

Topic Modeling/Clustering

Clustering and extracting topics from a set of documents

Yorick Estievenart



AI Center of Excellence – University Incubator
University of Mons
Sweden

October 30, 2023

Contents

1	Introduction	2
2	Datasets	3
2.1	Keywords	3
2.2	Distribution of classes	4
2.3	Distribution of documents lengths	4
2.4	Most frequent words	5
2.5	Distribution of words occurrences	5
3	Topic Clustering	7
3.1	Lbl2Vec and Lbl2TransformerVec	7
3.1.1	Theory	7
3.1.2	Embeddings models	9
3.1.3	Doc2Vec - Unknown keywords problem	10
3.1.4	Embeddings	10
3.1.5	Performances	10
3.1.6	General performances	13
3.1.7	Assignment confidence	14
4	Topic Modeling	16
4.1	Evaluation	16
4.1.1	Topic Coherence (TC)	16
4.1.2	Topic Diversity (TD)	17
4.1.3	Supervised Correlation	17
4.2	Latent Dirichlet Allocation (LDA)	18
4.2.1	Theory	18
4.2.2	Results	19
4.3	BERTopic	21
4.3.1	Theory	21
4.3.2	Results	22
4.4	Guided Topic Modeling	25
4.4.1	Guided LDA	25
4.4.2	Guided BERTopic	26
4.5	vONTSS	27
4.5.1	Theory	27
4.5.2	Results	28
4.6	Comparison	28
4.6.1	Performances	28
4.6.2	Transformation to topic modeling	29
5	Application	30
5.1	Dataset	30
5.2	Topic Clustering	30
5.3	Topic Modeling	30
6	Future work	31
6.1	Future work	31
7	Conclusion	32

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/clustering to help us divide the data into clusters, and then label the data using these clusters as classes¹. Topic modeling is defined as a statistical and probabilistic technique used to automatically identify topics or themes in a collection of documents. Whereas Topic Clustering is defined as a text analysis technique used to group similar documents or textual data according to their content.

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? And how can this help us to label a given dataset?** It will be divided into two parts. First, we will focus on the scenario in which the input data are documents and classes (represented by keywords) into which we want to classify the documents i.e. Topic Clustering. The second scenario is one in which the input data are documents only. We need to extract classes for each group of documents i.e. Topic Modeling.

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

2 Datasets

Before researching and applying models in an attempt to solve data labeling problems, we need data. Table 1 shows the following reference datasets that will be used in 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 manually extracted keywords 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: Datasets extracted from OCTIS and used throughout the project.

2.1 Keywords

The keywords used for each class in each dataset are shown in tables 3 and 2. **Why did we replace the original heading name with something more legible?** Because 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".

Topic	Topic Index	Keywords	Topic	Topic Index	Keywords
soc.religion.christian	0	religion christianism	2	0	archaeology
rec.sport.hockey	1	sport hockey	3	1	computer science
sci.space	2	science space	4	2	financial economics
comp.os.ms-windows.misc	3	computer operating systems microsoft windows	0	3	agriculture
comp.graphics	4	computer graphics	7	4	petroleum chemistry
comp.sys.mac.hardware	5	computer systems mac hardware	9	5	social science
rec.motorcycles	6	motorcycles	1	6	biology
comp.sys.ibm.pc.hardware	7	computer systems ibm pc hardware	8	7	physics
talk.politics.guns	8	politics guns	6	8	material science
sci.electronics	9	science electronics	5	9	industrial engineering
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			

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

Topic	Topic Index	Keywords	Topic	Topic Index	Keywords
business	0	business	2	0	computer vision
sport	1	sport	0	1	database
entertainment	2	entertainment	3	2	data mining
tech	3	technology	1	3	artificial intelligence
politics	4	politics			

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

2.2 Distribution of classes

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 results on the other datasets.

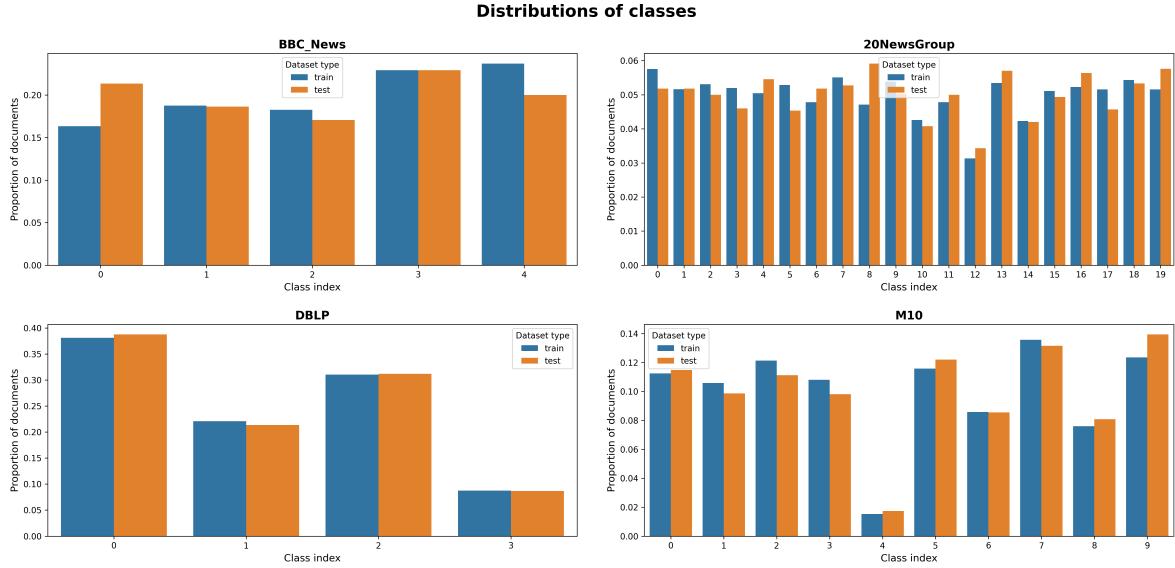


Figure 1: Class distribution for all datasets.

2.3 Distribution of documents lengths

Figure 2.3 shows the document length distribution for the datasets. The DBLP and M10 datasets contain shorter documents, which raises 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.

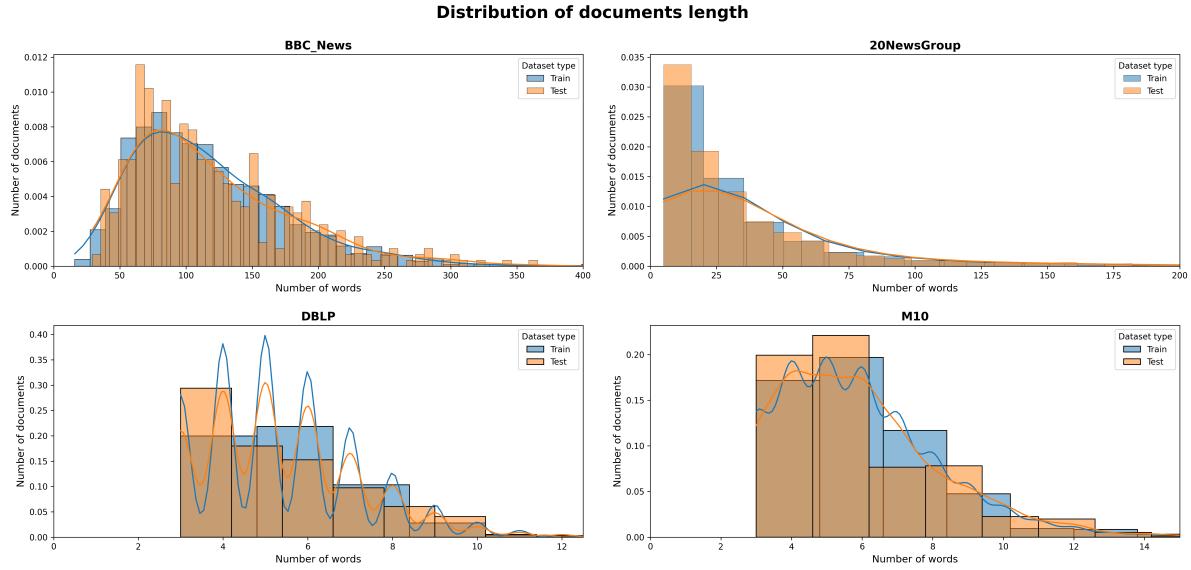


Figure 2: Document length distribution for all datasets.

2.4 Most frequent words

The 10 most frequent words for each dataset are shown in Figure 2.4. This could be even more useful if we had a visualization of the most frequent words directly by class. This will be done as part of some experiments.

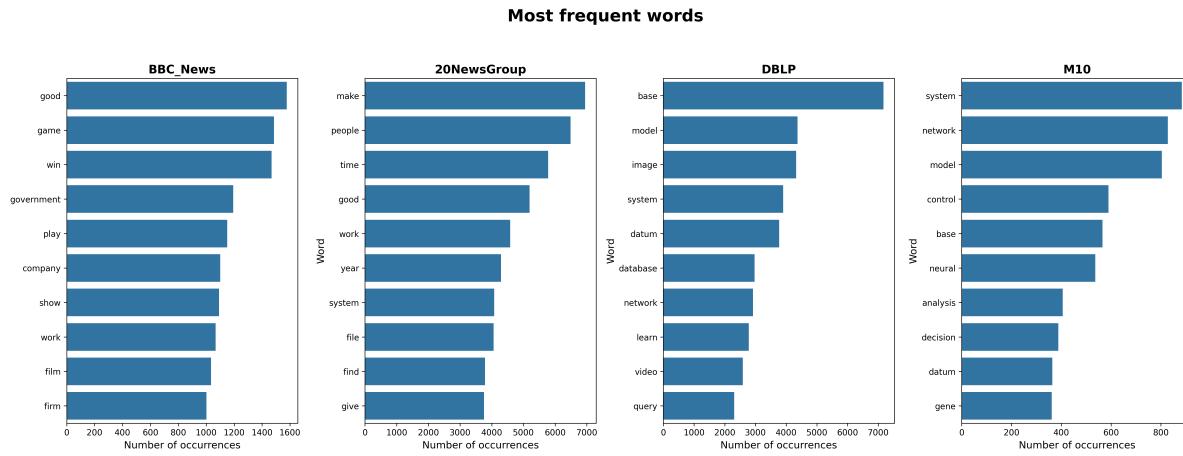


Figure 3: Most frequent words used in each dataset.

