Information Processing and Retrieval

Part 2 Project Report

Group 08:
Daniele Avolio, ist1111559
Michele Vitale, ist1111558
Luís Dias, ist198557

1 Problem statement

In the same dataset context as Part I of this project, we are now addressing the tasks of clustering and classification with both an unsupervised approach and a supervised one, using summaries that are being provided for each document as reference.
Tasks conducted in this second part are:

- **Part A: Clustering;** for each document, the goal is grouping sentences based on their features and similarities. With this done, it is easy to select the most relevant sentences based on some criteria and algorithm that we defined.
- **Part B: Classification;** given a document, the goal is to split it into sentences and, using a binary classifier, define wether each sentence belongs to a summary or not.

This report can not contain all the data and graphs that we produced, so for more complete informations it is strongly suggested to check the comments on the provided notebook.
Some tasks were again very intensive in term of computation, so we decided to not use BERT embedding representations, since it would increase by a lot the time needed to run the code. Thus, our attention was on space representation using TF-IDF.

2 Adopted solutions

Part A: Clustering

This part is conducted in an unsupervised approach, with the idea of grouping sentences with clustering algorithms. In particular, we used the **sklearn** library, with the AgglomerativeClustering class as suggested.
The main paths to explore in this part are:

- **Number of clusters and used metrics:** the main challenge here was the correct choice of the **number of clusters**, because it's a parameter than could have a big impact on the results. Using **silhouette score** we have solved this problem in a iterative way.
- **Sentences selection:** the second challenge was to select the sentences that would be used to build the summary. Using **centroids** of each clusters, we were able to build summaries that had more topics and were more representative of the original text.

Number of clusters and used metrics

A problem related to part A is to define a good number of clusters to represent the feature space of the document.
In this part, we tried to construct a custom metric taking into account Silhouette score, Calinski-Harabasz score and a function of the number of clusters. The idea was to consider the number of clusters, that can vary in the range $[2, numberOfSentences]$, to avoid high sparse clusters representation with single-sentence clusters, but in the end we noticed that the silhouette score by itself was performing overall better.
We are leaving the code for the custom metric in the notebook, but it is commented since it was not used in the final version of the code.

Sentences selection

Another relevant problem, once completed the clustering part of the project, was to decide the criteria to pick the sentences to use in the summarization. We decided to use the **centroid** of each cluster and, based on the distance from it, we pick the required number of sentences. We have done some tests on different criteria, but others ideas that we had were not really convincing on summaries, so we kept the centroid distance as metric.
It is important to note that this strategy has a limitation, since with high numbers of sentences picked from each cluster there might be some redundancy in summaries. Nevertheless, with a low number of picked sentences it is leading to convincing results, taking into the summary relevant sentences.

Part B: Classification

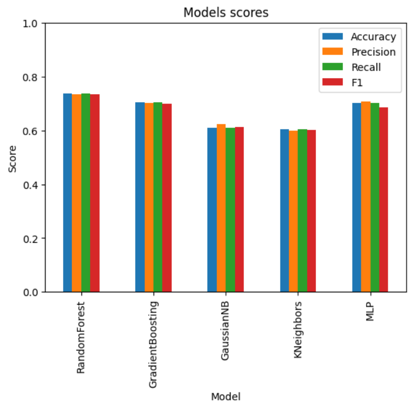
In this section the task is to create a binary classifier that has as goal to discern if a sentence should belong to the summary or not. We used models from the **scikit-learn** library, and in particular the machine learning techniques that we exploded are Random Forest, Gradient Boosting, Gaussian Naive Bayes, K-Neighbors and Multi-layer Perceptron.
Before the training phase, the challenge was to find the best features and the correct shape to represent data and build the models. Since the classifier should classify sentences, each row of our dataset represents one sentence in the original library. We ended up with the following structure.

similarity	n_sentence	n_words	n_stopwords	n_keywords	length_of_sentence	tfidf_score
position_in_doc	category	n_nouns	n_verbs	n_adjectives	n_adverbs	id
summary(target)						

