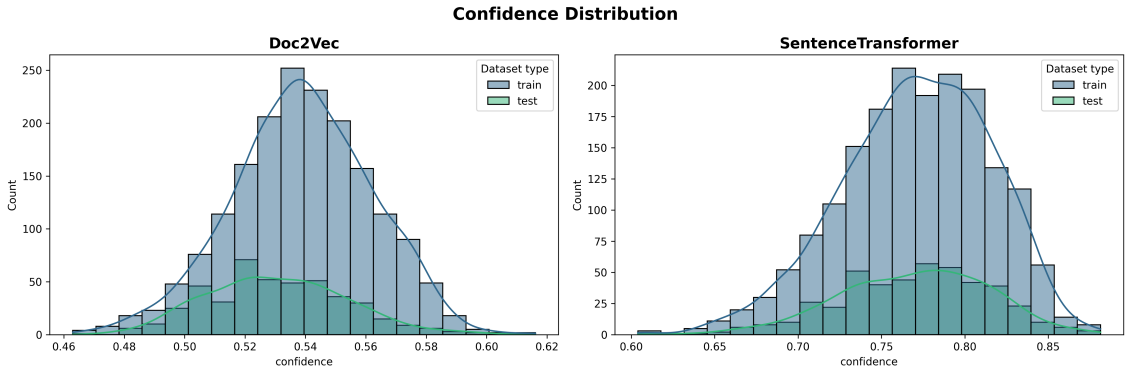

Figure 13: Confidence distributions obtained by applying the models to the BBC News dataset.

In order to distinguish correct from incorrect predictions on the basis of a threshold applied to the probability scores, we plotted these distributions when the models made both correct and incorrect predictions. This is shown in Figure 3.2.6. Unfortunately, the distributions are almost equal. This means that we cannot apply a threshold to differentiate between correct and incorrect predictions.

To ensure that no threshold could be found, we applied the transformer-based model to all four datasets and plotted the distributions of correct and incorrect predictions. The result is shown in Figure 3.2.6. Indeed, we can see that the distributions overlap on each dataset, hence there is no way to detect correct and incorrect predictions based on the confidence score.

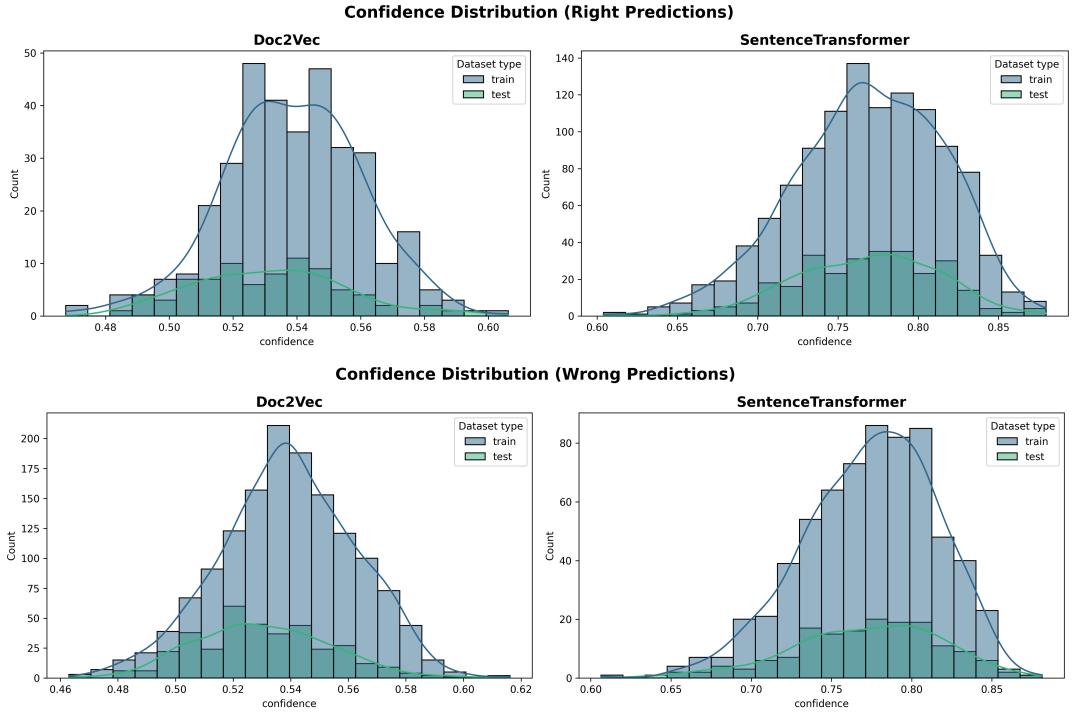

Figure 14: Confidence distributions on correct and incorrect predictions obtained by applying the models to the BBC News dataset.

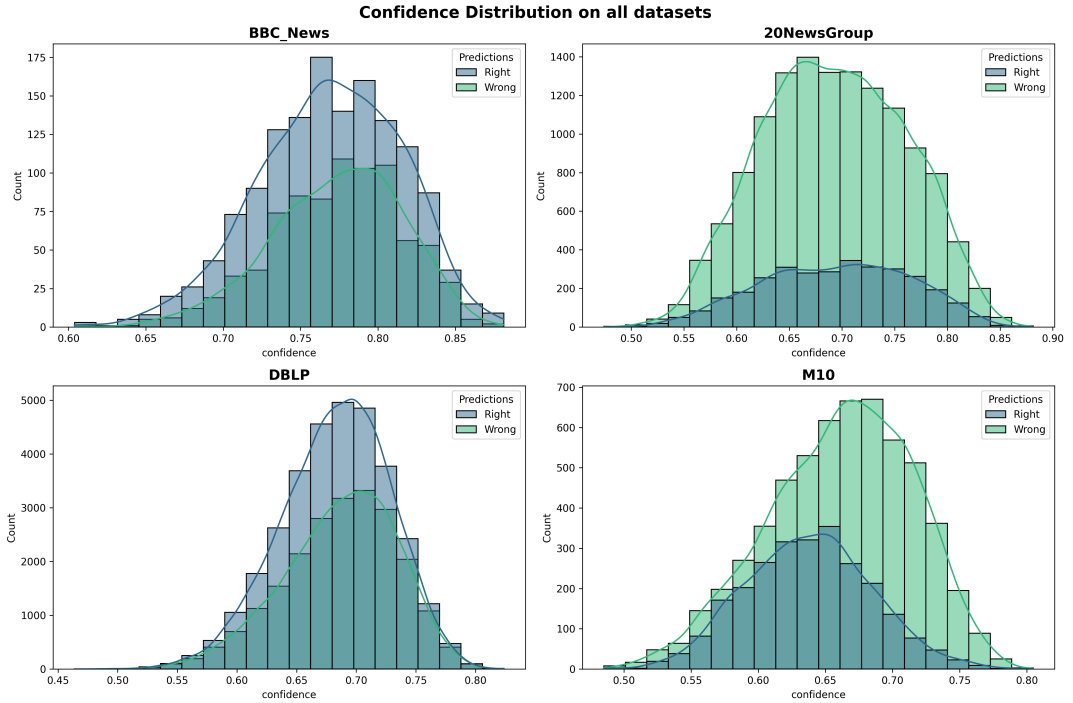

Figure 15: Confidence distributions obtained by applying the transformer-based model to all datasets.

4 Scenario 2

This section deals with the scenario in which we only receive input documents, and not the classes as we saw earlier. This means that we also have to extract the various topics. Compared with the previous section, this could mean that we can use the extracted topics as input data to return to the first scenario. This is one of the analyses that will be attempted in this section.

4.1 Background

The origins of topic modeling date back to the early 2000s, when researchers began exploring probabilistic graphical models to discover thematic structures in large text corpora. One of the fundamental models in this field is Latent Dirichlet Allocation (LDA) [2], a generative statistical model that assumes that each document in a corpus is a mixture of various topics, with each word attributable to one of these topics. Over time, various methods such as non-negative matrix factorization (NMF) [10] and latent semantic analysis (LSA) [9] have emerged. More recently, deep learning approaches that exploit neural networks and transformer architectures to improve topic extraction and representation have gained ground.

Among these deep learning approaches, BERTopic [4] stands out, exploiting the capabilities of transformer-based language models like BERT to generate rich representations of documents and words. Guided versions of this model have also been developed to orient the model towards specific topics. In this report, the main focus will be on GuidedLDA [6] and GuidedBERTopic for the guided version, although other models using similar ideas also exist. In addition, recent models such as vONTSS [16], using an encoder-decoder architecture, represent a significant advance in topic modeling, introducing new methodologies for discovering and understanding latent topics in textual data.

4.2 Evaluation

Before we start gathering information on the main method used in this scenario, we need to know how to evaluate these models. As we are not doing classification, basic measures such as F1 score, accuracy, etc. cannot be used. Evaluation measures must be linked to the topics extracted by the model and/or their correlation with the true labels in the dataset. Without using these true labels, we can define two general metrics, namely *Topic Coherence* and *Topic Diversity*, and we have created a specific supervised metric to assess the correlation between the topics found by the models and the true labels in the dataset.

4.2.1 Topic Coherence (TC)

This measure assesses the extent to which a topic is "supported" by a text set (called a reference corpus) [13, 11]. Four types of coherence metrics were compared, namely C_v (Cohesion Coherence), C_{uci} (Exclusive Coherence), C_{npmi} (Normalized Pointwise Mutual Information) and U_{mass} (Unsupervised Mass Coherence). Before briefly explaining each type, let's define a few quantities:

- $P(w_i)$: probability assigned to a word w_i.
- $P(w_i, w_j)$: probability of words w_i and w_j occurring together.
- Pointwise Mutual Information defined as PMI $= \frac{P(w_i,w_j)}{P(w_i)P(w_j)}$.