2.5 Distribution of words occurrences

The distribution of word occurrences for the datasets is shown in Figure 2.5. The general rule is that most words don't appear very often. Furthermore, with the exception of dataset M10, no dataset contains words that appear only 2 or 3 times. This can be explained in part by the fact that the dataset has already been processed and therefore contains 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.

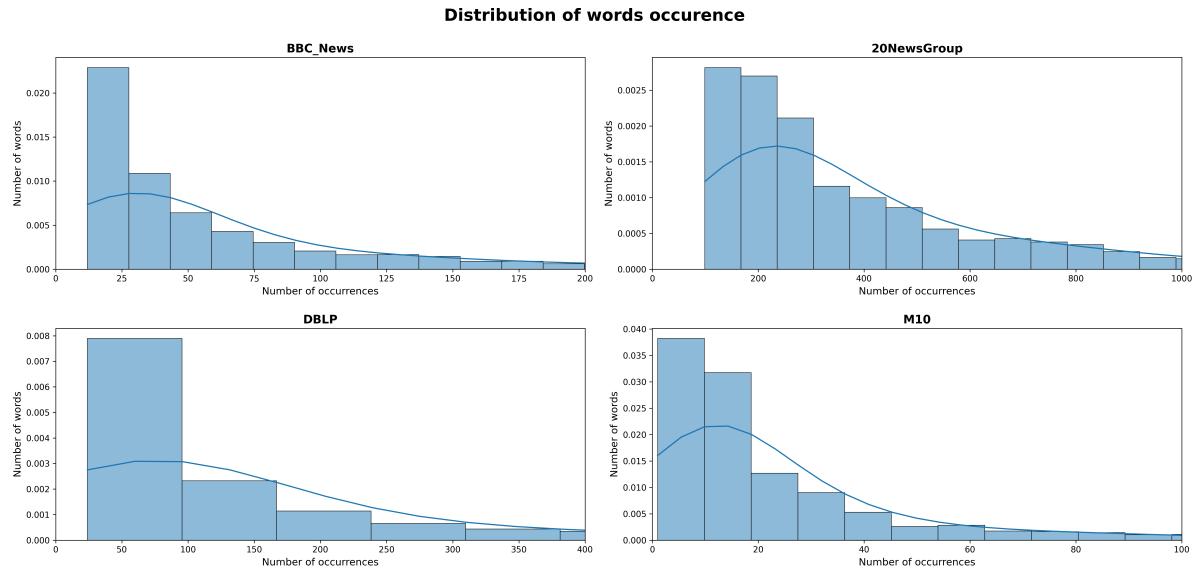


Figure 4: Distribution of word occurrences for each dataset.

3 Topic Clustering

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 Lbl2Vec and Lbl2TransformerVec

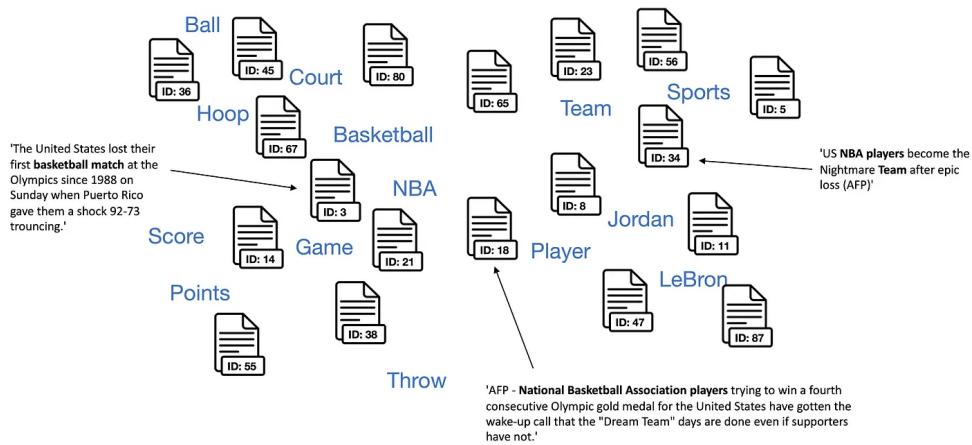
3.1.1 Theory

Lbl2Vec [6] 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:

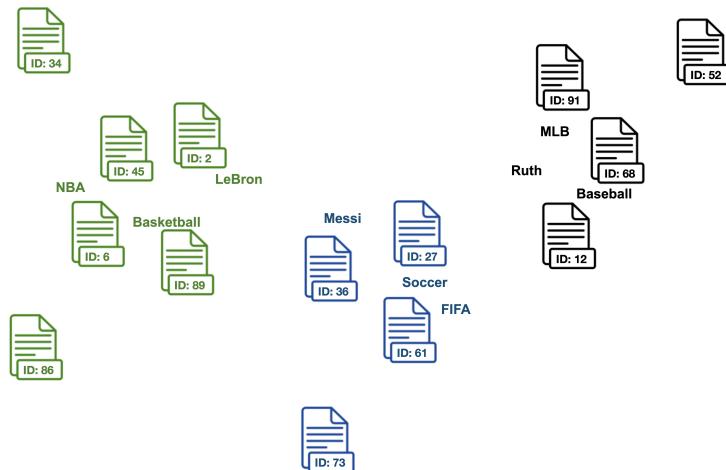
1. Use manually defined keywords for each category of interest.

Basketball	Soccer	Baseball
NBA	FIFA	MLB
Basketball	Soccer	Baseball
LeBron	Messi	Ruth
...

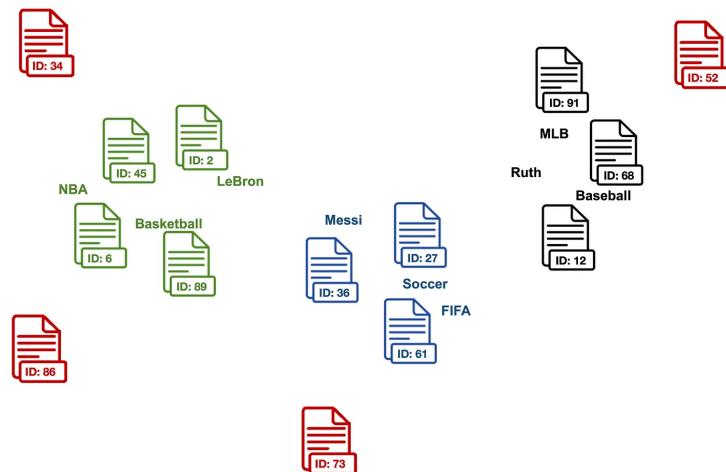
2. Create jointly embedded word and document vectors. The idea behind embedding vectors is that similar words or text documents will have similar vectors. Therefore, after creating jointly embedded vectors, documents are located close to other similar documents and close to the most distinctive words.



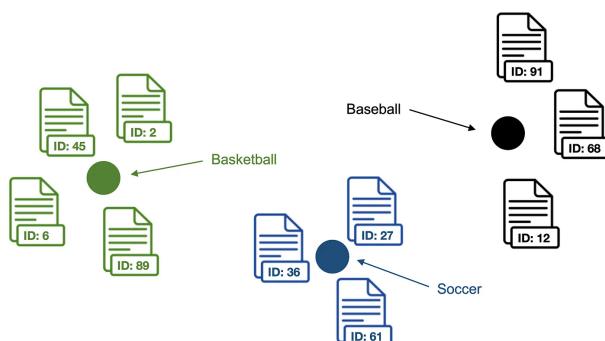
3. Find the document vectors that are similar to the keyword vectors for each classification category. We calculate cosine similarities between documents and manually defined keywords for each category. Documents that are similar to the category keywords are assigned to a dataset of candidate documents in the respective category.



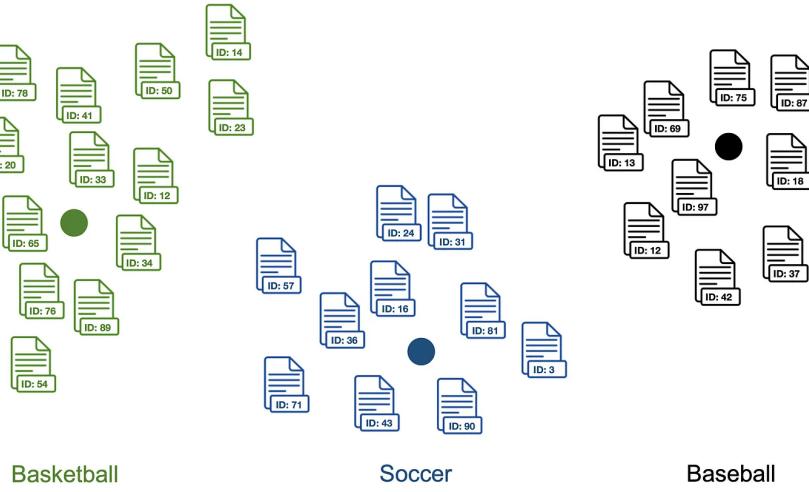
- Clean outliers for each classification category using an algorithm called Local Outlier Factor (LOF).



