Natural Language Processing:
Write me - Enron Email Dataset

Matteo Onger

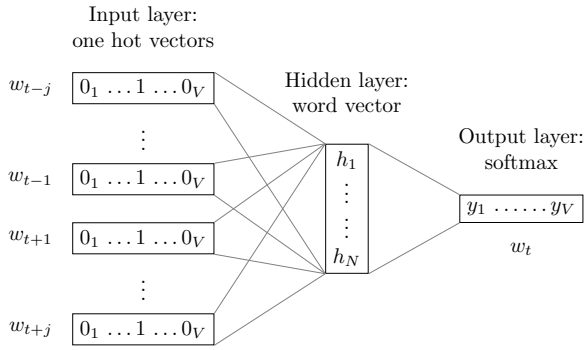
December 2024

1 Introduction

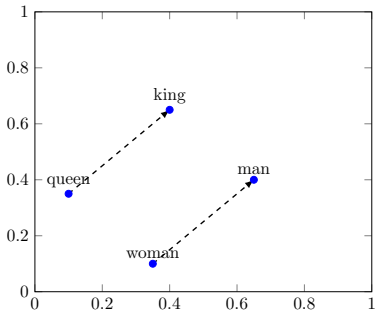
The *Enron Email Dataset* is a collection of approximately 500,000 emails, generated by Enron employees between 1997 and June 2002, made public by the Federal Energy Regulatory Commission (FERC) during its investigation about the company's collapse.
The aim of the project is to combine some techniques to extract the main themes and topics covered in the emails analyzed. Tasks like this are extremely common and therefore techniques developed to solve problems of this kind are numerous: from the simplest methods based on token frequency, see for example text clustering based on *TF-IDF* features, to topic modeling algorithms, like *Latent Dirichlet Allocation*, up to the most recent systems based on neural networks (DNNs, RNNs, transformers, ...), like *BERTopic* [3]. In Section 2 a simple alternative method to do this, based on *Word2vec*, is proposed and its performance is then discussed in Section 3, in which the topics found are also shown.
The chosen dataset was clearly not created for this purpose and thus it allows us to test the effectiveness and the applicability of the proposed approach on a real-world problem.

2 Research question and methodology

It is well known that, thanks to Word2vec, it is possible to obtain a vector representation of words, i.e. each word becomes a point in an N-dimensional space, where N is the length of vectors. The most interesting property of this mapping is that the vector space created is able to "capture the meaning" of a word based on the surrounding terms and at least part of the relationships that exist between them.



(a) Schema of the CBOW model. Here V is the size of the vocabulary.



(b) Example of semantically related words represented in a vector space.

Figure 1: Word2vec.

The proposed model aims to map documents in the same vector space and, in particular, the ambition would be to make the position of a document in space dependent on the topic being discussed. In this way, finding the nearest words in the defined vector space would be enough to know the main theme of a document. By doing so, a first advantage is that the distance between a document and the words closest to it can be used as an indicator of the quality with which those words represent the document and thus of the "classification" executed.

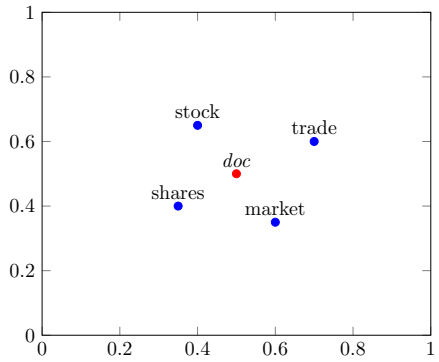


Figure 2: Example of the desired result. In this case we could conclude that the topic discussed is the stock market. In practice, the space is N-dimensional and the scale of values can be completely different.

Obviously, this method involves two basic assumptions:

1. a topic covered by a document can be summarized basically in one word, so the granularity of the topic is limited;

2. each document covers a single topic. If not, it could be possible to cluster the words that compose the document in the hope of identifying the different themes, but without being able to map the document into a single vector.

