PRI Project Report

Group 09

Dinis Araújo, ist186406 dinisaraujo@tecnico.ulisboa.pt
Inês Lacerda, ist186436 ines.lacerda@tecnico.ulisboa.pt
Maria Duarte, ist186474 maria.a.duarte@tecnico.ulisboa.pt

Keywords
Unsupervised Learning, Supervised Learning, Clustering, Classification, Naive Bayes, Random Forest, PageRank, IR evaluation.

1. INTRODUCTION

This is the report for the second delivery of the PRI course project. This delivery is divided into three sections: Clustering approach: organizing collections; Supervised approach: incorporating relevance feedback; and Graph ranking approach: document centrality.

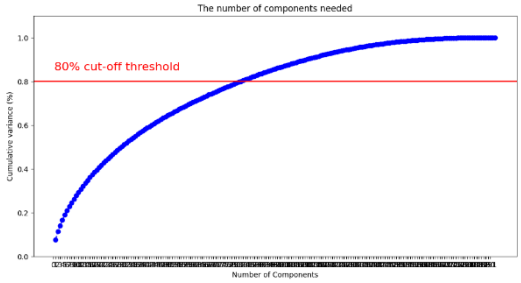
2. CLUSTERING APPROACHES

For this section, the primary goal is to understand the organization of topics in the provided topic collection and the organization of documents in the *RCV1* collection. The analysis is done using the *Dtrain* documents that are classified as non-relevant and relevant in the *Rtest* collection instead of the full *RCV1* collection, due to efficiency constraints. There are four clustering approaches: Partitioning, Hierarchical, Density-based and Model-based. Due to time constraints, we decided to implement the first three.

2.1 *Clustering* Function

2.1.1 Content Optimization

Before executing each Clustering approach, we applied first the *TFIDF* that calculates the weight of a word across a set of documents. After, we applied a *PCA* that reduces the dimensionality of the documents while minimizing information loss by retaining most data variation. To avoid overfitting, we can find the ideal number of components to include in the *PCA* by adding the cumulative variance ratio for each component until it reaches 80%, as shown in the **Figure below**.



2.1.2 Partitioning Approaches

We implemented three Partitioning Approaches: K-means, K-medoid and Affinity Propagation.

2.1.2.1 K-means

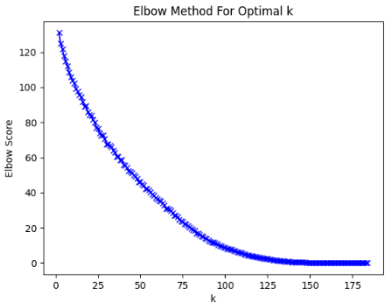
K-means is a vector quantization method that assigns samples to clusters with the nearest mean (i.e., cluster centroid or center).

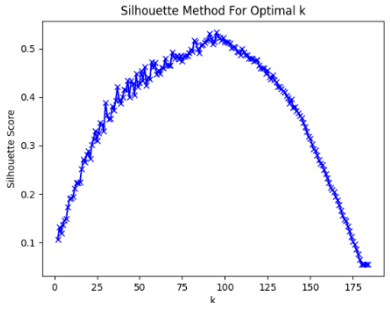
2.1.2.2 K-medoids

K-medoids is a variant of k-means and uses the most centrally located document instead of the mean point as centroid.

2.1.2.2.1 Elbow and Silhouette methods

To find the optimal number of clusters for k-means and k-medoids approaches, two methods were used: Elbow and Silhouette methods. The **Elbow method** calculates the squared error distances from samples to their closest centroid. The optimal number of clusters is represented by the elbow curve, as shown in the **Figure below in the left side**. However, this method may be ambiguous and so we decided to also use the Silhouette method. The **Silhouette method** measures how similar a sample is to its own cluster. The higher the Silhouette score, the more samples are placed in the correct cluster, thus, the optimal number of clusters is the one with the highest score, as shown in **Figure below in the right side**.