- Calculate the centroid of outlier vectors cleaned as a label vector for each classification category.



- The algorithm computes label vector \Leftrightarrow document vector similarities for each label vector and document vector in the dataset. Finally, text documents are classified as category with the highest label vector \Leftrightarrow document vector similarity.



This algorithm raised a number of questions:

- **How do you choose the defined keywords?**
- **How do you create embeddings?**
- Is LOF really useful for model performance? (Future work)

Note: the original article provides a library for the efficient use of models. But there are two problems. Firstly, we have no freedom over the internal model. We can't define a random state for the internal model, and it's 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, while it has a very poor 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)).

3.1.2 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 framework for visualization purposes. Doc2Vec can directly generate 2D embeddings, which is not the case 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*.

Doc2Vec		Transformer-based	
Advantages	Disadvantages	Advantages	Disadvantages
Fast to calculate	Can only use keywords seen during training Poor performance Random	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.

3.1.3 Doc2Vec - Unknown keywords problem

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).

3.1.4 Embeddings

Figure 3.1.4 shows an example of 2D embeddings created for Doc2Vec and a transformer-based model. We can see that the transformer-based model has a better ability to find embeddings because they are more clustered. But we must bear in mind that we are in a 2D configuration and it is possible that Doc2Vec also creates good embeddings in a higher dimension. **Are Doc2Vec's embeddings really relevant since they don't fulfill the property that similar words will have closer embeddings?**

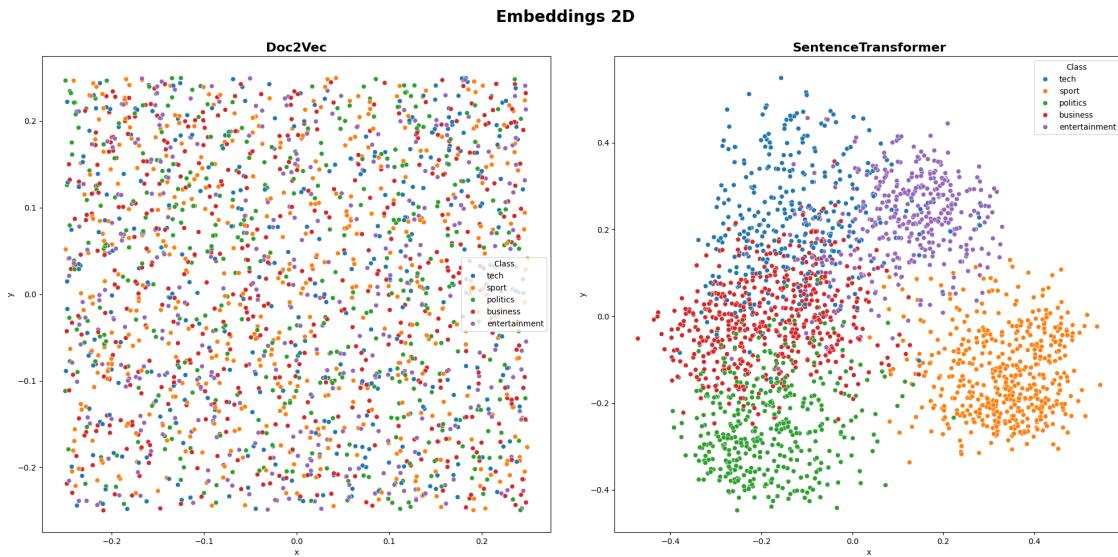


Figure 5: 2D embeddings generated by Doc2Vec and a transformer-based model on the BBC News dataset.

3.1.5 Performances

This section will attempt to answer the following question: **How does model performance vary according to the keywords used?** We will extract three types of keywords: the most frequent (supervised), the most similar (unsupervised) and the most important (supervised).

3.1.5.1 Performance using the most frequent words

The keywords used in this section are the 5 most frequent words per class, and have been extracted in a supervised way. They are listed in Table 5.

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 3.1.5.1 shows the performance of the two models on the BBC News dataset using these keywords. The measure used is the F1 score. We can see that the transformer-based model performs better than Doc2Vec, but is still very unbalanced. The score obtained for the "tech" class is much lower than that obtained for the "politics" class.

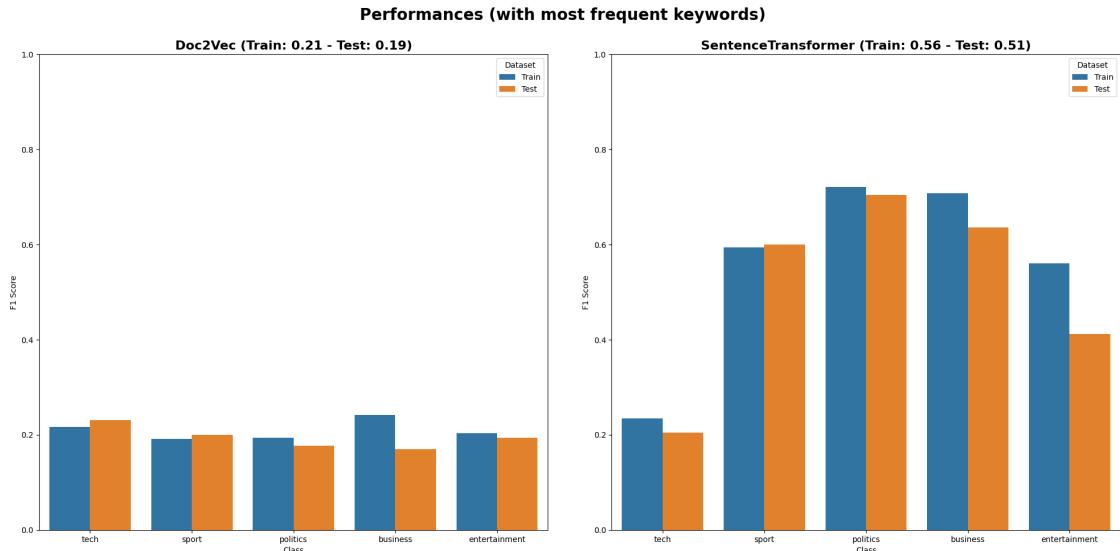


Figure 6: F1 score obtained after applying Doc2Vec and a transformer-based model to the BBC News dataset using the most frequent words in each class.

3.1.5.2 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 we need to use another method. One solution is to use the keywords defined in section 2.1 and use the spaCy algorithm as explained in section 3.1.3. We use the 5 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 3.1.5.2. The 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 return to a supervised solution to see what the results are with the "best" method.

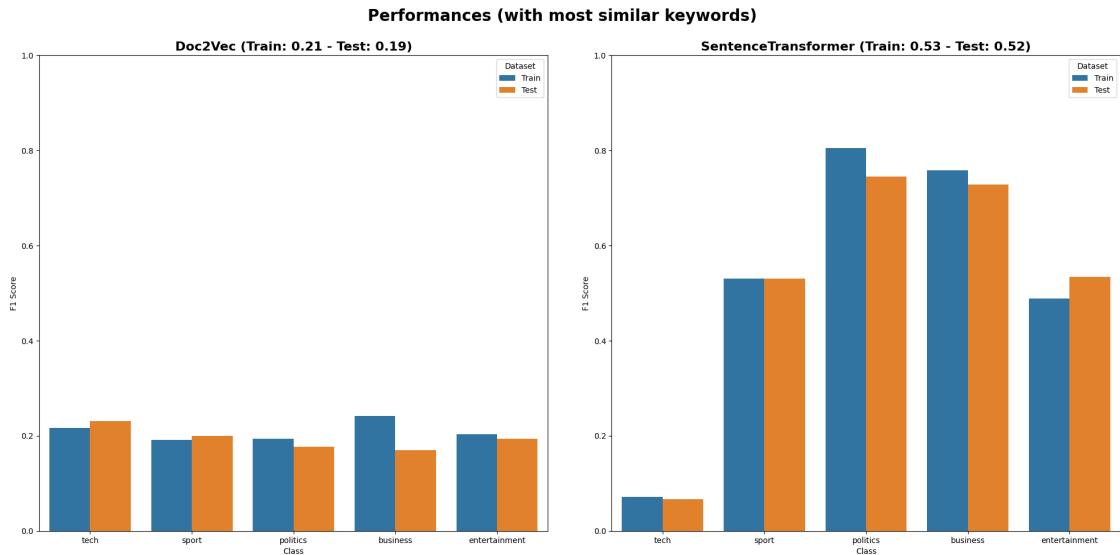


Figure 7: F1 score obtained after applying Doc2Vec and a transformer-based model to the BBC News dataset using the most similar words in each class.

3.1.5.3 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 applied a *TfidfVectorizer* to the documents before running a *Logistic Regression* model. The model predicts the classes with an F1 score of 0.99, meaning that the model will be able to extract the important words accurately. We extract the 100 most important keywords and then use them 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 can be seen in figure 3.1.5.3. We can see 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 scores have not improved. This may mean that our unsupervised method of finding keywords is as good as a supervised method.