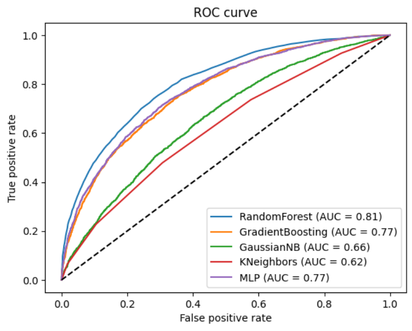
We then analyzed the features with a correlation matrix, the Shapley value of each features via the **SHAP** library and the Scikit-learn built-in feature importance. This process lead us to drop some features, with the following schema being the one used to train our models.

similarity	n_sentence	n_words	n_stopwords	n_keywords	length_of_sentence
tfidf_score	position_in_doc	n_nouns	n_verbs	n_adjectives	n_adverbs
category_business	category_entertainment	category_politics	category_sport	category_tech	Summary(target)

At this point, we could train our models and evaluate them with the common machine learning evaluation metrics. The one performing better is, in our case, RandomForest.
More on this can be found in the notebook, as well as in the following chapter.



(a) Models scores comparison



(b) Models ROC curves comparison

Figure 1: Descrizione generale delle immagini

3 Proposed questions

Part A: Clustering

3.1 Question 1

Do clustering-guided summarization alters the behavior and efficacy of the IR system?

To answer this question we ran the **clustering based** algorithm using the same set of documents used in the *first part of the project*. The result shows a pretty big difference between the two approaches.

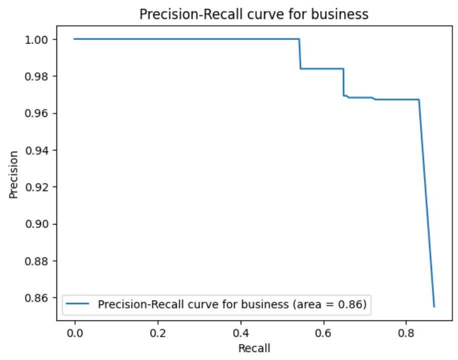


Figure 2: First Part Approach

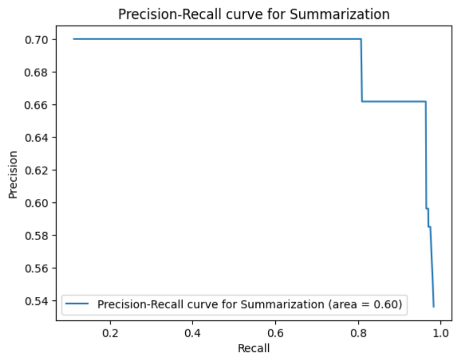


Figure 3: Clustered Approach

This result doesn't guarantee that the **clustering based** approach is worse than the approach using TFIDF and BM25, because this is based on our personal implementation of the algorithm. However, it is clear that the clustering approach is not as effective as the first approach in this case. A possible way to improve the algorithm could be to consider other sentences choices rather than focusing on the distance from the centroid of each cluster.

3.2 Question 2

How sentence representations, clustering choices, and rank criteria impact summarization?

We benchmarked the performance of the clustering algorithm using a set of metrics:

Max Clusters	Number of Sentences	Our Metrics
2	3	cosine euclidean
3	5	
4	7	
6	9	
8	11	
10	13	

We didn't include any **different representations** because we only used the **TFIDF** representation. The result are indicating a very low performance of the algorithm using a specific set of metrics.

Table 1: Results of the clustering algorithm using different metrics

#cluters	**#sentences**	**metric**	**avg_prec**	**avg_rec**	**f1**	**m_a_p**
2	3	cosine	0.453504	0.449012	0.451247	0.646069
2	3	euclidean	0.453504	0.449012	0.451247	0.646069
2	5	cosine	0.457060	0.633170	0.530891	0.561392
2	5	euclidean	0.457060	0.633170	0.530891	0.561392
2	7	cosine	0.469898	0.768889	0.583312	0.492352
...	...	...	...	...	...	...
10	9	euclidean	0.484872	0.934921	0.638568	0.418472
10	11	cosine	0.486431	0.941682	0.641495	0.344811
10	11	euclidean	0.486431	0.941682	0.641495	0.344811
10	13	cosine	0.486635	0.943571	0.642110	0.336604
10	13	euclidean	0.486635	0.943571	0.642110	0.336604

More insight on this can be found in the *notebook* file.