Subsequently, if a document could be mapped into a "meaningful" vector, it would then be possible to apply a clustering algorithm to group together documents that deal with similar topics, thus adding a further level of abstraction.
Even supposing that the assumptions made are true, the main difficulty of this method is evident: the most intuitive way to associate a vector with a document given the vectors of its words would be to calculate their average, but the mean is influenced by all the words present, even those that do not determine the main theme and which therefore can lie in distant regions of the latent space. These words can be many and behave like outliers: they move the document vector away from the words that best represent its theme and which, on the contrary, are often quite close to each other.

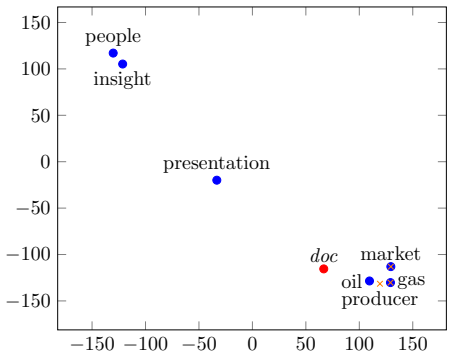


Figure 3: Example of the processing performed. Email *ID:431296*. The red point represents the computed document vector, in blue the vectors of the tokens present in the email's body and in orange the words closest to the document vector. t-SNE was used to project vectors into a two-dimensional space.

So the idea is to use a weighted mean of the word vectors. This is obviously the most complicated part: ideally, tokens that determine the topic of a document should have a high weight, while the remaining ones should have a weight tending to zero. But if we knew the ideal weights, the whole problem would basically be already solved; in practice we can only try to estimate these values as best as possible. The initially proposed solution uses as weight the TF-IDF, which is itself an indicator of the representativeness of a word for a document, and a score that, using the vector representation of the tokens, estimates the similarity between a word and the other words present in the document under consideration: because it is likely that different words contribute to determining a topic, but it is also likely that these are semantically related and therefore close in the latent space created by the Word2vec model. So, the vector associated with the document d

is calculated according to Formula 1, where $|d|$ is the length of the document in number of tokens, $\{d\}$ means the set of tokens present in the document, therefore without duplicates, $vec(\dots)$ is the vector representation of the argument and $sim(\dots)$ is a value between 0 and 1 that indicates how similar two tokens are using their vector representation; while $tf(\dots)$, $idf(\dots)$ and $tfidf(\dots)$ are obviously the TF, the IDF and the TF-IDF score of a token.

$$\begin{aligned} vec(d) &= \frac{1}{|d|} \sum_{tk \in d} vec(tk) \cdot idf(tk) \cdot \left(\frac{1}{|d|} \sum_{\substack{tk' \in d: \\ tk' \neq tk}} sim(vec(tk), vec(tk')) \right) \\ &= \sum_{tk \in \{d\}} vec(tk) \cdot tfidf_d(tk) \cdot \left(\sum_{\substack{tk' \in \{d\}: \\ tk' \neq tk}} tf_d(tk') \cdot sim(vec(tk), vec(tk')) \right) \\ &= \sum_{tk \in \{d\}} weight(tk, d) \cdot vect(tk) \end{aligned} \tag{1}$$

The formula was subsequently improved by introducing three additional parameters (α, β and ϵ) in the calculation of the weights, which are finally normalized using the L1 norm before actually being used to compute the document vector. Equations 2 and 3 show the changes made: by introducing the exponents α and β, and the threshold ϵ, it is hoped to polarize the weights, further penalizing "outlier" tokens and favoring the topic-related ones.

$$w_{tk,d} = (tfidf_d(tk))^{\alpha} \cdot \left(\sum_{\substack{tk' \in \{d\}: \\ tk' \neq tk}} tf_d(tk') \cdot sim(vec(tk), vec(tk')) \right)^{\beta} \tag{2}$$

$$weight(tk, d) = \begin{cases} \dfrac{w_{tk,d}}{\sum_{tk' \in \{d\}} w_{tk',d}} & \text{if } w_{tk,d} \geq \epsilon \\ 0 & \text{otherwise} \end{cases} \qquad \forall tk \in \{d\} \tag{3}$$