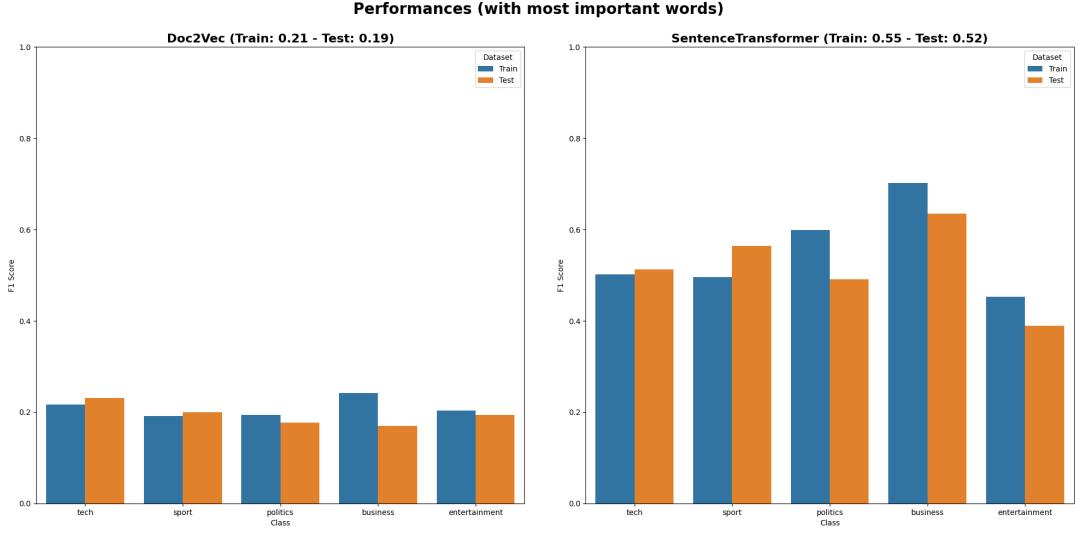


Figure 8: F1 score obtained after applying Doc2Vec and a transformer-based model to the BBC News dataset, using the 100 most important words in each class.

3.1.6 General performances

As the previous observation suggests, we will see how the models perform on each dataset using the most similar keywords method, and using the 10 most similar keywords. The results can be seen in figure 3.1.6. We have also plotted the F1 score obtained using a random method, i.e. by selecting a class at random. We can see that Doc2Vec is often worse than this random classifier, which may be explained by its poor integration performance. Interestingly, the transformer-based model outperforms the random classifier. 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 pretty poor. The results are just as bad when we use the paper library, with the exception of the BBC News dataset where the F1 score obtained with Doc2Vec is strangely good. **What could be different about our library that could be causing this behavior?**

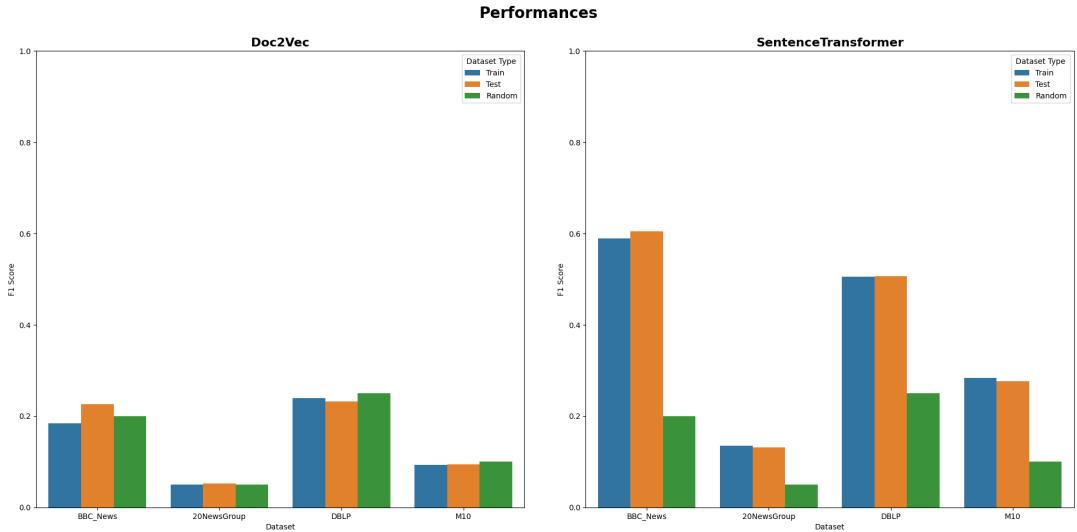


Figure 9: F1 scores obtained after applying Doc2Vec and a transformer-based model to all datasets.

3.1.7 Assignment confidence

In the previous section, we observed that the results are not very good. But we wanted to relate this to assignment confidence. **How confident is the model in the event of a bad prediction? Can we detect the data points that will cause problems?** To answer these questions, we began by plotting the confidence score of the two models applied to the BBC News dataset. This can be seen in figure 3.1.7. We can see that Doc2Vec is not at all confident. It is as if he is making random guesses. Fortunately, the transformer-based model gives better results.

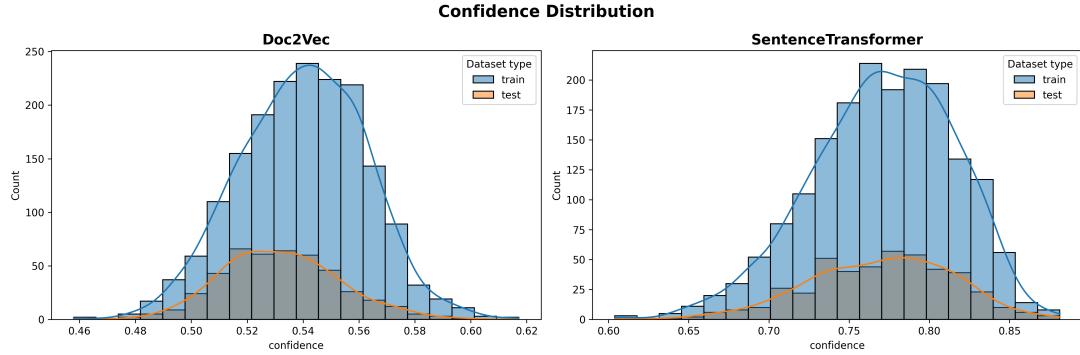


Figure 10: Confidence distributions obtained by applying the models to the BBC News dataset.

In order to differentiate between correct and incorrect predictions on the basis of a threshold applied to the confidence probability scores, we plotted these distributions when the models made both correct and incorrect predictions. This is shown in Figure 3.1.7. Unfortunately, the distributions are almost equal. This means we can't apply a threshold to differentiate between correct and incorrect predictions.

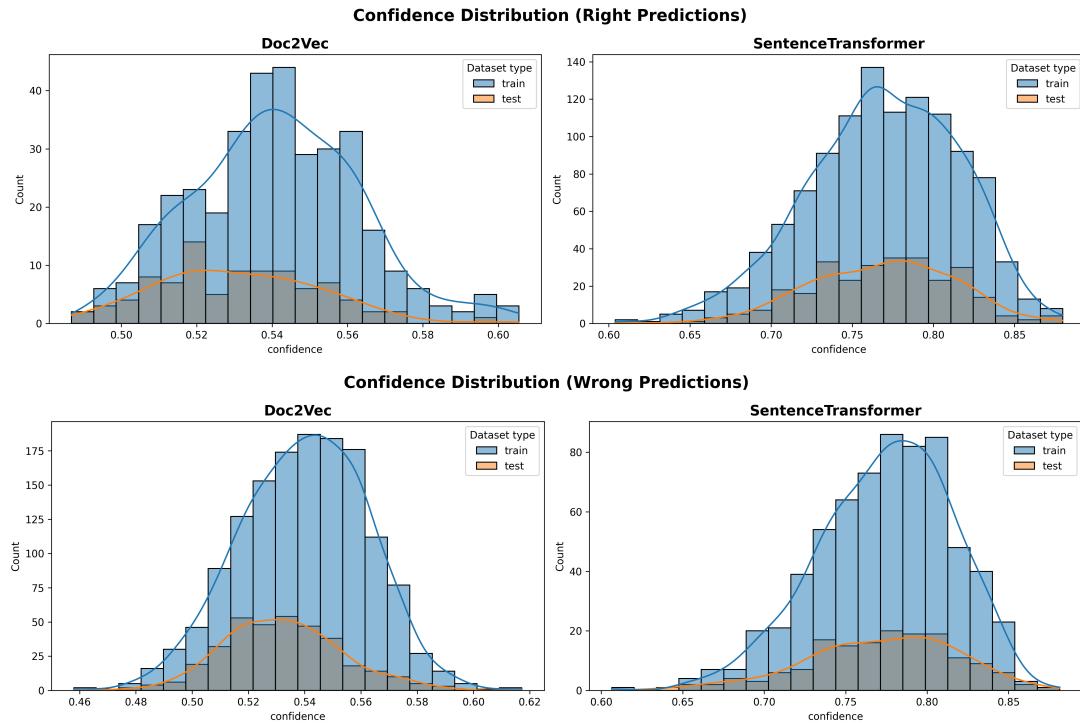


Figure 11: Confidence distributions on correct and incorrect predictions obtained by applying the models to the BBC News dataset.

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 3.1.7. Indeed, we can see that the distributions overlap on each dataset, so there is no way to detect right and wrong predictions based on the confidence score.