■ 3.3 Question 3

Are anchor sentences (capturing multiple topics) included? And less relevant outlier sen- tences excluded? Justify

Since our algorithm is based on the **distance from the centroid** of each cluster to select the sentences, we are not able to handle the **anchor sentences** and the **outlier sentences**. We are not able to give a clear answer to this question, but a possible way to handle this could be to consider the **distance from the centroid** and the **distance from the other sentences** inside other clusters. Sentences that are **more far** from the centroid of the cluster could be very **relevant** and could be considered as **anchor sentences**, since that sentence could be holding information between more topics.

■ 3.4 Question 4

Given a set of documents, plot the distribution of the number of keywords per document. Are keywords generally dissimilar? If not, how would you tackle this challenge? For this question we decided to use documents from **500** to **700** as range. The result shows that the distribution of the number of keywords per document is not uniform.

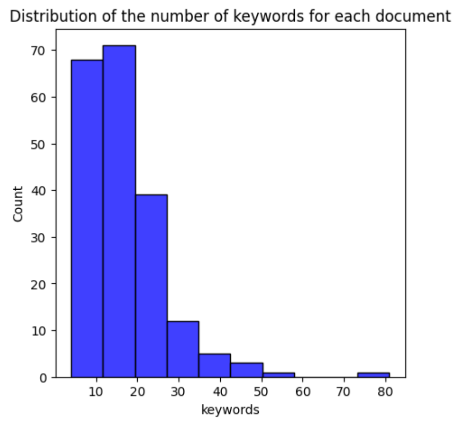


Figure 4: Distribution of the number of keywords per document

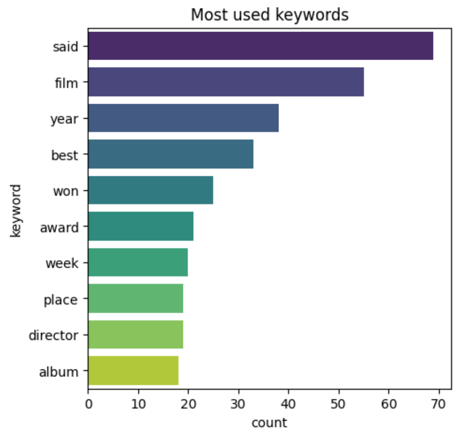


Figure 5: Distribution of the number of keywords per document

The main problem is that the **keywords** are not **dissimilar** and this could be a problem for the **clustering algorithm**. In fact, we can see that a lot of keywords are repeated in the documents. A possible way to tackle this challenge could be to use a **different representation** of the sentences in the space. Moreover, using multiple documents of the same category could help to improve the performance of the algorithm since the keywords are more likely to be repeated in the same category.

Part B: Supervised IR

3.5 Question 1

Does the incorporation of relevance feedback from ideal extracts significantly impact the performance of the IR system? Hypothesize why is that so.

We can say that a **model** that is trained using the **relevance feedback** from the **ideal extracts** could be a nice approach **only if** the relevance feedback is **correct**. This is because the model task is to follow the indications given by the guidance, so if the guidance is wrong, the model will be wrong too. That's not a rule for this project, but it's a general rule for machine learning models.

3.6 Question 2

Are the learned models able to generalize from one category to another? Justify.

To answer this question, we treid to train the model using the **tech category** as a training set, and all the other categories as a test set. We used *Random Forest* as a model, since it was the best between the others. The results show that the model achieved ≈ 0.69 as it's best performance and ≈ 0.65 as it's worst performance. This result is not bad, but it's not good either. This happens because each category is different from another, and the set of features that we are using may not be the best. However, to *effectively test the model*, what we could do is to train another model using a training set of the same size as the *tech category training set*, but using a mixture between all the categories. In this way we could have a better vision of the model's performance in the generalization task.

3.7 Question 3

Which features appear to be more relevant to the target summarization task? Do sentence- location features aid summarization?

The most important feature that we found is the score given by the *similarity* between the sentence and the document. To get this insight, we used a library called *SHAP* that helps us to understand the importance of each feature in the model. It uses the concept of *Shapley Values*, a concept from game theory that given a model and a set of features, it assigns to each feature a value that represents how much the feature contributes to the model's prediction.

To answer the second part of the question, we can say that the *sentence-location* feature is a good one, because it the 4th most important.