Overall, the proposed procedure involves the following steps: (step 1) the implemented preprocessor, mainly based on *spaCy*, tokenizes and normalizes the corpus received as input, the output is then a list of tokens for each document in the collection. The way the corpus is preprocessed also depends on the arguments provided during object initialization, mainly the preprocessor can:

- lowercase all the tokens;
- remove tokens and/or entire lines matching the set regular expressions;
- remove stop words (customizable);
- keep only/remove tokens labeled with certain part-of-speech (POS) tags;
- keep only/remove tokens labeled with certain entity tags;
- merge noun chunks;
- lemmatize tokens by replacing them with their base form;

After the preprocessing, the tokenized corpus can be passed to an object of the class `myDoc2vec` (step 2), which, using the word vectors and the `TfidfVectorizer` defined during the initialization of the object, and the formula previously shown, computes a vector for each document in the corpus. Once the document vectors are obtained, (step 3) it is possible to list the words closest to them, which should constitute a sort of label, indicating the topic being discussed. The document vectors obtained can also be clustered in order to identify documents dealing with similar topics and, exactly as before, it is possible to find for each cluster the closest words, which should indicate the topic. Clustering is performed here using the *K-Means* algorithm, the *Elbow* and the *Silhouette methods* are used to estimate K, but obviously other solutions can be applied. The Figure 4 schematically summarizes the various steps just described.

The next section shows the topics found and the results of some simple experiments to verify and reflect on the applicability and the performance of the proposed solution.

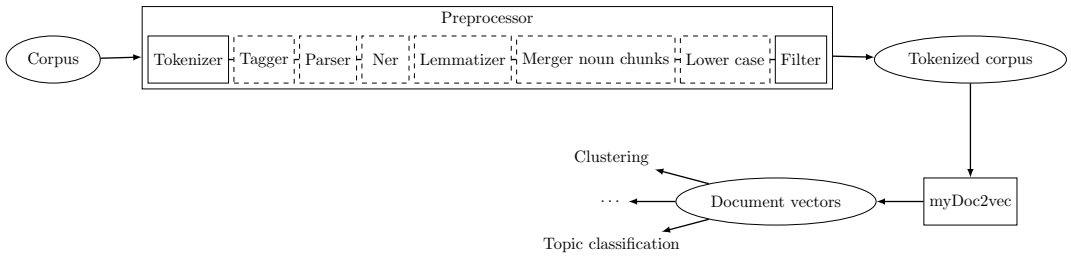


Figure 4: Diagram of the proposed solution. Dashed steps are only executed if enabled, while ellipses denote inputs and outputs.

3 Experimental results

3.1 About the dataset

Before focusing on the proposed model, it is necessary to make a few remarks on the dataset used and the assumptions made in Section 2.
The most detailed information about the topics covered is obviously contained in the bodies of the emails, unfortunately the latter also turns out to be the most difficult to process. This is mainly due to the high variability of the content: from formal and well-written texts to informal messages and slang-heavy textspeak, from bullet points and tables to nested emails (headers included), such as in case of forwarded messages. In addition to this, as the Chart 5 shows, also the length of emails varies greatly: from just a few dozen to hundreds of thousands of characters.