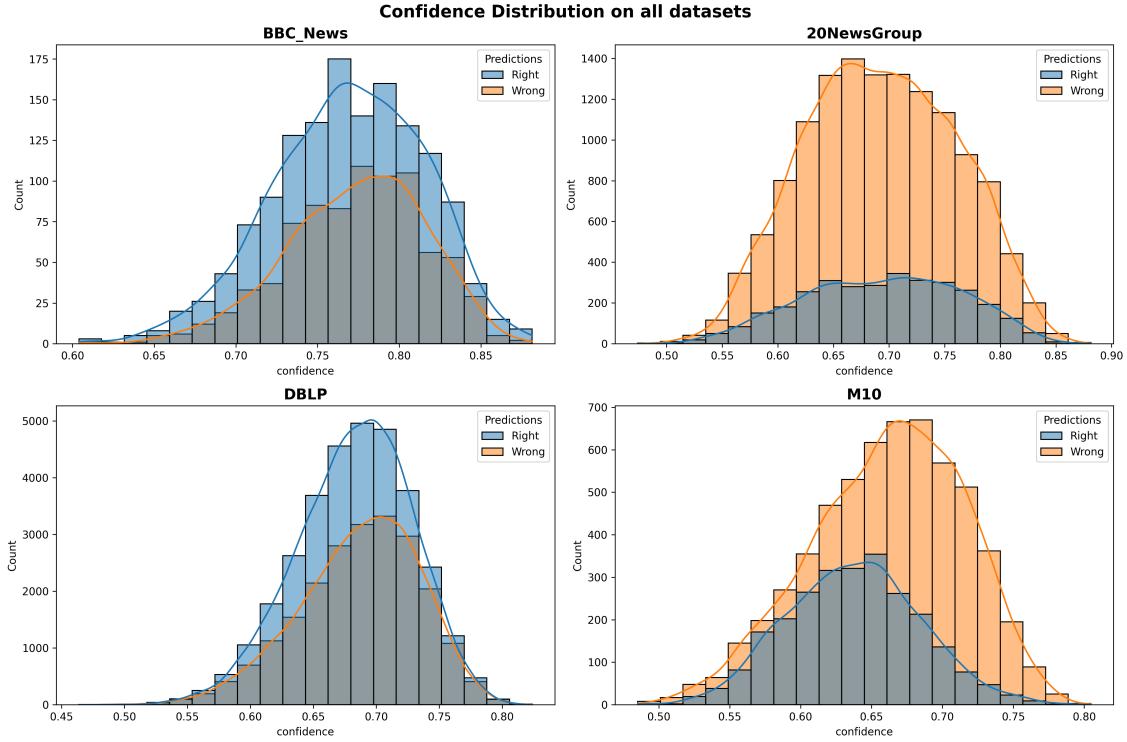


Figure 12: Confidence distributions obtained by applying the transformer-based model to all datasets.

4 Topic Modeling

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.

4.1 Evaluation

Before we start gathering information on the main method used in this scenario, we need to know how to evaluate these models. Since we are not doing classification, we cannot use basic measures such as F1 score, accuracy, etc. Evaluation measures must be linked to the topics extracted by the model and/or their correlation with the real 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 topic to evaluate the correlation between the topic found by the models and the true dataset labels.

4.1.1 Topic Coherence (TC)

This measure assesses the extent to which a topic is "supported" by a text set (called a reference corpus) [5]. 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 $\text{PMI} = \frac{P(w_i, w_j)}{P(w_i)P(w_j)}$.

4.1.1.1 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:

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

4.1.1.2 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.

4.1.1.3 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.

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

4.1.1.4 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.

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

4.1.2 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.

4.1.3 Supervised Correlation

Now, to compare the correlation between the extracted topics and the actual labels in the datasets, we can use the first N words for each topic/label and 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 word to take into account the number of words contained in the documents. Figure 4.1.3 shows a simple example of this metric. This process will be carried out for each actual label for a given topic, and the highest score will be retained, as a topic is only linked to one class and not to all classes. It will also be carried out for each topic, and the final score will be the average of each individual score.

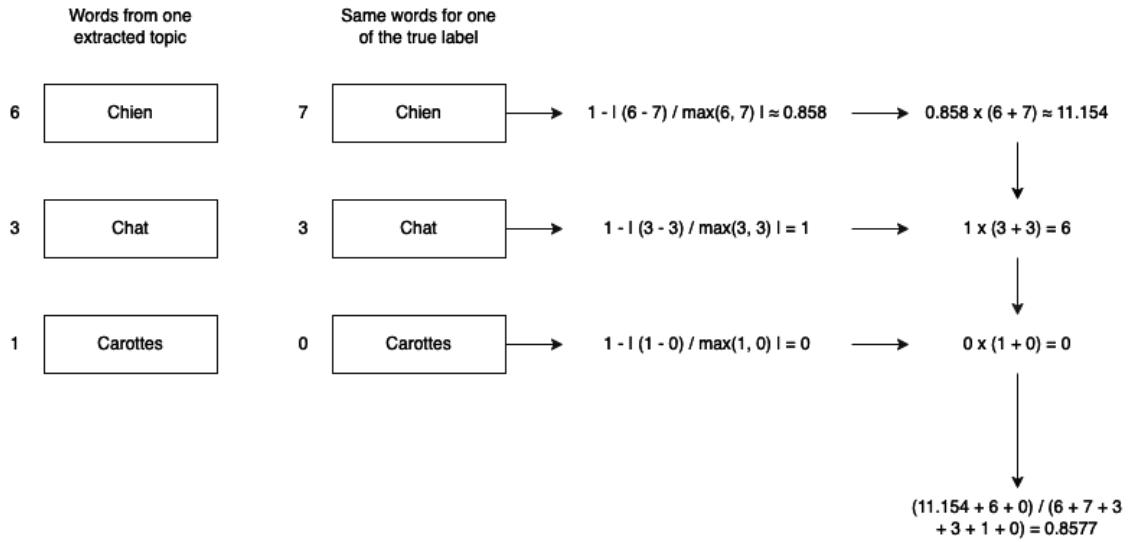


Figure 13: Example of the application of this weighted metric for one given topic and one given true labels.

The main problem with this metric is that it does not take into account the fact that each highest score can always be associated with the same class, i.e. that all extracted subjects are associated with the same class. We make this assumption by taking into account the diversity metric. If the diversity metric is high, we can assume that subjects are sufficiently diverse to be associated with different classes.

In the following sections, we will explain and discuss the results of different topic modeling models. It should be noted that no hyperparameters were used that would improve the results. Firstly, because it takes a long time to

find the best hyperparameters for a model. Secondly, 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.

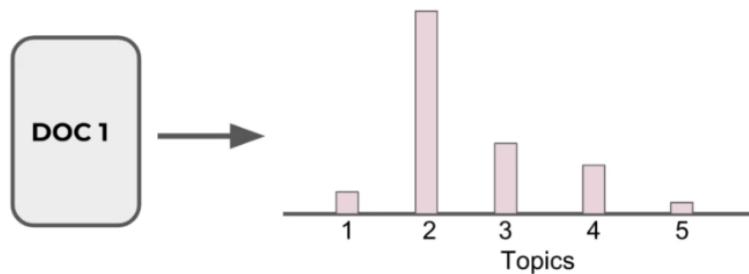
4.2 Latent Dirichlet Allocation (LDA)

4.2.1 Theory

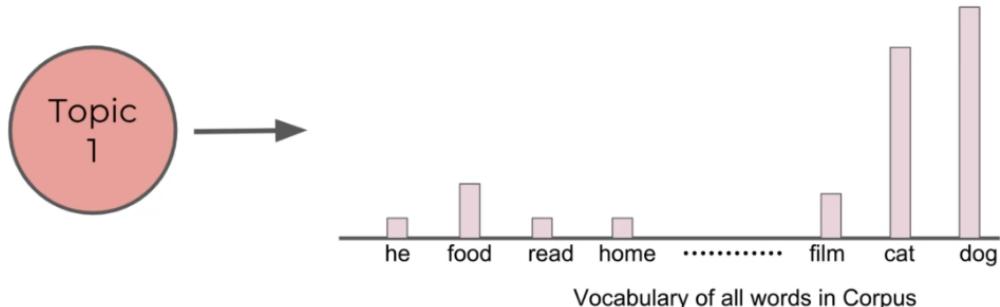
4.2.1.1 Assumptions

For this model to work, multiple assumptions are done [3, 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 subjects 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.



4.2.1.2 Algorithm

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:

- $\mathbb{P}(\text{topic } t \mid \text{document } d)$: the proportion of words in document d that are assigned to topic t .
- $\mathbb{P}(\text{word } w \mid \text{topic } t)$: 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

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

4.2.2 Results

4.2.2.1 Words in extracted topics

The results 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 is shown in Figure 14. As LDA is random, we set a random state in the model for these results. We can see that some topics are clearly linked to the real label. Topic 0 can be associated with "politics", topic 1 with "sport", topic 2 with "entertainment", topic 4 with "business" and topic 3, although more subtle than the others, can be associated with "technology". **How can we get this "automatic" detection when we're in the full input scenario?**