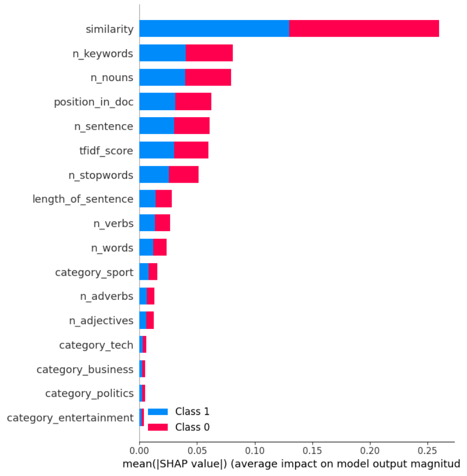


Figure 6: SHAP values for the Random Forest model

3.8 Question 4

In alternative to the given reference extracts, consider the presence of manual abstractive summaries, can supervised IR be used to explore such feedback? Justify. This task might result difficult to be conducted with a machine learning approach, since there is no direct access to a target function for sentences, neither a way to build up a summary from a set of features without human supervision. This second fact makes the only possible approach to summarization being the extractive one.
The task of supervised summarization can not be conduced as we did during the part B of this delivery, since we can not construct a dataset that use sentences as features.
The abstractive approach is a summarization based on the context of the document, trying to reproduce the same concept in a different shape. This ends up in sentences from the text not being in the summary and, thus, the dataset we have created can not be replicated in the exact same shape.

A possible solution to the task with abstractive summaries could be to use the deep learning BERT, a bidirectional encoder based on Transformer architecture that is able to transpose the input text into a latent space, in which similar concepts are close to each other.
i9 Exploiting this, our suggestion is to use the BERT model to encode both the summary, that serves as an anchor reference as provided ground truth, and the sentence we want to classify. The resulting embeddings can be used in a classifier or to compute other metrics, like distance ones, and then use those features to select sentences based on a criterion. This will still produce an extractive summary.

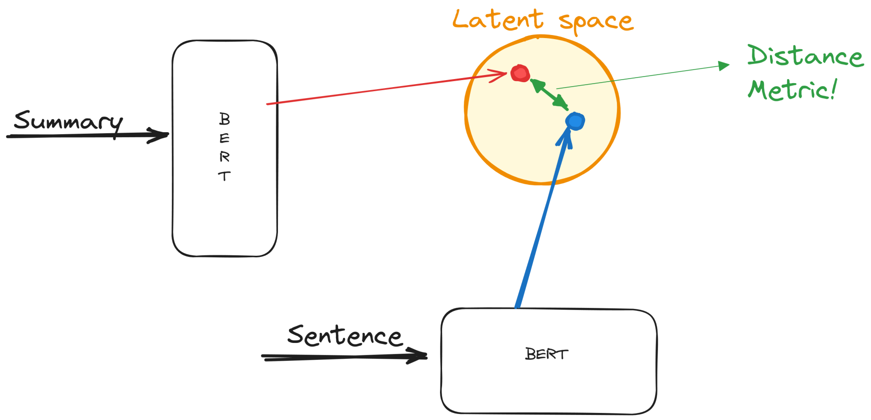


Figure 7: Possible architecture using abstractive summaries

Another approach that can be explored is based on keywords from the summary. This can be divided in two steps: during the first one, we construct a reference set of keywords, extracting them from the whole set of summaries and

expanding them with one iteration on a predefined thesaurus, to make sure to have also synonims of relevant words.
In the second phase, a sentence can be classified calculating a score which is the rate of number of keywords from the sentence that are present in the summary-extracted ones (also referred as **keywords batch** from now on) over the number of keywords in the document that contains the sentence, as it follows.

$$score = \frac{|keywordsFromSentenceInBatch|}{|keywordsInDocument|}$$

Note that this formula has the number of keywords in the document as denominator to prefer, among the same document, sentences with a higher keywords ”density”.
Once calculated the score, it is possible to use it to select the sentences that will be part of the summary using a fixed threshold, that can eventually be updated based on a feedback system that collects them from the final user. Here it is presented a pseudo code to better explain the algorithm.

```
keywords_batch = get_keywords_from_summaries()

for sentence in document:
  score = calculate_score(sentence, document, keywords_batch)
  if score >= threshold:
    summary.add(sentence)

return summary
```