4.2.1.1 Cohesion Coherence (C_v)

This metric emphasizes the coherence of words within the topic by calculating the average pairwise PMI for all word pairs within the topic. It is defined as:

$$C_v = \frac{2}{n(n-1)} \sum_{i=2}^{n} \sum_{j=1}^{i-1} \text{PMI}(w_i, w_j) \tag{1}$$

4.2.1.2 Exclusive Coherence (C_{uci})

The formula is the same as for C_v, but C_{uci} focuses more on the exclusivity of words within the topic, emphasizing the uniqueness of co-occurrences within the topic relative to the corpus. In practical terms, the difference may lie in the way probabilities are estimated and the specific context in which these measures are applied.

4.2.1.3 Normalized Pointwise Mutual Information (C_{npmi})

C_{npmi} calculates the normalized PMI, which is the PMI divided by the negative log probability of the word pair. This normalization helps to bring the value within a specific range and emphasize the strength of association between words.

$$C_{npmi} = \frac{2}{n(n-1)} \sum_{i=2}^{n} \sum_{j=1}^{i-1} \frac{\text{PMI}(w_i, w_j)}{-\log(P(w_i, w_j))} \tag{2}$$

4.2.1.4 Unsupervised Mass Coherence (U_{mass})

U_{mass} computes the coherence of a topic by comparing the co-occurrences of words within the topic against a background of all the words in the corpus. The addition of the smoothing term ϵ ensures that the logarithm is defined even for very small probabilities.

$$U_{mass} = \frac{2}{n(n-1)} \sum_{i=2}^{n} \sum_{j=1}^{i-1} \log \frac{P(w_i, w_j) + \epsilon}{P(w_j)} \tag{3}$$

4.2.2 Topic Diversity (TD)

This metric is defined as the percentage of unique words for all topics. The measure ranges from $[0, 1]$ where 0 indicates redundant topics and 1 indicates more varied topics. It is calculated by taking the first k words of each subject and calculating how many words are identical in different sets.

4.2.3 Supervised Correlation

In order to compare the correlation between the extracted topics and the actual labels in the datasets, the first N words for each topic/label can be used to define a metric that counts the number of similar words and provides a value between 0 (totally different topic) and 1 (same topic). This metric will be weighted by the number of each main words to take into account the number of words contained in the documents. Figure 4.2.3 shows a simple example of this metric. This process will be carried out for each real label and for each extracted topic. The result will be a $N \times N$ matrix of similarity scores. To avoid each extracted topic being associated with the same class and vice-versa, this task will not be performed iteratively. We could perform it iteratively to reduce computation time, since computing the assignment of an $N \times N$ matrix is an NP-hard problem. Instead, we decided to use the Hungarian algorithm [7], which can solve this problem in $O(n^3)$. This algorithm will be able to maximize the overall similarity of the assignment.

In the following sections, we explain and discuss the results of various topic modeling models. It should be noted that no hyperparameters were used that could improve the results. On the one hand, this is because it takes a long time to find the best hyperparameters for a model. On the other hand, this is not the aim of this report. We want to compare several methods in general, even if they can be improved for a given problem.

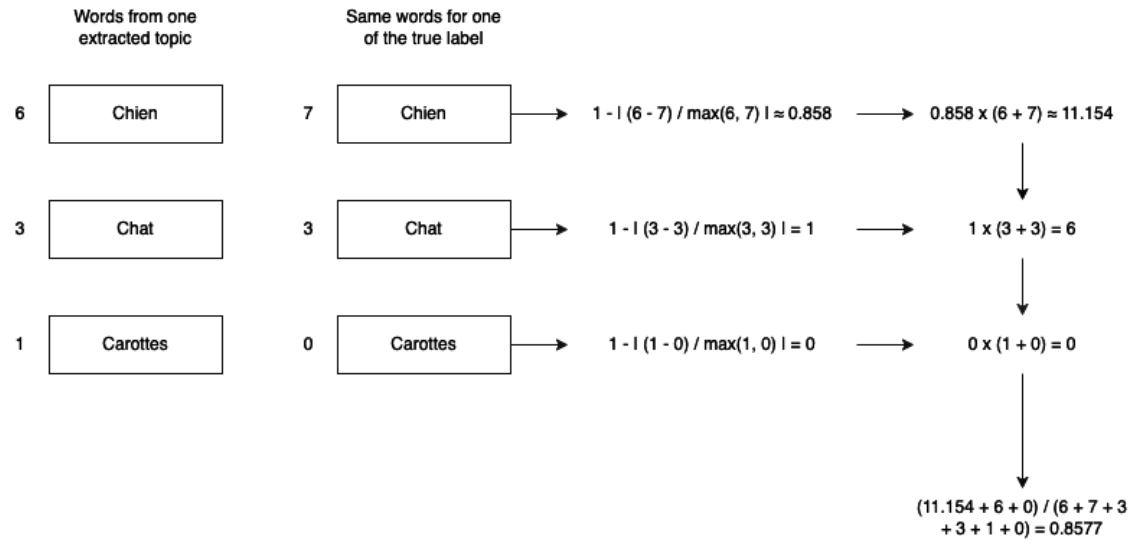

Figure 16: Example of the application of this weighted metric for a given extracted topic and a given true label.

4.3 Latent Dirichlet Allocation (LDA)

4.3.1 Theory

For this model to work, multiple assumptions are done [8, 1]:

- Each document is simply a collection of words or a ”bag of words” (BOW). Thus, word order and the grammatical role of words (subject, object, verbs, etc.) are not taken into account in the model.
- Documents with similar topics use similar word groups.
- Words that appear in at least 80%-90% of documents can be eliminated, without losing any information.
- The number of topics we want is specified in advance. This number is represented by the K parameter.
- Documents are probability distributions over latent topics, meaning that a given document will contain more words on a specific topic. The topics themselves are probability distributions over words. See figure 4.3.1 for a visual explanation of these two assumptions.

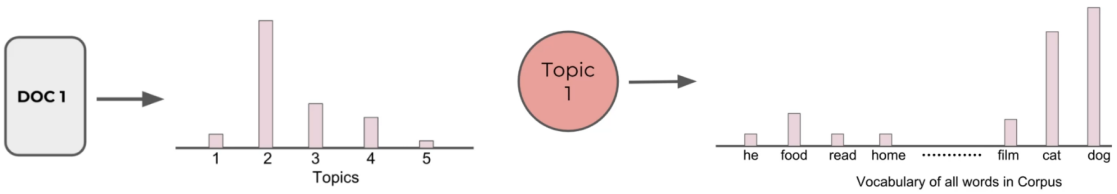

Figure 17: Visual explanation of the assumptions that documents and topics are probability distributions over latent topics and words respectively.

Once these assumptions have been made, the LDA algorithm works as follows:

1. For each document, randomly assign each word in the document to one of the K subjects.

2. For each document d and for each word w in the document, calculate:
   - $\mathbb{P}$(topic t | document d): the proportion of words in document d that are assigned to topic t.
   - $\mathbb{P}$(word w | topic t): the proportion of assignments to topic t over all documents that come from this word w. This tries to determine how many documents are in subject t because of the word w.
3. Update the probability of the word w belonging to the subject t, as

$$\mathbb{P}(\text{word } w \text{ with topic } t) = \mathbb{P}(\text{topic } t \mid \text{document } d) * \mathbb{P}(\text{word } w \mid \text{topic } t) \tag{4}$$

4. Repeat steps 1 to 3 until it converges.

4.3.2 Words in extracted topics

The results presented in this section are the results of applying the LDA model to the BBC News dataset. The first thing we wanted to show is the list of the most important words by topic and their level of importance. This list is presented in Figure 19. As LDA is random, a random state has been set in the model for these results. It can be seen that some topics are clearly linked to the actual label. Topic 0 can be associated with ”politics”, topic 1 with ”sports”, topic 2 with ”entertainment”, topic 4 with ”business” and topic 3, although more subtle than the others, can be associated with ”technology”. We need to bear in mind that these results are only obtained by specifying a random state, and that this assignment may not be generalizable. The question is: how can we achieve this ”automatic” detection when we are in scenario 1, i.e. when we have the documents and the topics in which to classify them? This question will be answered in Section 4.7.2.

The second result we can observe is the attribution of a topic directly to the words of a document. This is shown in Figure 18. Note that these are only the first 10 words of each document. This means that even if no words from the assigned topic appear, there may be nothing wrong with the model. This may be because all the other unseen words come from that topic.

Words with colors indicating the associated topic

Doc 0	hit	shelf	combine	medium	player	phone	gaming	gadget	sale	price	|	**Topic 2**
Doc 1	bid	hope	join	host	apply	host	tournament	aim	rugby	traditional	|	**Topic 1**
Doc 2	lord	wrong	detainee	straw	straw	attack	decision	high	court	detain	|	**Topic 3**
Doc 3	leak	answer	minister	explain	budget	detail	print	newspaper	hour	speech	|	**Topic 0**
Doc 4	delight	manager	pay	tribute	goal	striker	beat	win	talk	interested	|	**Topic 1**
Doc 5	ready	information	act	thousand	public	body	ill	prepare	freedom	information	|	**Topic 0**
Doc 6	bank	chief	fund	director	run	central	chairman	banking	agree	subject	|	**Topic 4**
Doc 7	takeover	win	takeover	battle	phone	firm	report	firm	expect	seal	|	**Topic 3**
Doc 8	tip	world	player	shortlist	newly	crown	european	personality	field	man	|	**Topic 1**
Doc 9	set	return	career	threaten	injury	event	return	action	statement	website	|	**Topic 1**

Figure 18: The first 10 words of the first 10 documents with their associated topics.