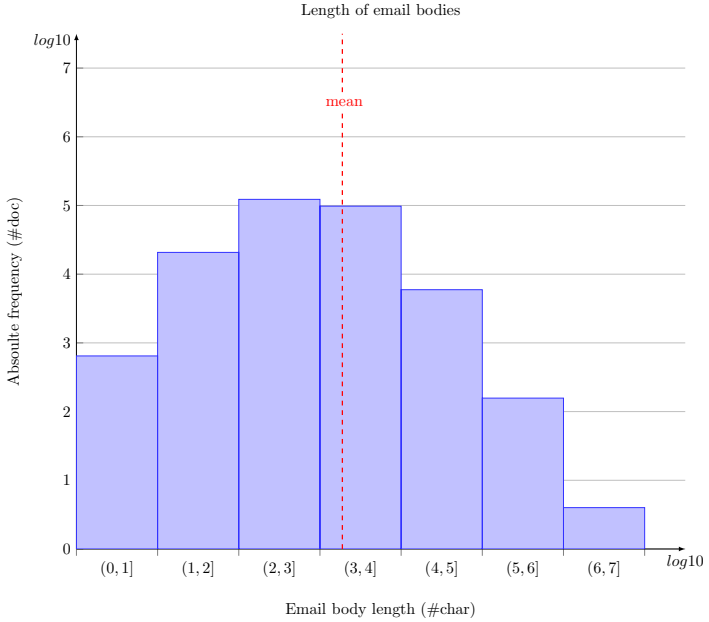


Figure 5: The plot shows the length of email bodies in number of characters. In red the average length. The values on the axes are in $log10$ scale.

This implies that the assumptions made in the previous section are only partially true, because there are very long emails, typically reports, which cannot be associated with a single topic without losing information. All this makes preprocessing extremely necessary and non-trivial.

The central role of the preprocessor, however, is not a unique feature of the proposed algorithm: testing other common algorithms, like TF-IDF, LDA and even BERTopic, it was verified that none of them is able to produce good results without a specific preprocessing phase for the problem under consideration. Certain tokens, but also entire lines and paragraphs, must be discarded because they are effectively devoid of useful information and, if retained, they negatively affect the classification. The most insidious tokens are the words like “time”, “email”, “sender”, etc., which are rare enough to still have a good TF-IDF score but are transversal to almost all topics, so they make it more difficult to identify clusters and the theme that characterizes each of them. See, for instance, Example 6.

```
CLUSTER ID => MAIN TOKENS
-------------------------------------------------
00 => [thank time day question week]
01 => [deal thank pm number contract]
02 => [pm meeting thank time conference]
03 => [message email mail information recipient]
04 => [fax phone thank number mail]
05 => [employee company consumer energy retirement]
```

Figure 6: Clusters found by applying the K-Means algorithm to the TF-IDF matrix of emails between 800 and 1300 characters. Note that several main tokens are shared by different clusters, making it difficult to identify a theme.