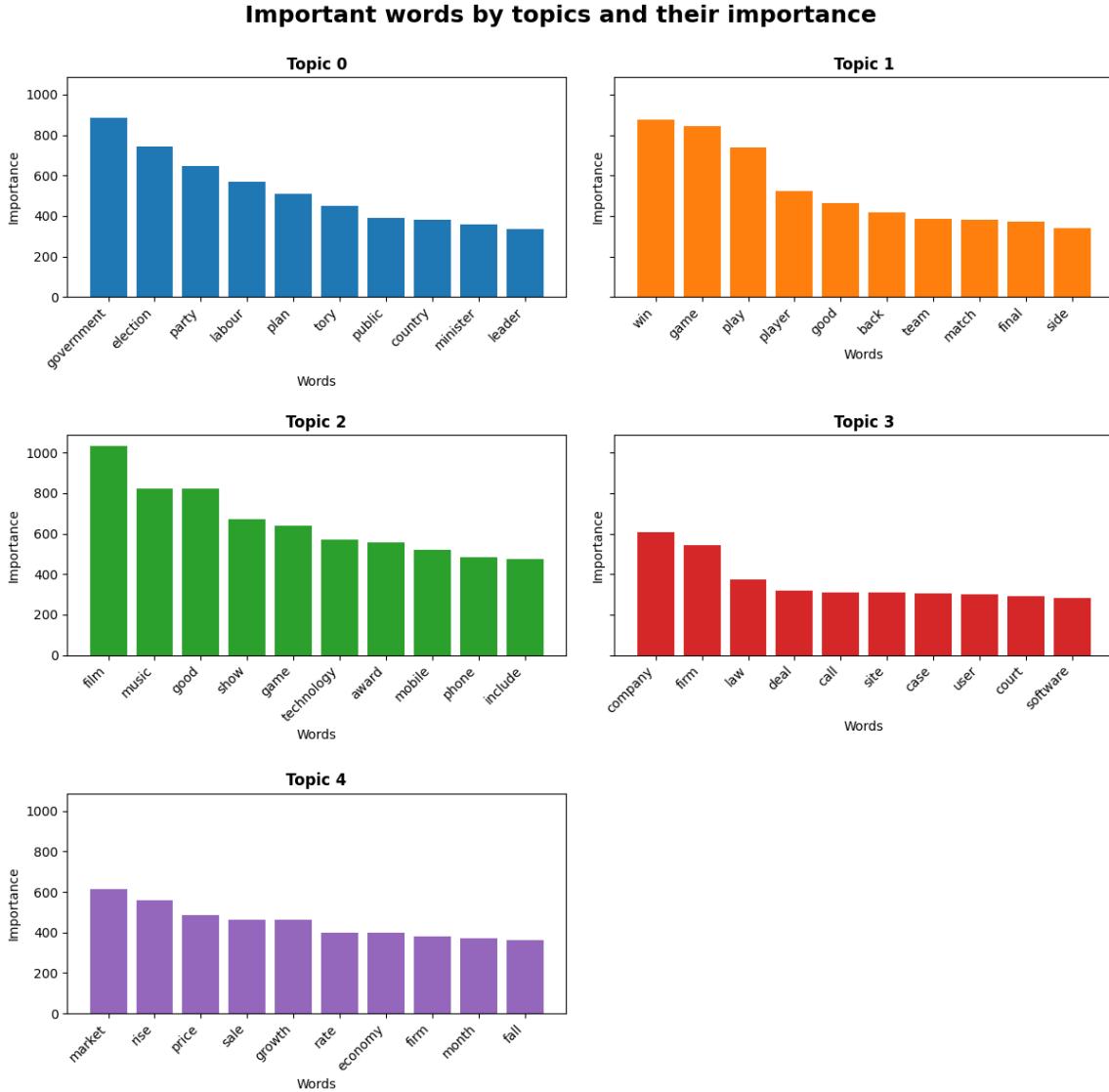


Figure 14: Important words by topic by applying the LDA method to the BBC News dataset.

The second result we can observe is the assignment of a topic directly to the words of a document. This is shown in Figure 15. Note that these are only the first 10 words of each document. This means that even if no words from the assigned topic appear, it does not mean that there is a problem with the model. This may be because all the other invisible 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 15: The first 10 words of the first 10 documents with their associated topics.

4.2.2.2 Randomness

As LDA is random, one way of evaluating the model is to calculate several iterations of the model and average all the scores. We wanted to know if the scores were always close to the mean, or if the variance was high. This is shown in Figure 16. The number of iterations used is fixed at 10. It can be seen that all metrics remain close to the mean, which means that this solution can be used. The fact that some boxplots are larger than others (e.g. for u_{mass} and u_{uci}) can simply be explained by the fact that, unlike the other metrics, they are not bounded between 0 and 1.

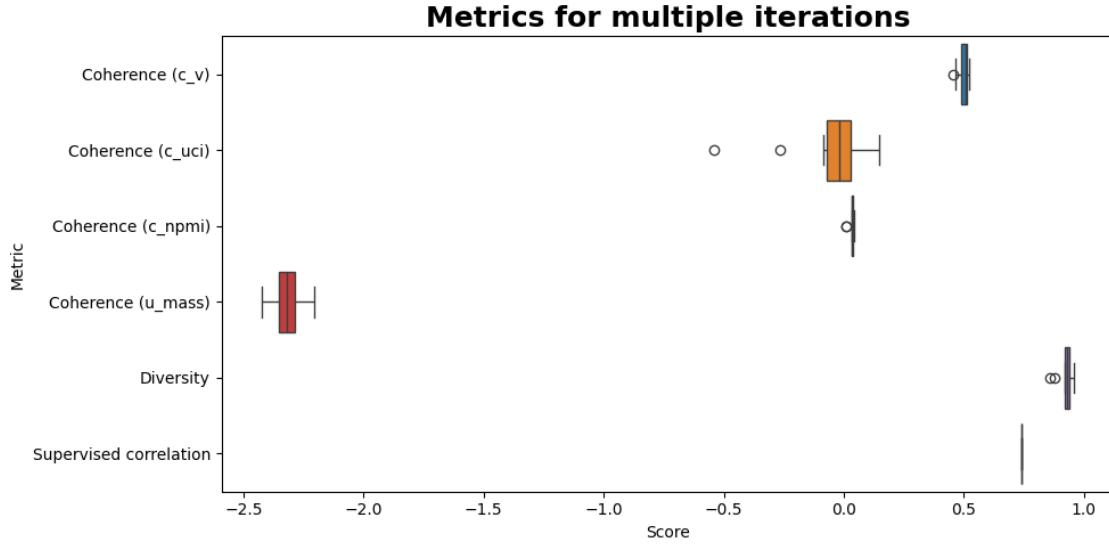


Figure 16: Box plots for all scores obtained by running the LDA model for 10 iterations.

4.3 BERTopic

4.3.1 Theory

BERTopic works in the following way [4, 2]:

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:

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

- (*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:

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

- *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:

$$\text{TF-IDF} = TF * IDF \quad (7)$$

4.3.2 Results

4.3.2.1 Words in extracted topics

As in the LDA section, the results come from applying the model to the BBC News dataset, also by setting a random state in the model. First of all, we wanted to show the most important words per topic and their level of importance. This is shown in figure 17. We can see that some topics are also linked to the actual 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, however, cannot be assigned to "tech".

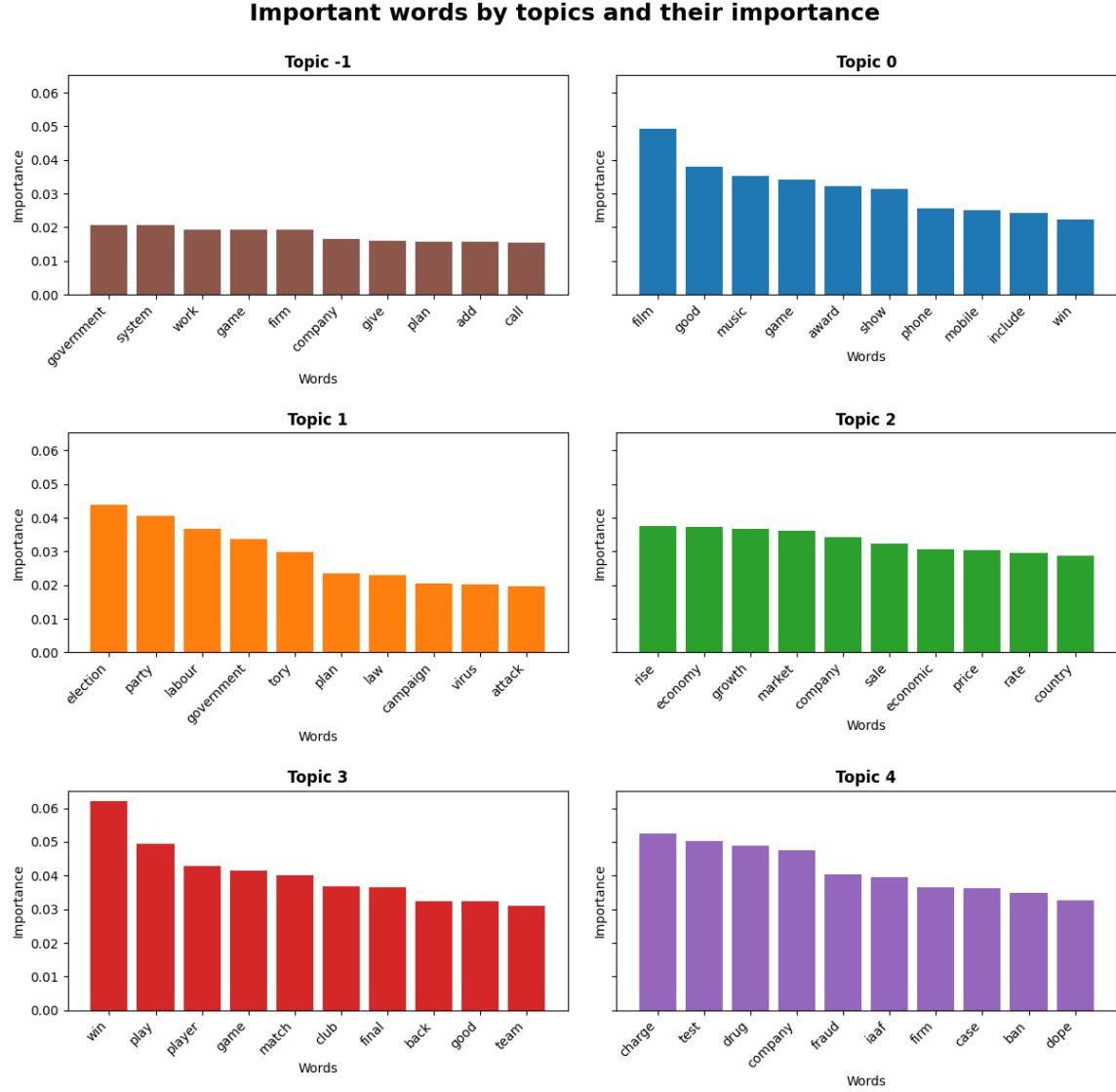


Figure 17: Important words by topic by applying BERTopic to the BBC News dataset.