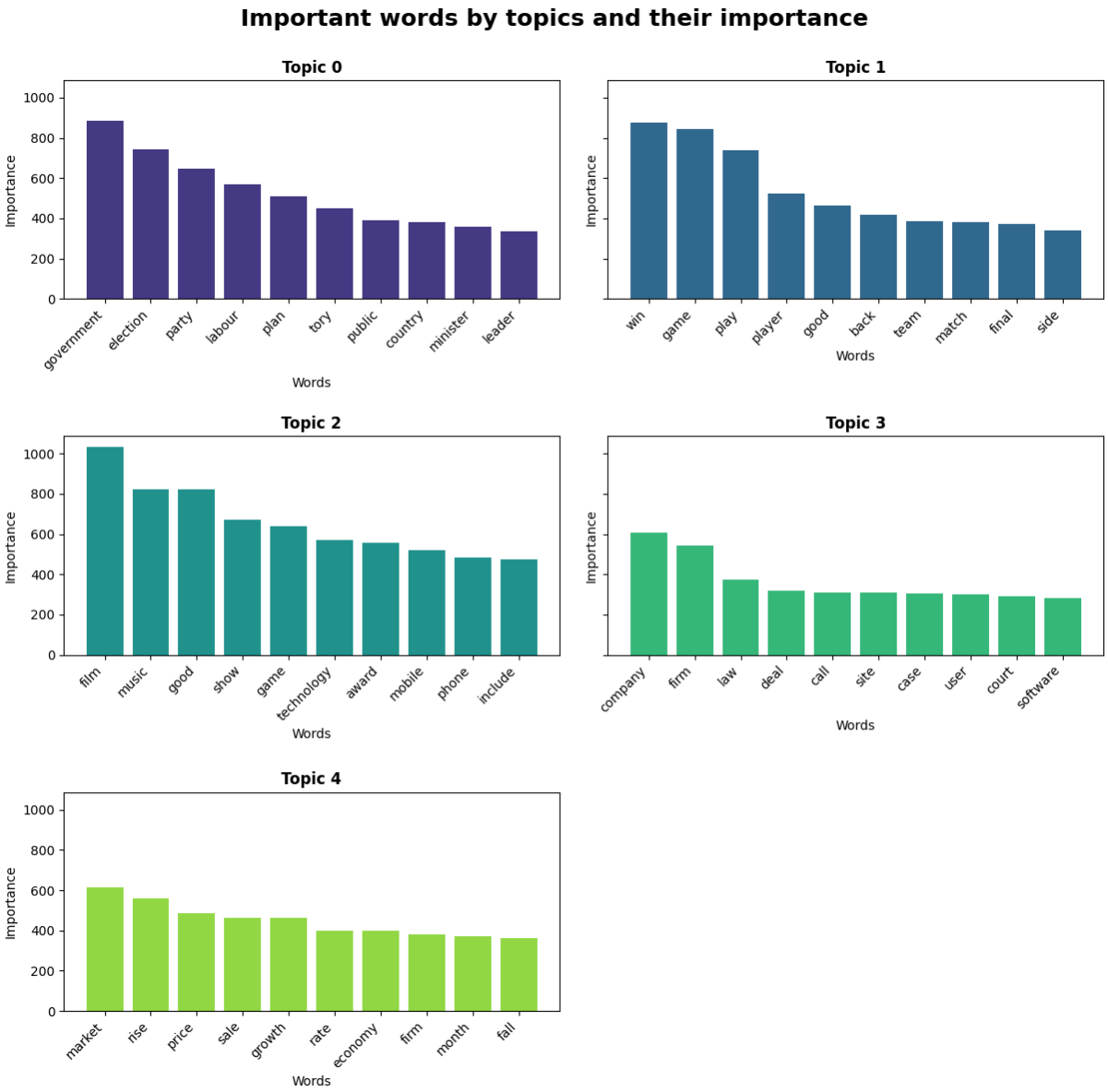

Figure 19: Important words by topic by applying the LDA method to the BBC News dataset.

4.3.3 Randomness

Since LDA is random, one way of evaluating the model is to naively run the model several times and average the results. This will be done in many other experiments, as one of the main problems with topic modeling is that the models are random. Future work should really try to address this problem and find another, less naive, solution. The aim was to find out whether the scores were always close to the mean, or whether the variance was high. This is shown in Figure 20. The number of iterations used is fixed at 10, and all metrics remain close to the mean, meaning that this solution can be used. The fact that some boxplots are larger than others (e.g. for u_{mass} and u_{uci}) is simply explained by the fact that, unlike the other metrics, they are not bounded between 0 and 1.

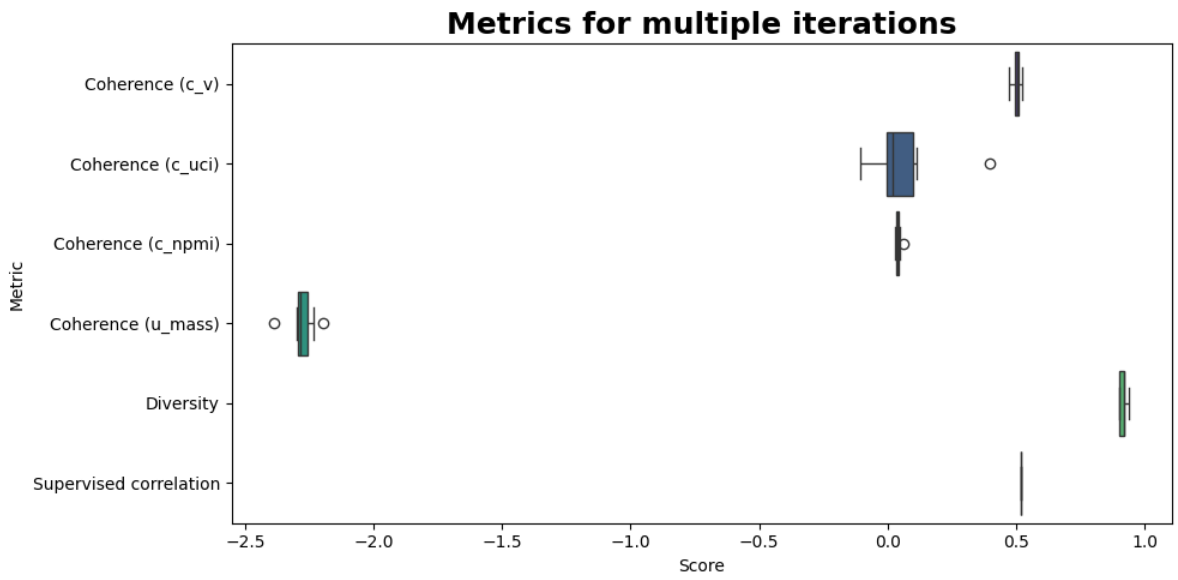

Figure 20: Box plots for all scores obtained by running the LDA model for 10 iterations.

4.4 BERTopic

4.4.1 Theory

BERTopic is another topic modeling model that uses SBERT to extract topics. It works as follows [12, 4]:

1. Convert documents into embeddings using any sentence transformer. The default transformer is *Sentence-BERT*.
2. Reduce the dimensionality of the embeddings using the *Uniform Manifold Approximation and Projection (UMAP)* algorithm. The main problem is that UMAP is random, which makes BERTopic random.
3. Group the reduced embeddings into clusters using an algorithm called *HDBSCAN (Hierarchical DBSCAN)*.
4. Extract topics from each cluster using their *c-TF-IDF scores*.

    - *Term Frequency (TF)*: how frequently a term/word appears in a document, calculated as:

$$TF = \frac{\text{Number of times term appears in a document}}{\text{Total number of terms in the document}} \tag{5}$$

    - *(Class-Based Term Frequency (CTF)*: for a term within a specific class or category is the number of times that term appears in documents belonging to that class. It measures how frequent a term is within documents of a particular class.)

- *Inverse Document Frequency (IDF)*: Inverse document frequency measures how unique or rare a term is across the entire corpus. The purpose of IDF is to give more weight to terms that are relatively rare in the corpus, as they are likely to carry more meaningful information. It is calculated as:

$$IDF = \log \frac{\text{Total number of documents in the corpus}}{\text{Number of documents containing the term}} \tag{6}$$

- *TF-IDF Score*: It quantifies how important a term is within a specific document while considering its rarity across the entire corpus. It is calculated as:

$$\text{TF-IDF} = TF * IDF \tag{7}$$

It should be noted that, unlike LDA, BERTopic produces another topic, which is topic -1. This topic is supposed to contain all documents that do not match any other extracted topic, i.e. they may be potential outliers. We need to take this into account when comparing with the real classes and decide whether we want this topic to be considered as an extracted topic or as an outlier detection. Considering it as a topic allows us to take into account all the documents and not just some of them, but it also means that a topic can be very different from the real topics. Conversely, considering it as a bin of outliers has the advantage of improving the correlation metric, but we only consider a part of the documents, which can be specific in certain situations (e.g. for medical records where all documents must be taken into account).