3.2 MyDoc2vec - Email's body

Since, due to the limited computational power available, it is not possible to consider the entire dataset, the proposed model was used to process only emails between 1,000 and 20,000 characters (103,050 documents). The clusters obtained and the topics associated with them are shown below.

```
CLUSTER ID (size) => CLOSER TOKENS (cosine similarity)
-------------------------------------------------------------------------------------------------------------------
00 (5119) => [gas (0.81), capacity (0.68), pipeline (0.65), facility (0.64), peaking (0.637)]
01 (7586) => [group (0.71), objective (0.71), effort (0.69), involvement (0.67), coordination (0.66)]
02 (4441) => [lunch (0.71), guy (0.68), dinner (0.68), touch (0.64), town (0.64)]
03 (1595) => [employee (0.80), company (0.67), retirement (0.64), workforce (0.61), bankruptcy (0.60)]
04 (5062) => [agreement (0.82), contract (0.78), transaction (0.70), counterparty (0.69), provision (0.68)]
05 (6139) => [question (0.70), following (0.68), information (0.66), regard (0.65), asap (0.64)]
06 (3531) => [power (0.88), electricity (0.83), state (0.77), utility (0.76), energy (0.75)]
07 (4479) => [payment (0.77), monie (0.71), payable (0.67), remittance (0.65), receivable (0.64)]
08  (917) => [file (0.97), spreadsheet (0.72), format (0.66), worksheet (0.64), template (0.62)]
09 (3136) => [joy (0.75), friendship (0.73), happiness (0.73), treasure (0.72), hug (0.69)]
10 (1980) => [message (0.97), addressee (0.65), communication (0.62), important (0.60), attachment (0.58)]
11 (1928) => [deal (0.97), enpower (0.67), sitara (0.61), orig (0.60), annuity (0.60)]
12 (6804) => [issue (0.68), proceeding (0.65), extent (0.65), applicability (0.64), determination (0.63)]
13 (3656) => [meeting (0.92), conference (0.77), agenda (0.76), discussion (0.73), attendance (0.72)]
14 (5714) => [homepage (0.65), website (0.65), address (0.64), information (0.62), online (0.61)]
15 (5476) => [commodity (0.70), trader (0.66), trading (0.66), market (0.66), risk (0.65)]
16 (5317) => [price (0.76), computation (0.68), mwh (0.66), rate (0.65), volume (0.65)]
17 (2453) => [student (0.74), hire (0.73), recruitment (0.72), resume (0.69), recruiter (0.68)]
18 (4664) => [draft (0.83), document (0.78), revision (0.77), comment (0.75), clarification (0.70)]
19 (2335) => [fare (0.73), nonrefundable (0.73), travel (0.71), ticket (0.70), reservation (0.68)]
20 (7572) => [news (0.52), world (0.51), leader (0.49), opportunity (0.48), company (0.48)]
21 (4298) => [information (0.63), use (0.62), affiliate (0.60), entity (0.59), purpose (0.57)]
22 (1098) => [game (0.98), fantasy (0.66), offseason (0.65), backfield (0.65), championship (0.64)]
23 (4028) => [datum (0.74), database (0.72), interface (0.69), functionality (0.68), application (0.67)]
24 (3722) => [company (0.87), share (0.69), profitability (0.67), stock (0.65), investor (0.65)]
```

Figure 7: Main clusters found by applying the 25-Means algorithm to the document vectors. The five closest words to each centroid are reported here to indicate the topic covered.

Clearly, the results obtained are affected by how the model's hyper-parameters are set, which are many and difficult to optimize, also due to the lack of a ground truth.
Among the parameters that most determine the document vectors computed, there are the word vectors: if they are obtained from pre-trained models, like the *Google News Word2vect*, they will have a more general and complete "knowledge" of the meaning of the words, but, if the corpus under analysis is big enough, they can also be obtained by directly training a Word2vec network from scratch. In this case, knowledge is deeply corpus-based, but this is not necessarily a bad thing, because in this way it is possible to capture relationships between words that are characteristic of the analyzed dataset. Consider, for example, the Table 1 which shows the proximity between the word `sitara`, which appears in Cluster 11, and three other peculiar terms.

Words:	`deal`	`eol`	`tagg`
$\cos(vec(\texttt{sitara}), vec(\cdot))$	0.72	0.60	0.58

Table 1: Cosine similarity computed by training a Word2vec model from scratch on the corpus. The similarity between these terms is justified here [2].

As regards the preprocessor's hyper-parameters, the best results were obtained by keeping only the nouns, which also intuitively are those that say the most about the topic. This choice, together with the hyper-parameters `min_df` and `max_df` [4], also helps to significantly reduce the computation time, but it does not solve the problem discussed in Section 3.1. To reduce the effect of those words that, despite having a good TF-IDF score, are not indicative of a topic, the list of stop words has been customized (this however requires an a priori knowledge) and the Formula 3, thanks to the average similarity term, must be able to assign low weights to these tokens.

Given the complexity of the dataset and the model used, rather than trying to analyze the results obtained with one of the standard methods for unsupervised problems, it was decided to ask some supervisors to evaluate a subset of the emails processed. The documents analyzed are obviously few, and all this can be considered a simple preliminary study, but they are still sufficient to get a first idea of the quality of the work done by the proposed model.