4.3.2.2 Similarity matrix

Another thing we wanted to see is related to the diversity score. We wanted to see how similar the subjects were to each other. This is shown in Figure 18. We can see that the topics are somewhat similar, which can be explained by the fact that there are few topics. But in general, they are different.

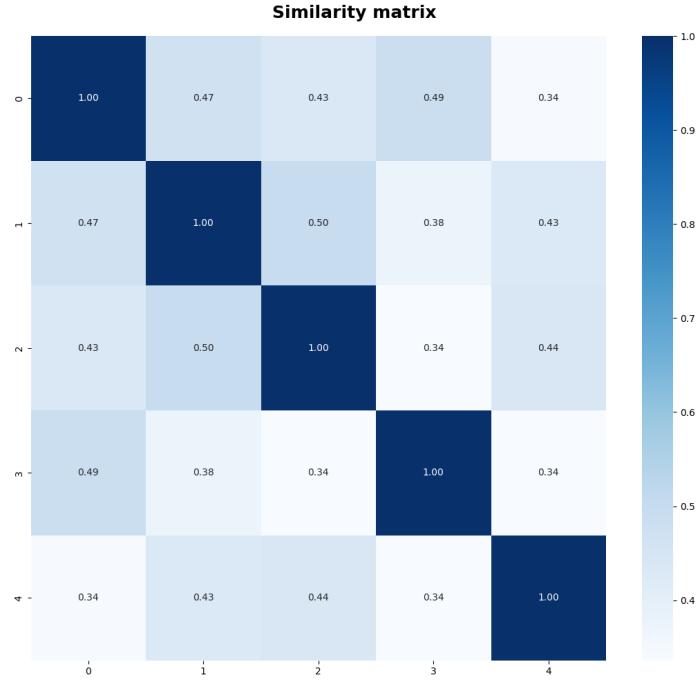


Figure 18: Similarity matrix between topics obtained by applying BERTopic to the BBC News dataset.

4.3.2.3 Randomness

As with LDA, we wanted to see if the method of taking the mean of the scores from several iterations could handle the randomness of the model. This is shown in Figure 19. The number of iterations is also fixed at 10. It can be seen that, similarly to 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 for the remainder of the report.

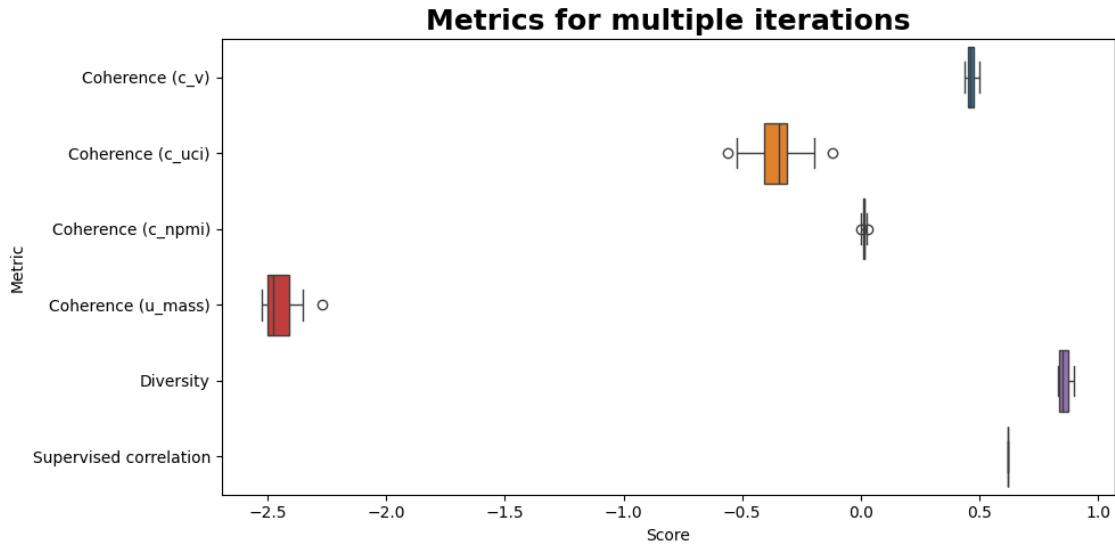


Figure 19: Box plots for all scores obtained by running the BERTopic model for 10 iterations.

4.4 Guided Topic Modeling

Guided Topic Modeling 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. This approach has practical applications in various fields, such as text analysis, recommender systems and sentiment analysis, and enables more accurate and insightful results to be obtained from textual data.

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

4.4.1 Guided LDA

What we 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. Curiously, the results are not as good as expected. The supervised correlation metric is worse for the guided version than for the base version. But this could be due to the definition of a random state, and the results could be better on average.

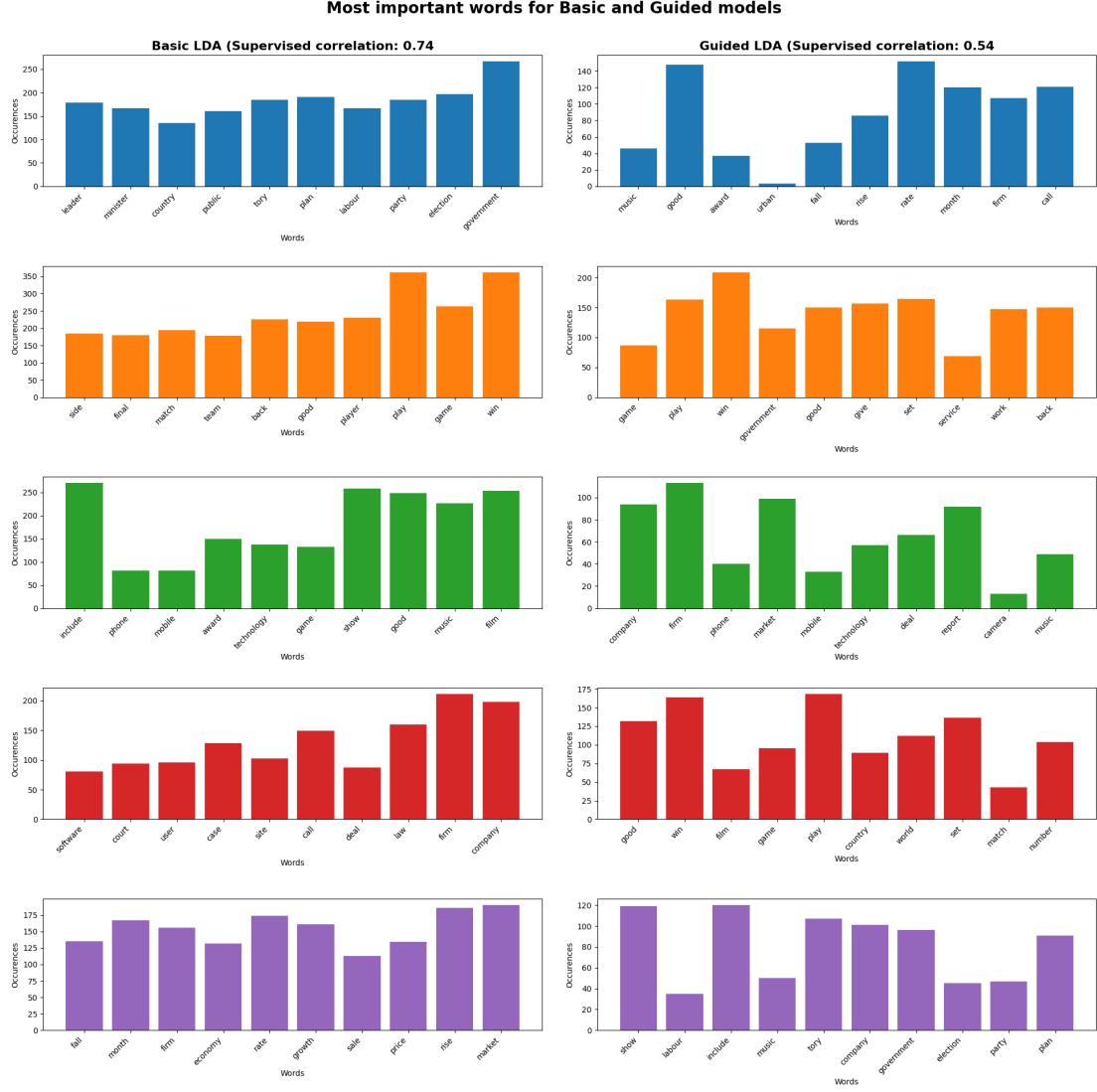


Figure 20: The most important words found by the basic model compared with those found with the guided version.