4.4.2 Words in extracted topics

Similar to the LDA section, the results come from applying the model to the BBC News dataset, also by defining a random state in the model. First, we wanted to show the most important words per topic and their level of importance. This is shown in Figure 22. It can be seen that some topics are also linked to the current label. Topic -1 contains all documents that cannot be classified in any other topic. We can clearly make the following assignments: topic 0 with the class ”entertainment”, topic 1 with ”politics”, topic 2 with ”business” and topic 3 with ”sport”. Topic 4, on the other hand, cannot be assigned to the ”technology” class. However, it should be borne in mind that these results are only valid for this specified random state.

4.4.3 Similarity matrix

Another thing we wanted to see was the diversity score. It was important to see how similar the topics were to each other. This is shown in Figure 21. It can be seen that the topics are somewhat similar, which can be explained by the fact that there are so few of them. But in general, they are different.

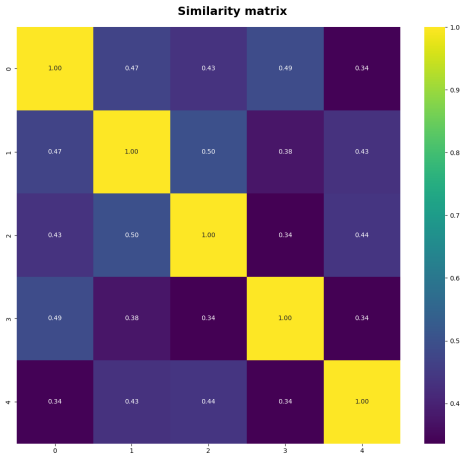

Figure 21: Topic similarity matrix obtained by applying BERTopic to the BBC News dataset.

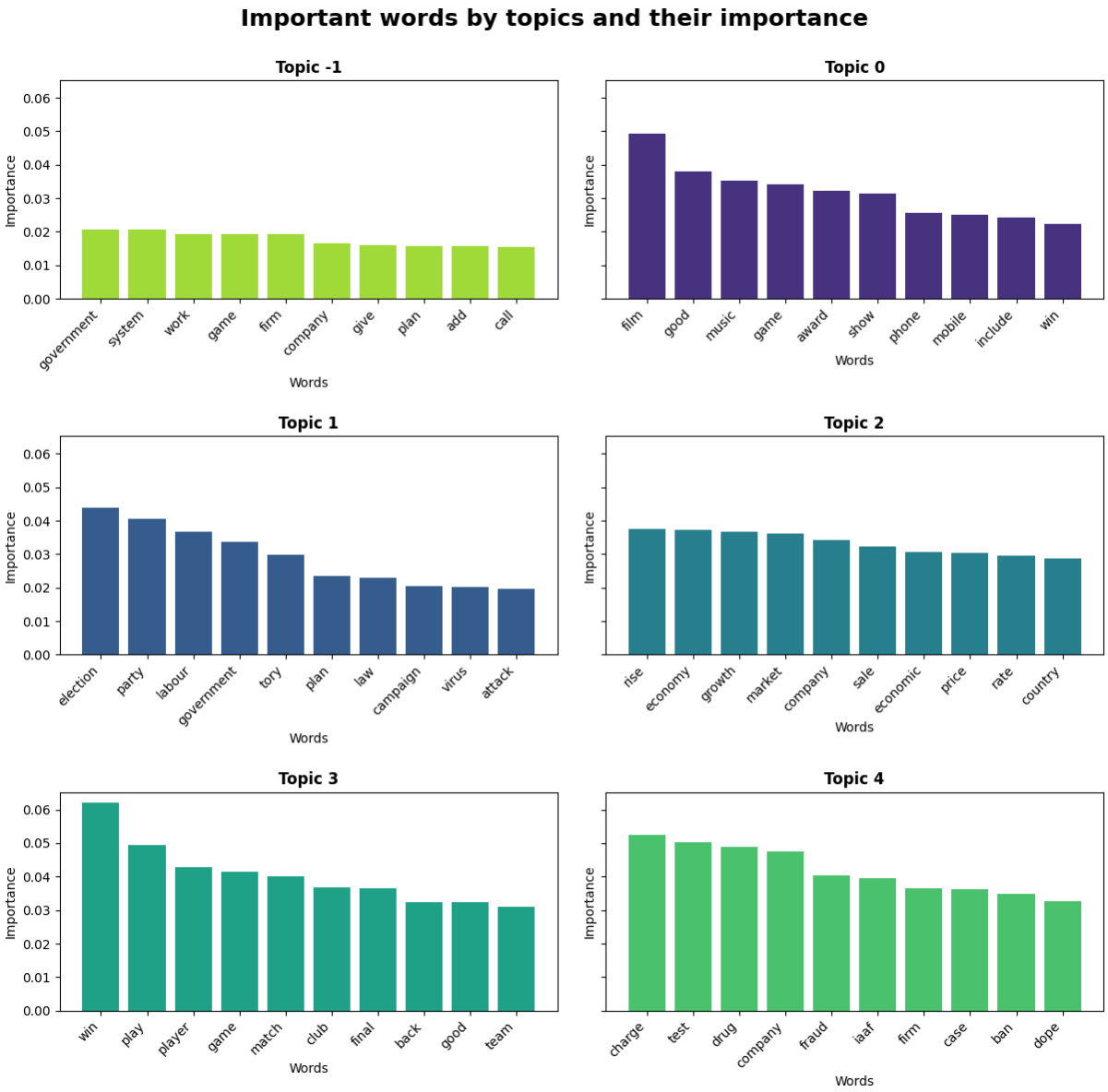

Figure 22: Important words by topic by applying BERTopic to the BBC News dataset.

4.4.3.1 Randomness

As with LDA, we wanted to see if the method of taking the average of scores from several iterations could handle the randomness of the model. This is shown in Figure 23. The number of iterations is also fixed at 10. It can be seen that, in the same way as for LDA, this method can be used to counteract the randomness of the model. Indeed, all iterations produce values closer to the mean, i.e. the variance is not high. This method will therefore be used in the remainder of this report.

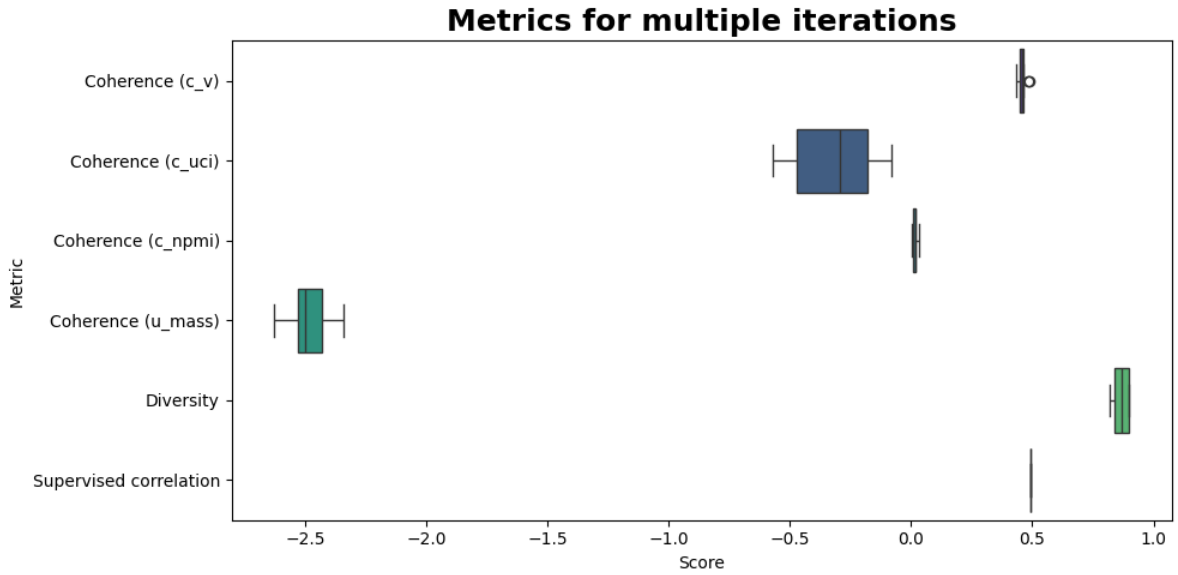

Figure 23: Box plots for all scores obtained by running the BERTopic model for 10 iterations.

4.5 Guided Topic Modeling

Guided Topic Modeling [6] is a method of discovering specific topics or themes within a dataset of documents. Unlike traditional techniques, it allows users to provide clues or hints to guide the model towards particular areas of interest. Users can enter relevant words or phrases, which act as cues for the model to prioritize certain topics during analysis. By incorporating these hints, the resulting topics are more focused and aligned with the user's expectations, making them easier to interpret.

The aim of this section is to compare the models with and without the guided version and to see how useful it can be. For this section, we have specified a random state in the different models in order to visualize the result. It is assumed that, in general, the result will be quite similar for other random states. The keywords used are those defined in Section 2.1.

4.5.1 Guided LDA

The purpose of guided topic modeling is to guide the model in the discovery of certain topics. Therefore, what we naturally wanted to see in this section was to compare the different topics obtained with the LDA models with and without the guided version. To do this, we calculate the most important words for both models on the BBC News dataset and plot them to see if there is any visual similarity. In addition, we use the supervised correlation metric to see if it improves. As expected, the guided version is better than the basic version, and we can clearly see the assignment of topics to the 5 real classes. Moreover, it can be seen that the supervised correlation metric improves from 0.52 to 0.59.