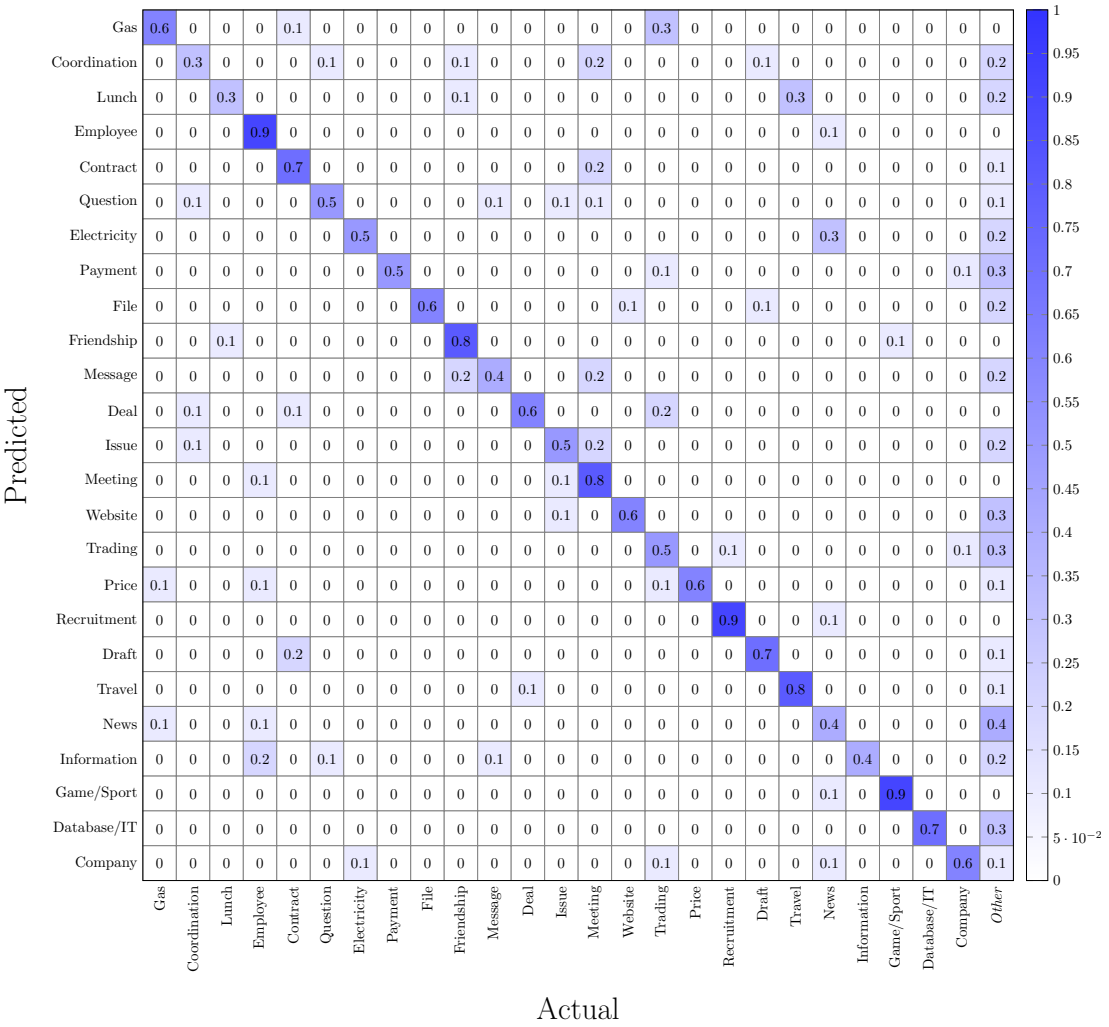


Figure 8: Confusion matrix computed using a small sample of 250 emails evenly distributed over the 25 clusters/topics found.