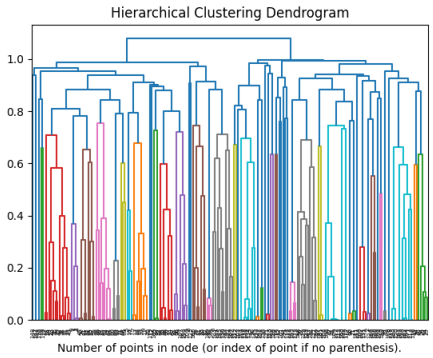


2.1.2.3 Affinity Propagation

Affinity Propagation uses a graph-based approach that exchanges messages between data points until a high-quality set of exemplars (i.e., representative of clusters) is obtained. Unlike k-means and k-medoid, AP does not need a specific number of clusters as input. It finds the optimal number of clusters without needing any additional methods. However, sometimes AP does not converge and the clusters' centers become an empty array, this is a problem we encountered when using the AP approach from the *sklearn.cluster* package.

2.1.3 Hierarchical Approach

We implement the Agglomerative Clustering Approach. Agglomerative is a bottom-up approach that treats each sample as a singleton cluster, and then merges pairs of the most similar clusters into one big cluster, as shown in the dendrogram in the **Figure below**.



To compute the similarity between samples, we decided to use the cosine distance metric. We chose to use an average linkage, because it is the most flexible as it accounts for the mean distance between all points. To find the optimal number of clusters, we created a linkage matrix based on the distance information between each sample and used the *fcluster* function that forms flat clusters based on the hierarchical clustering.

2.1.4 Density-based Approach

We implement the HDBSCAN approach which is a variant of the DBSCAN approach. HDBSCAN transforms the space according to its density and performs a single linkage based on that transformed

space. Similarly, to the AP approach, HDBSCAN finds the optimal number of clusters without needing any additional methods.

2.2 Hypothesized number of clusters

We gathered, for each clustering approach, the average number of clusters for the topic and document collections using three subsets (i.e., 20, 50 and 100 topics and documents categorized as relevant and non-relevant for those topics).

2.2.1 Topics

Approaches	20_topics	50_topics	100_topics
K-means	8.75	23	45.31
K-medoids	8.94	22.44	44.63
AP	4	10	24.95
Agglomerative	14	34	57
HDBSCAN	4	5	15.59
AVERAGE	**7.9**	**18.9**	**37.5**

Based on the average of all approaches, we can conclude that the hypothesized number of clusters for 20 topics is 7.94, for 50 topics is 18.9 and for 100 topics is 37.5.

2.2.2 Documents

Approaches	20_subset	50_subset	100_subset
K-means	132.38	244.25	523.13
K-medoids	133.88	249.25	500.5
AP	59.91	103.81	204.27
Agglomerative	136	220.25	540.5
HDBSCAN	83.5	151.72	283.58
AVERAGE	**109.1**	**193.9**	**410.4**

Based on the average of all approaches, we can conclude that the hypothesized number of clusters for the documents associated with 20 topics is 109.1, associated with 50 topics is 193.9 and associated with 100 topics is 410.4.

2.3 Clusters' cohesion and separation

To evaluate the Cluster's cohesion and separation, we calculated, in the *Evaluate* Function, the *Silhouette* score over all samples in the collections. The ***Silhouette* score** is an internal measure that computes the similarity of a sample to its own cluster (i.e., the cohesion) compared to the other clusters (i.e., separation). Values near 1 indicate that the clusters are cohesive and nicely separated. Scores of less than 0 means that the samples belong to the wrong cluster and scores of 0 means that clusters are overlapping.