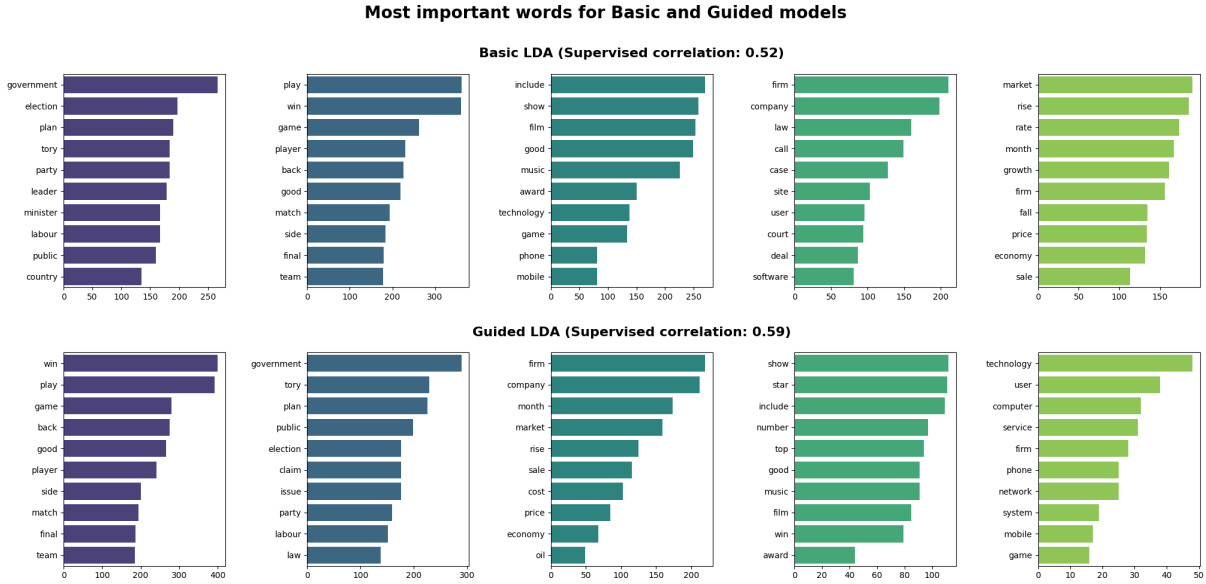

Figure 24: The most important words found by the basic model compared with those found by the guided version.

4.5.2 Guided BERTopic

As with guided LDA, we wanted to see how close the clusters found by the guided version of the BERTopic model were to the true clusters, i.e. those manually labeled in the dataset. The same methodology as before was applied, but instead of plotting only the words, we plot the embeddings found by the SBERT model. The result can be seen in Figure 26. We only take into account clusters other than -1, i.e. those containing possible outliers. This result is interesting because, as with LDA, the clusters appear to be closer to the true clusters found. Moreover, the supervised correlation metric improves considerably. One hypothesis that follows from these observations is that these guided topic modeling methods could be used for clustering as in the first scenario. This point will be detailed in Section 4.7.2.

4.6 vONTSS

This section has been added to provide a brief overview of a recent model created in July 2023. This model is a combination of the two scenarios, similarly to guided topic modeling. Indeed, vONTSS [16] is a semi-supervised topic modeling method that takes a few keywords as input and, instead of classifying documents into topics defined by these keywords as in the first scenario, it generates topics based on these keywords as in the second scenario. As this method generates topics and therefore meets the definition of topic modeling as in the second scenario, it is added to this section rather than the first.

4.6.1 Theory

The architecture of this model is shown in Figure 25. In this section, we will explain how it works, but not why. We leave that for future work. First, the encoder network ϕ transforms the document representation X_d into a latent vector generated by the von Mises-Fisher distribution and generates a sample η_d. A temperature function τ is applied to this sample with a softmax function to obtain a probabilistic distribution of topics z_d. Finally, the decoder uses a modified topic-word matrix E to reconstruct the BoW representation of the document X_d.

In this architecture, L_{recon} represents the reconstruction loss, L_{KL} represents the KL divergence and L_{OT} represents the optimal transport loss. Explanations of the different losses and how this architecture is trained given a set of keywords S associated with topics T can be found in the article. Compared with other architectures, they have introduced a temperature function to modify the radius of the vMF distribution.

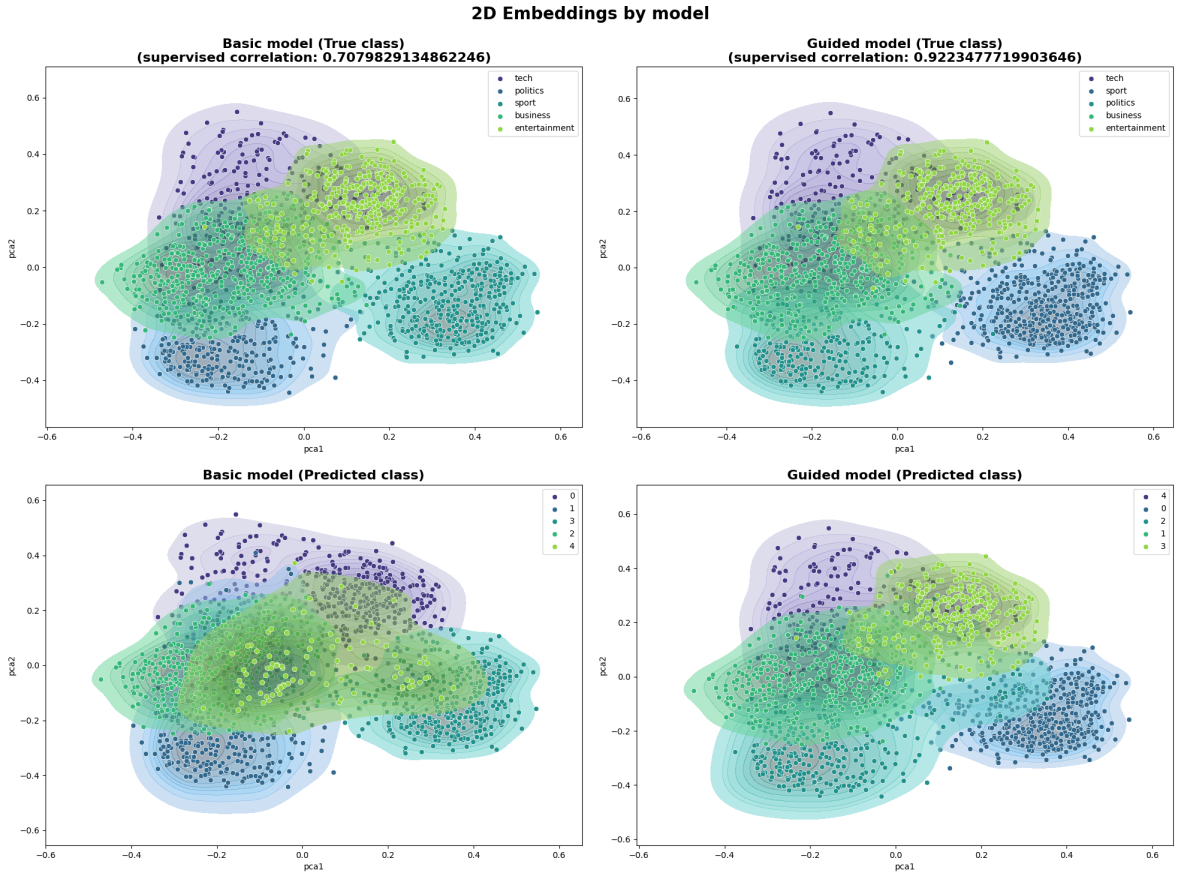

Figure 26: Clusters found by the basic model versus those found with the guided version.

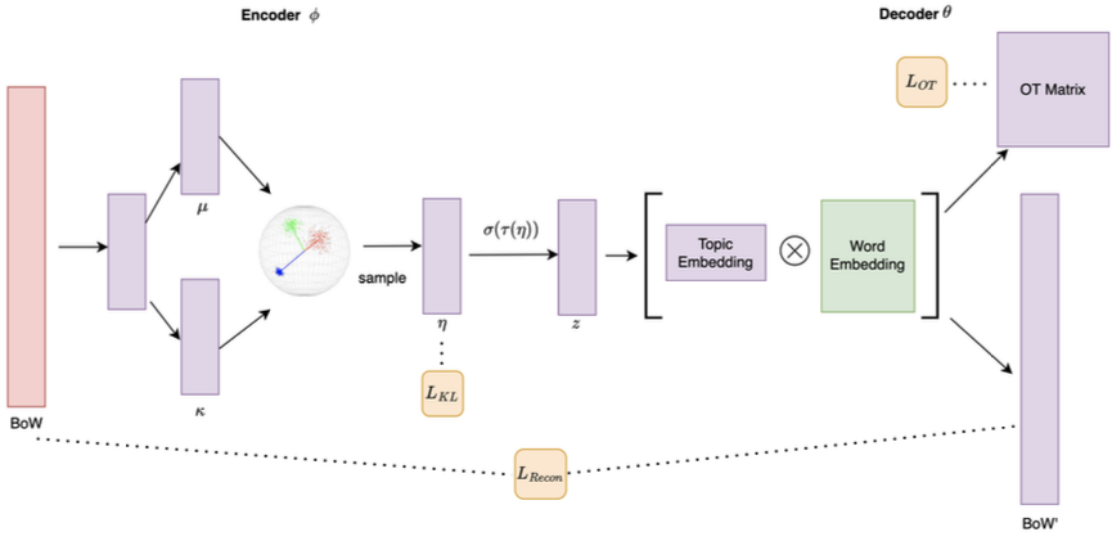

Figure 25: Architecture of the vONTSS model.