From the data in the Figures 7 and 8, it can be seen that the majority of the topics extracted, applying the proposed model, are absolutely reasonable given the nature of the dataset. And, even if some topics may seem apparently strange, they are actually correct: for example, considering Cluster 22, there are indeed several emails regarding sports, generated primarily by employees who subscribed to NFL newsletters or similar services.
Certain clusters, like the 18th or 23th, have, as closest words, terms that allow to easily identify a precise topic for each of them and in these cases, usually, almost all emails belonging to these groups are correctly classified. In some other cases, instead, when the closest words are more generic or similar to those of other groups, understanding the topic being discussed becomes more difficult and classification is more prone to errors. This happens for example for Clusters 10 and 21. In addition, certain topics, although clearly defined, are interconnected, like the 3th and 20th, and therefore it is predictable that the model may risk confusing documents belonging to these groups.

All this, of course, is influenced by the preprocessor and the model proposed for computing the document vectors, but also by the clustering algorithm, its hyper-parameters and the methods used to estimate K, i.e. the number of clusters, which unfortunately, usually, work worse in high-dimensional spaces due to the curse of dimensionality.

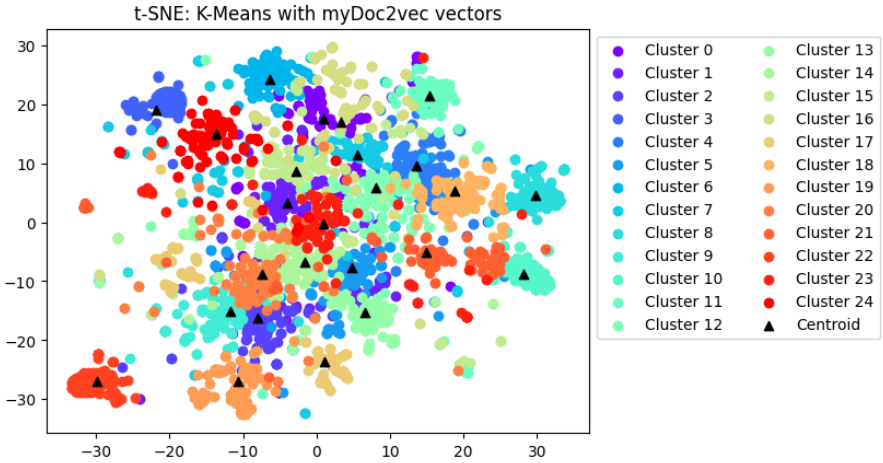


Figure 9: For each category, the centroids and 100 document vectors representing email bodies are projected in a two-dimensional space using t-SNE technique in order to facilitate visualization.

Finally, as shown in Table 2, it can be observed that, at least on average, the incorrectly classified documents are farther from the points representing their respective topics than the ones correctly labeled. The distance can therefore be used as a first, simple indicator of the quality of the classification: the greater the distance, the greater the risk of errors.

Cluster ID	Mean distance		Delta
	correctly labeled	wrongly labeled	
00	0.311	0.321	0.010
01	0.244	0.381	0.137
02	0.216	0.429	0.213
03	0.035	0.129	0.094
04	0.194	0.343	0.149
05	0.293	0.411	0.118
06	0.180	0.236	0.056
07	0.325	0.429	0.104
08	0.111	0.171	0.060
09	0.337	0.554	0.217
10	0.035	0.142	0.107
11	0.074	0.280	0.206
12	0.322	0.370	0.048
13	0.237	0.300	0.063
14	0.358	0.486	0.128
15	0.306	0.324	0.018
16	0.326	0.328	0.002
17	0.313	0.430	0.117
18	0.225	0.405	0.180
19	0.228	0.453	0.225
20	0.532	0.600	0.068
21	0.264	0.351	0.087
22	0.100	0.021	-0.079
23	0.326	0.432	0.106
24	0.274	0.258	-0.016
Mean	0.247	0.343	0.096

Table 2: The table shows, for each cluster, the average cosine distance (from the respective centroids) of correctly classified emails and incorrectly classified ones.