4.4.2 Guided BERTopic

As with the guided LDA, we wanted to see how close the clusters found by the guided version of the model were to the true clusters, i.e. those manually labeled in the dataset. We applied the same methodology as before. The result can be seen in Figure 21. We only take into account clusters other than -1, i.e. those containing possible outliers. This result is interesting because we can see that the clusters seem more similar to the true clusters found. In addition, the supervised correlation metric improves considerably. One hypothesis that follows from these observations is that these guided topic modeling method could be used for clustering as in the first section. This point will be detailed in Section 4.6.2.

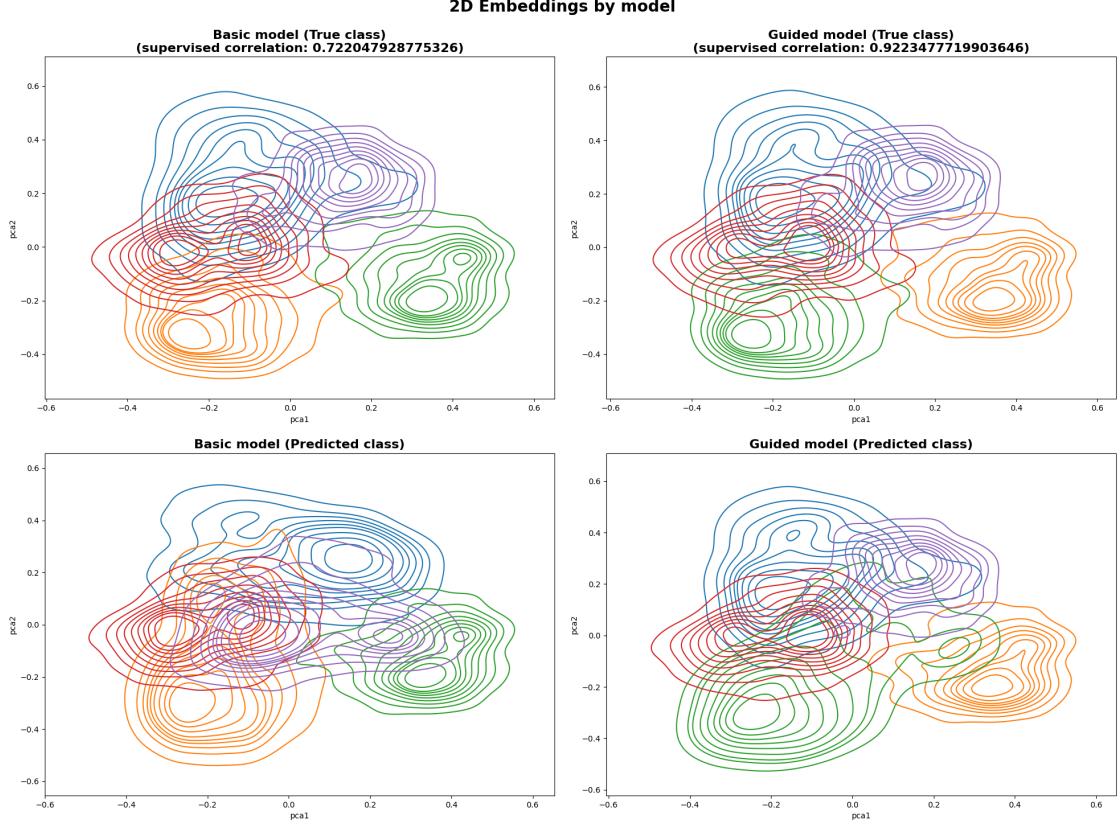


Figure 21: Clusters found by the basic model versus those found with the guided version.

4.5 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. Indeed, vONTSS 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, it is added to this section rather than the first.

4.5.1 Theory

The architecture of this model is shown in Figure 22. 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.

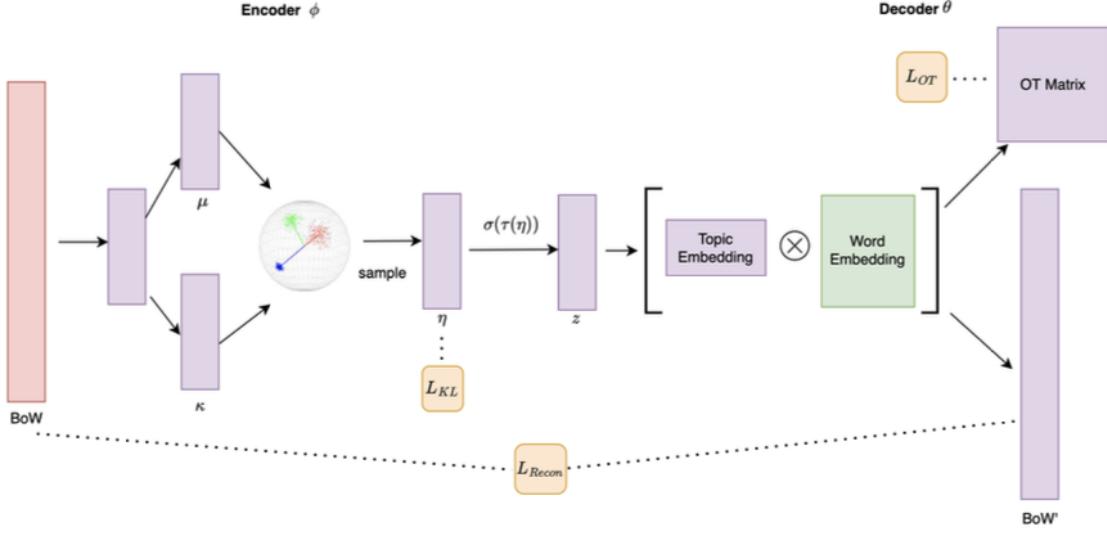


Figure 22: Architecture of the vONTSS model.

4.5.2 Results

TODO

4.6 Comparison

4.6.1 Performances

Comparison using the same number of classes than in the true labels can be seen in Table 7.

Dataset	Models	Coherence (C_v)	Coherence (c_uci)	Coherence (c_npmi)	Coherence (u_mass)	Diversity	Supervised correlation
BBC News	LDA	0.507245	-0.224381	0.026653	-2.237404	0.920000	0.824987
	BERTopic	0.480479	-0.194804	0.021948	-2.220241	0.856667	0.701449
20NewsGroup	LDA	0.600076	0.660461	0.091718	-1.767074	0.743000	0.611222
	BERTopic	0.510362	0.175887	0.067621	-2.255179	0.827619	0.382447
DBLP	LDA	0.566795	0.094171	0.038986	-1.931211	0.830000	0.612566
	BERTopic	0.555418	-0.304531	0.022946	-2.246640	0.760000	0.299963
M10	LDA	0.457851	-0.061817	0.016673	-2.270753	0.838000	0.520814
	BERTopic	0.515526	-2.204118	-0.050275	-2.546092	0.914545	0.626878

Table 7: Advantages and disadvantages of the Doc2Vec architecture compared to transformer-based models using the same number of classes than in the true labels

Comparison using a different number of classes than in the true labels can be seen in Table 8. The models therefore extracted themselves the "right" number. Here we don't compare the supervised correlation metric since it make no sense because this is two different number of classes. Also the Guided version have not be shown for the same reason.

Dataset	Models	Coherence (C_v) ↗	Coherence (c_uci) ↗	Coherence (c_npmi) ↗	Coherence (u_mass) ↘	Diversity ↗
BBC News	LDA	0.507245	-0.224381	0.026653	-2.237404	0.920000
	BERTopic	0.480479	-0.194804	0.021948	-2.220241	0.856667
20NewsGroup	LDA	0.600076	0.660461	0.091718	-1.767074	0.743000
	BERTopic	0.510362	0.175887	0.067621	-2.255179	0.827619
DBLP	LDA	0.566795	0.094171	0.038986	-1.931211	0.830000
	BERTopic	0.555418	-0.304531	0.022946	-2.246640	0.760000
M10	LDA	0.457851	-0.061817	0.016673	-2.270753	0.838000
	BERTopic	0.515526	-2.204118	-0.050275	-2.546092	0.914545

Table 8: Advantages and disadvantages of the Doc2Vec architecture compared to transformer-based models using a different number of classes than in the true labels.

4.6.2 Transformation to topic modeling

TODO

5 Application

This section presents some of the results of applying topic clustering and modeling to real data from one of the company's use cases. The use case is related to ChatBot. Many users contact ChatBots to talk about topics for which the ChatBot is not prepared to answer. The aim is therefore to use topic modeling/clustering to help understand these messages and distinguish the right ones from the wrong ones.

5.1 Dataset

TODO

5.2 Topic Clustering

TODO

5.3 Topic Modeling

TODO

6 Future work

- About Lbl2Vec:
- Is LOF really useful for model performance?

6.1 Future work

- About the scenario in which we have documents and classes in which we want to classify the documents:
- Only *all-minilm-l6-v2* was used for Lbl2Vec, could we use other LLMs?
- Are there any models other than Lbl2Vec that use LLMs in a different way to classify documents?
- Is there a way to fine-tune LLMs to fit to the problem instead of using the pre-trained version?
- For BERTopic:
 - Is there other transformer more relevant for this problem ?
 - Can other dimensionality reduction methods be used ?
 - Can other clustering methods be used ?

7 Conclusion

References

- [1] H. Bansal. Latent dirichlet allocation - analytics vidhya - medium. 12 2021.
- [2] M. Grootendorst. Bertopic: Neural topic modeling with a class-based tf-idf procedure, 2022.
- [3] R. Kulshrestha. A beginner's guide to latent dirichlet allocation(lda). 12 2021.
- [4] Priyanka. Understanding bertopic intuitively - level up coding. 2 2023.
- [5] 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.
- [6] 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.