4.6.2 Words in extracted topics

Similar to the other sections on topic modeling, we simply wanted to see the most important words in each topic extracted by vONTSS. This is shown in Figure 27. One of the main problems with this model is that it fails to always assign documents to a given topic. This is because the topics extracted by vONTSS are sometimes very different from the actual topics, as shown in the graph below. Consequently, some documents cannot be classified within the extracted topics. To solve this problem, a naive method is to run the model several times until all topics are assigned at least one document. But this is problematic given that we also need to iterate to limit randomness by taking the mean of the results. This is therefore extremely computationally expensive. Future work should try to use a more practical method, or to understand why this problem occurs with vONTSS. Compared with other topic modeling models, it can be seen that the topics are really different from the ”real” topics. Topics 0 and 4 may be associated with ”sport” and ”technology”, but the others are totally different from ”business”, ”politics” and ”entertainment”. Future work should try to see if this has an impact on model performance, or if it is simply due to the random state chosen.

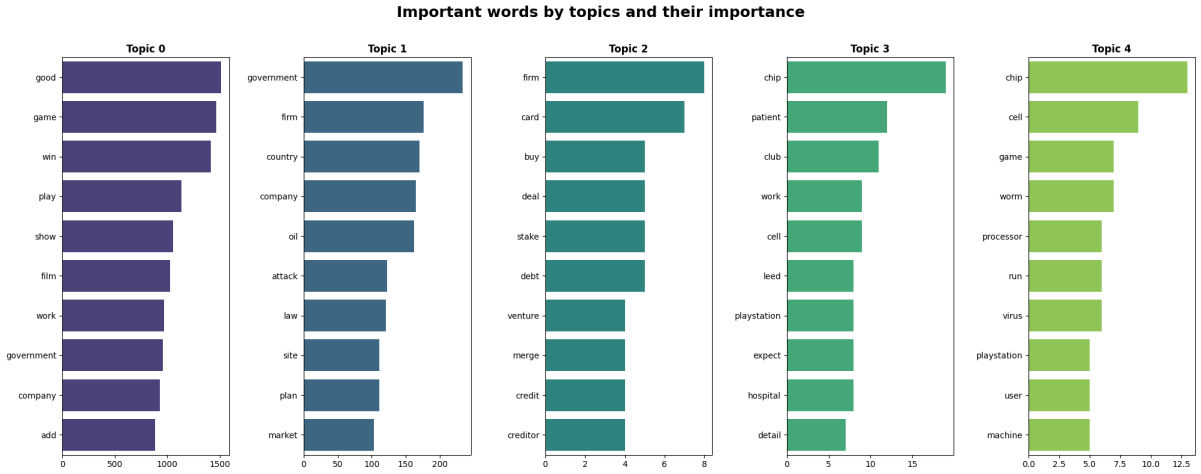

Figure 27: Important words by topic by applying vONTSS to the BBC News dataset.

4.6.2.1 Randomness

As far as randomness is concerned, we would like to see whether the methodology of taking the mean is always close to the mean, as we did in the LDA and BERTopic sections. We could have assumed that this is the case and used this methodology for the rest of this report, but the randomness issue explained above is important for checking whether this method can be used or not. The randomness of topic modeling models remains an important issue and should be one of the main questions to be addressed in future work. The result can be seen in Figure 28. As it can be seen, the metric seems to be as variant as for the other models, which means that this method can also be used to handle the randomness of the model.

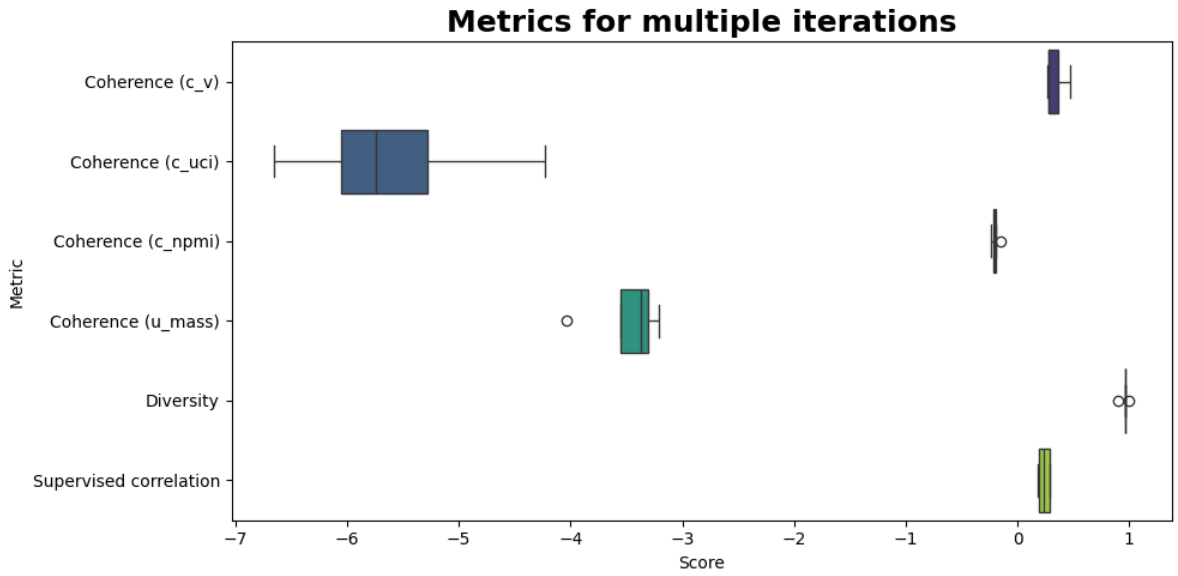

Figure 28: Box plots for all scores obtained by running the BERTopic model for 10 iterations.

This model is still in an experimental state, and is really difficult to use due to its highly random nature and the lack of appropriate documentation. Its randomness means that we have to perform many iterations to find a model that assigns each topic to at least one document, and for each of these iterations we also have to run it for several iterations to average the results. The computation time is therefore enormous. Because of these problems, we decided not to use this model in the comparison. Future work should attempt to readapt this model when it becomes more advanced. Nevertheless, as this section has shown, this model looks promising due to the use of a new architecture, and should be explored in more detail.

4.7 Comparison

4.7.1 Performances

The comparison with the same number of topics as the true labels is shown in Table 7. It can be seen that ... TODO

Dataset	Models	Coherence (C_v) ↗	Coherence (c_uci) ↗	Coherence (c_npmi) ↗	Coherence (u_mass) ↘	Diversity ↗	Supervised correlation ↗
BBC News	LDA	0.493955	0.279265	0.040366	-2.368924	0.920000	0.760189
	BERTopic	0.433929	-0.419129	0.002830	-2.499619	0.840000	0.627419
	GuidedLDA	0.498162	0.082306	0.037583	-2.235177	0.920000	0.588504
	GuidedBERTopic	0.490397	-0.188388	0.026225	-2.232617	0.883333	0.922348
20NewsGroup	LDA	0.603940	0.673476	0.096662	-1.883472	0.761000	0.401317
	BERTopic	0.472277	0.125097	0.053755	-2.306378	0.803810	0.268840
	GuidedLDA	0.462860	-0.205873	0.025873	-2.436411	0.985000	0.361511
	GuidedBERTopic	0.528895	0.203075	0.071613	-2.263779	0.814286	0.318357
DBLP	LDA	0	0	0	0	0	0
	BERTopic	0	0	0	0	0	0
	GuidedLDA	0	0	0	0	0	0
	GuidedBERTopic	0	0	0	0	0	0
M10	LDA	0	0	0	0	0	0
	BERTopic	0	0	0	0	0	0
	GuidedLDA	0	0	0	0	0	0
	GuidedBERTopic	0	0	0	0	0	0

Table 7: Metrics obtained by running all models on all datasets and specifying the same number of topics as for the true labels.

4.7.2 Application to Scenario 1

As we have seen in previous comparisons throughout the report, some models perform well when it comes to finding topics similar to the true topics, i.e. those that have been manually labeled in the dataset. Hence the idea of using the topic modeling models of Scenario 2 to perform the task of Scenario 1. To assign each extracted topic to a true topic, we will calculate the similarity between the keywords defining each topic (extracted and true) and assign the most similar set exactly as we did for the supervised correlation metric, with the Hungarian algorithm, which can solve the assignment problem in $O(n^3)$. This algorithm will be able to maximize the overall similarity of the assignment.

Figures 29 and 30 show topic assignment using LDA and BERTopic on the BBC News dataset. We use principal component analysis to represent embeddings in two dimensions. For BERTopic, we considered topic -1 as a topic and not as outliers, since we have not yet made predictions for these documents. One thing that could be done is to take these potential outliers into account when calculating F1 scores, but we will leave that for future work. It can be seen that the assignment algorithm is capable of finding similar clusters, which is very promising. It can also be seen that BERTopic extracts topics as similar as with LDA, but the -1 class is obviously less accurate. Nevertheless, this cluster still seems to cover most of the true class.