3.3 MyDoc2vec - Email's subject

Another field that can contain extremely useful information regarding the topic of an email is the subject, if only it were always present and well formulated. The same procedure used for email bodies can therefore be applied to subjects with some variations: in general, the problem described in Section 3.1 becomes less significant, and the similarity term in the Formula 2 plays a much smaller role due to the reduced length of the field considered.

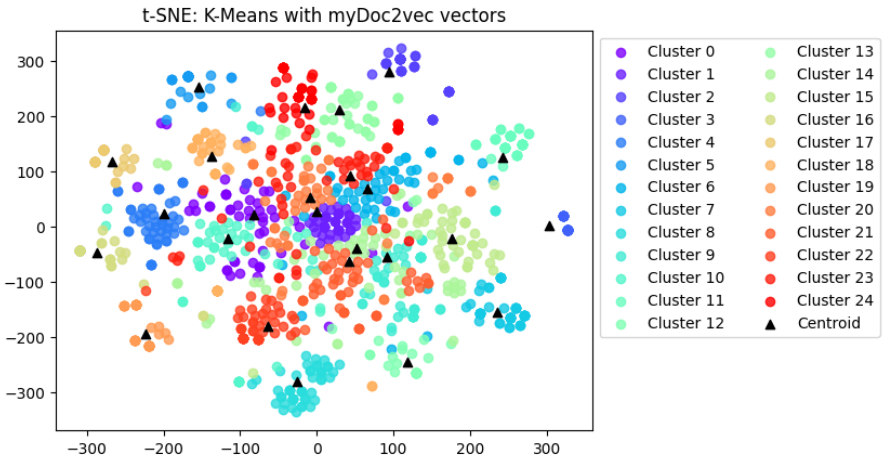


Figure 10: Projection of 50 emails for each category plus the centroids of each cluster.

Furthermore, it is not possible to get good word vectors by training a Word2vec network on such short texts, therefore, it is better to continue training the model on email bodies as well, or to use pre-computed vectors. Bodies and subject lines can be combined and used together to identify topics, but due to the brevity of the former, if simply "appended" to the email body, the subject risks having an extremely limited impact on the classification.

4 Concluding remarks

In conclusion, the proposed model has proven to be capable of extracting topics from the dataset and correctly classifying at least some of the emails. Of course, given the simplicity of the solution, the results are not comparable to those that much more complex models could produce; moreover, all of this can be seen as a simple preliminary study: many more tests would need to be conducted in order to obtain an accurate assessment of the real performance, ideally on supervised datasets. Many improvements can then be made: the formula used for computing the weights could be modified, new terms could be added, or it could be replaced by an AI model, like a neural network, to obtain better weights for the tokens. Since the current implementation is designed to use word vectors generated by Word2vec, tokens not present in its vocabulary must be discarded; by using other models, such as *fastText* [1], this limitation could be overcome. As for computation time, the greatest benefits would certainly be obtained by parallelizing the preprocessing phase, which turns out to be by far the most time-consuming.

Acknowledgments

Thanks to Giulia and Luca for patiently labeling the email samples used for the confusion matrix.

References

[1] Piotr Bojanowski et al. "Enriching Word Vectors with Subword Information". In: *arXiv preprint arXiv:1607.04606* (2016).

[2] FERC. *Enron's Energy Trading Business Process and Databases*. 2020. URL: https://www.ferc.gov/electric/industry-activities/addressing-2000-2001-western-energy-crisis/enrons-energy-trading-business-process-and-databases (visited on 2024).

[3] Maarten Grootendorst. "BERTopic: Neural topic modeling with a class-based TF-IDF procedure". In: *arXiv preprint arXiv:2203.05794* (2022).

[4] Scikit-learn. *TF-IDF Vectorizer*. 2024. URL: https://scikit-learn.org/1.5/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html (visited on 2024).

[5] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. "How to Use t-SNE Effectively". In: *Distill* (2016). DOI: 10.23915/distill.00002. URL: http://distill.pub/2016/misread-tsne.