Using this assignment strategy, we compare the F1 score obtained using several models on the datasets and compare it to the random predictor and to Lbl2Vec. As a reminder, the random predictor simply assigns a class at random. Consequently, the F1 score of this predictor is equal to 1 divided by the number of classes. As we can see in Figure 31, the LDA model present often really good performances. One possible intuition for this is that LDA works directly with the words contained in the documents. While other transformer-based model have been trained in a global way, meaning that they perhaps try to generalize too much. Interestingly, its guided version is often worse, which is interesting since its supposed to guide the model into finding similar topic. We have to keep in mind that more the extracted topic are similar to the topic, obviously it will help to have a higher F1 score (since if we have the same topic we will have the same prediction). Nevertheless, it could be possible that multiple topics overlap,

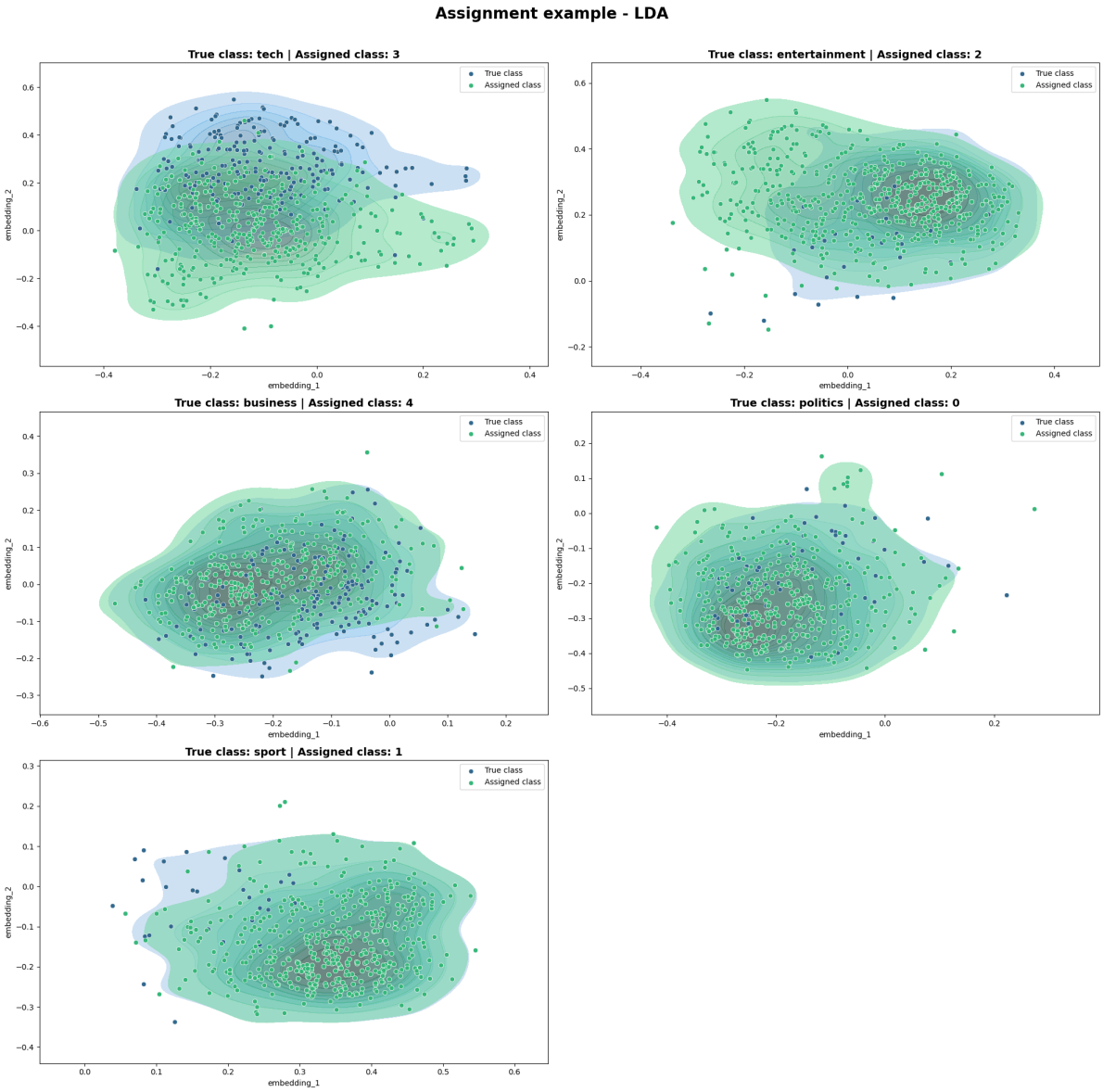

Figure 29: Example of assignment of topics using LDA on the BBC News dataset using the Hungarian algorithm.

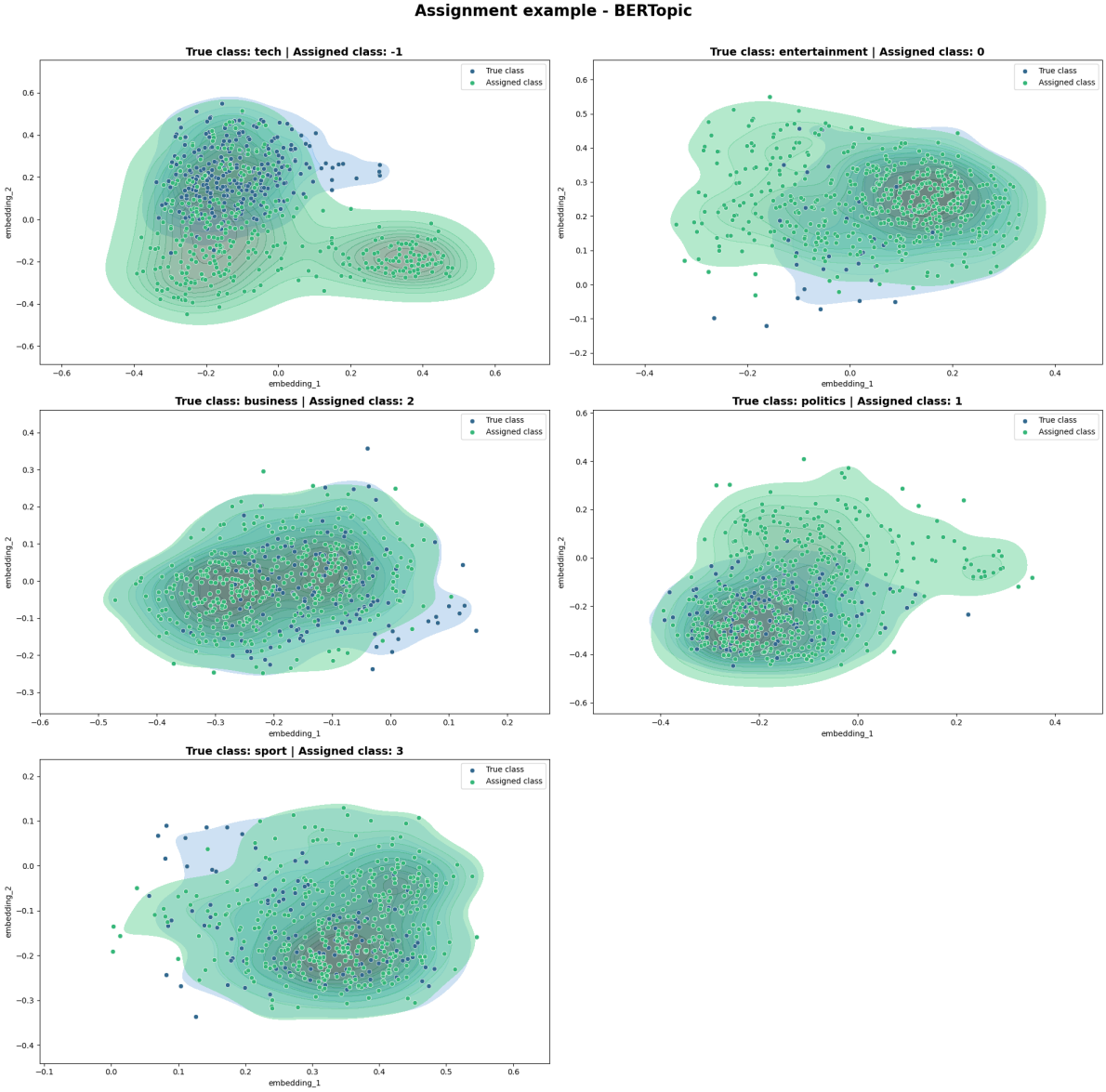

Figure 30: Example of assignment of topics using BERTopic on the BBC News dataset using the Hungarian algorithm.

and therefore even if the topics are similar, the F1 score will not be good. Hopefully, the model are almost always more performant than the random predictor, except for DBLP which is considered as a ”difficult” dataset. The bad performances on 20NewsGroup could be explained by the fact that a lot of true topic overlap (i.e. we have multiple thing for ”computer”, ”science”, etc.). It is the same for DBLP. The four class are all topic about specific domain in computer science. A lot of words from this topic overlap. It seems like BERTopic and its guided version are better when the true topic are more distinct. This is in fact a good thing since BERTopic try to create topic as diversified as possible. The assignment is therefore possibly difficult since the topics are different. Future work should try to (1) find a better way to compute the assignment or to compute the similarity between two topics and (2) put the F1 score in correlation with the assignment score (.i.e. the supervised correlation metric with the F1 score) and see how one affect the other. This last question is truely interesting if we put this chart in correlation with the supervised correlation metrics that we obtained in Table 7. It seems that ... TODO

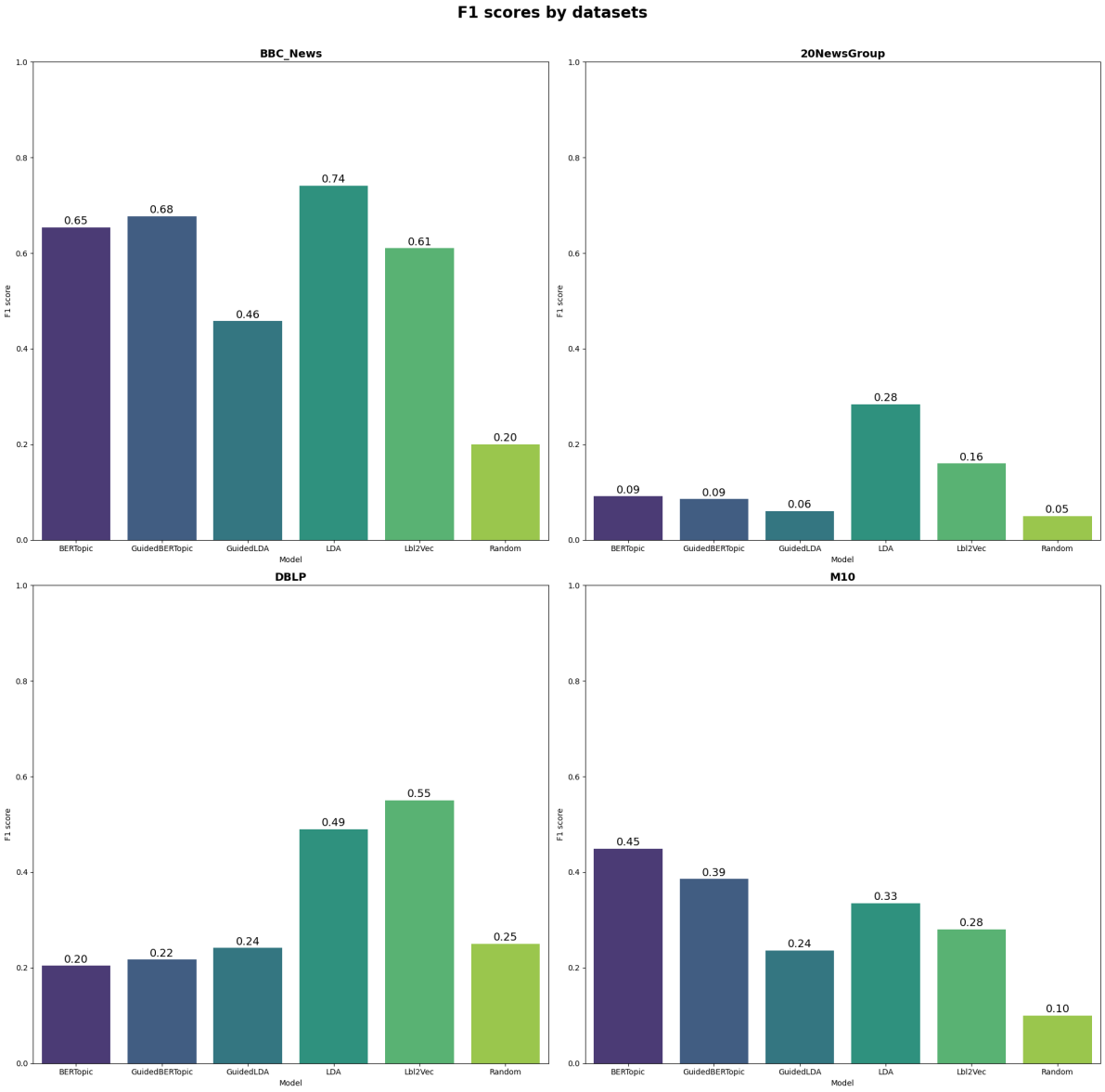

Figure 31: F1 scores obtained by executing every models to all the datasets and by using the assignment methodology explained above.

5 Future work

This section provides a summary of all the ideas for future work presented in this report. It is also supplemented by contributions, not from the report, but from the questions most frequently asked throughout the project. The main questions are:

- Is it better to have a large number of shorter documents or a few larger ones? Does this depend on the model used? And if so, can we improve the models by increasing/reducing the data?
- How can we solve the problem of randomness in topic modeling models in general? In fact, we need to average results over several iterations, but this is not viable for one product. How can we make the models deterministic?
- How do we choose whether to take topic -1 (i.e. irrelevant documents) as outliers or as a used topic? And what is the impact on model performance?
- Is there a way to fine-tune LLMs to the problem, instead of using the pre-trained version? And does this make sense?
- For Lbl2Vec:
  - Is LOF really useful for performance?
  - Only *all-minilm-l6-v2* has been used, could we use other LLMs?
  - How can we efficiently find the best keywords to pass on to the model?
  - Are there models other than Lbl2Vec that use LLMs in a different way to perform unsupervised text classification?
  - Instead of considering the words most similar to predefined keywords directly in documents, would it be possible to consider the vocabulary of an entire language?
  - Possibly review the part of the code on Doc2Vec. Are there any errors causing this poor performance?
  - Plot the embeddings for the defined keywords for both embedding models. Do they overlap?
- For BERTopic:
  - Are there other transformers more suitable for this problem?
  - Can other methods of dimensionality reduction be used?
  - Can other clustering methods be used?
  - How to deal with the extracted topic concerning irrelevant words (-1)?
- For vONTSS:
  - Why does this model work? This report explains how, not why.
  - How does this model perform? It was not advanced enough to be tested in this report.
  - How can we solve the problem that some documents are never assigned to any topic?
  - The decoder seems to generate different topics from the real ones, even when given predefined keywords. How can we guide the decoder to generate more similar results?
- Is there a better way of calculating similarity between topics than cosine similarity between the words defining the topics?
- Is there a more appropriate way of calculating the assignment between two lists of topics in order to maximize a given score (i.e. similarity, F1, ...)?
- How does the similarity score between two lists of topics affect the F1 score obtained after an assignment? And vice versa?

6 Conclusion

In this report, two scenarios have been considered, depending on the input data. One in which we receive documents (i.e. a corpus of text) and the topics into which we need to classify the documents. The second scenario involves receiving documents only, and extracting topics directly from the words contained in the document. In the first scenario, a lot of time was spent analyzing Lbl2Vec, a model that uses predefined keyword embeddings representing classes and document embeddings to calculate the distance between the two, and thus assign each document to the topic with the smallest distance. This model was promising, but a problem in the internal model made the results of its Doc2Vec version rather disturbing. Fortunately, its transformer-based version produces better results.

In the second scenario, we explored several well-known topic models such as LDA and recently BERTopic, which harness the power of transformers to find topics based on word embeddings. Next, we explored their guided version, meant to guide the model to find the desired topics. We also briefly explored a very recent topic, vONTSS, which seems to be at too early an experimental stage to be used effectively. Next, we looked at how all the models compare with each other. Do they deliver coherent results? Are the topics diversified? Do extracted topics seem similar to real ones? This section provides useful results and can be seen as the central point of this work. Finally, we wanted to see whether these models could be used to fulfil the same task as in scenario 1.

If one sentence could sum up this work, it would be: ”Topic modeling is random”. Most models are random, and it seems that no model is really better than another. There is no specific answer, and each model can be powerful depending on the problem, the datasets and the way the model is trained. This is a problem because we are working in an unsupervised setting, and we would like our models to be good in general i.e. regardless of the problem. One observation that can be made is that transformers are excellent, but it seems that focusing directly on document content should be a better route (e.g. as LDA). Indeed, transformers are trained to understand language in a general way, and therefore they could transfer this property to the model, rather than finding specifics directly in documents. AI in general is often seen as a black box, and it does not appear that this is any different for topic modeling. Future work should avoid focusing on comparing models, but rather on how a model differs when faced with a specific problem, with specific data. Consequently, there is still plenty of room for improvement, and we hope this report will help future readers.

References

[1] H. Bansal. Latent dirichlet allocation - analytics vidhya - medium. 12 2021.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 3 2003.

[3] A. Dhuriya. What is Topic Modeling? - Analytics Vidhya - Medium. 12 2021.

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

[5] I. Hendrickx. Topic modelling only works if you have the right document collection.

[6] J. Jagarlamudi, H. Daumé III, and R. Udupa. Incorporating lexical priors into topic models. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 204–213, 2012.

[7] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 3 1955.

[8] R. Kulshrestha. A beginner's guide to latent dirichlet allocation(lda). 12 2021.

[9] T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse Processes*, 25(2-3):259–284, 1 1998.

[10] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 10 1999.

[11] J. Pedro. Understanding topic coherence Measures - towards data science. 1 2022.

[12] Priyanka. Understanding bertopic intuitively - level up coding. 2 2023.

[13] M. Röder, A. Both, and A. 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.

[14] T. Schopf, D. Braun, and F. Matthes. Lbl2vec: An embedding-based approach for unsupervised document retrieval on predefined topics. In *Proceedings of the 17th International Conference on Web Information Systems and Technologies*. SCITEPRESS - Science and Technology Publications, 2021.

[15] D. Stammbach and E. Ash. Docscan: Unsupervised text classification via learning from neighbors, 2022.

[16] W. Xu, X. Jiang, S. S. H. Rao, F. Iannacci, and J. Zhao. vONTSS: vMF based semi-supervised neural topic modeling with optimal transport. In *Findings of the Association for Computational Linguistics: ACL 2023*. Association for Computational Linguistics, 2023.