2.3.1 Topics

Approaches	20	50	100	Average	Average2
K-means	0.24	0.28	0.37	**0.30**	**0.38**
K-medoids	0.25	0.27	0.38	**0.30**	**0.36**
AP	0.16	0.17	0.29	**0.21**	**0.21**
Agglomerative	0.38	0.36	0.35	**0.36**	**0.65**
HDBSCAN	0.09	0.11	0.20	**0.13**	**0.36**

Based on the average of all subsets for each approach, we can conclude that Agglomerative is the one with the highest silhouette scores and HDBSCAN is the one with the lowest score.

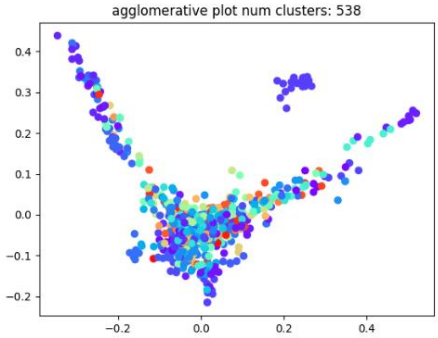
2.3.2 Documents

Approaches	20	50	100	Average	Average2
K-means	0.49	0.48	0.38	**0.45**	**0.57**
K-medoids	0.439	0.47	0.38	**0.43**	**0.56**
AP	0.40	0.38	0.31	**0.37**	**0.38**
Agglomerative	0.51	0.47	0.39	**0.46**	**0.69**
HDBSCAN	0.45	0.43	0.34	**0.41**	**0.62**

Based on the average of all subsets for each approach, we can conclude that Agglomerative is the one with the highest silhouette scores and AP is the one with the lowest score.

2.3.3 Conclusions

We hypothesize that Agglomerative is the one with the highest silhouette score, because it is more informative and good at discovering hidden structures, unlike the unstructured flat clusters returned by the other approaches. We hypothesize Agglomerative had a higher score than the partitioning approaches because Agglomerative clustering performs well when the variance of distribution of samples is non-spherical, which is the case as shown in **Figure below**.



We hypothesize HDBSCAN is the one with the lowest score in the topics collection because it only considers points that are close and assumes the rest is noise (i.e., outliers), thus, it fails to take into consideration the full dataset.

Another finding is that clusters with size 1 get a silhouette score of 0, therefore, there is a correlation between the average silhouette scores and the number of outliers. To confirm this hypothesis, we computed the average (i.e., column AVERAGE2 in the tables) silhouette scores for each approach without outliers. The AP approach had similar silhouette scores with and without outliers because it is robust to outliers, but the remaining approaches improved their scores as they are sensitive to outliers. HDBSCAN is the one with the highest improvement, because it supports the GLOSH outlier detection algorithm that detects outliers that are not necessarily global but may also be points in their local regions.

We can conclude that all approaches produce clusters that are cohesive and nicely separated, with Agglomerative clustering being the best approach with and without outliers.

2.4 Topics' organization and independence

For each topic subset we analyzed the average number of outliers and checked if any topics were in the same clusters for all approaches.

Approaches	20_subset	50_subset	100_subset
K-means	3.5	9.5	13.25

K-medoids	3.88	10.38	11.63
AP	0	0	0
Agglomerative	8	21	26
HDBSCAN	12	29	42.86

Shown in the **table above**, HDBSCAN is the one with the highest number of outliers, meaning that it assumes there is a higher independence between topics. On the contrary, AP considers there is dependence between topics.

Furthermore, we noticed that 11 clusters contained the same topics for all approaches, therefore, there are in fact highly similar topics. For instance, topics 163, 169, 185 and 186 were all in the same cluster for all approaches, thus, are considered similar topics with "community" and "european" as the most frequent words. This is an accurate presumption because all these topics are indeed about "European community". We can conclude that the topics are organized based on how similar they are.

2.5 *Interpret* Function

This function describes the documents in a specific cluster considering two criteria: median and medoid. The median criteria finds the top terms based on the TFIDF median values for all topics in the collection. The medoid criteria finds the topic index with the lowest mean distance to the remaining topics in the cluster.

2.6 Topics' medoids and representations

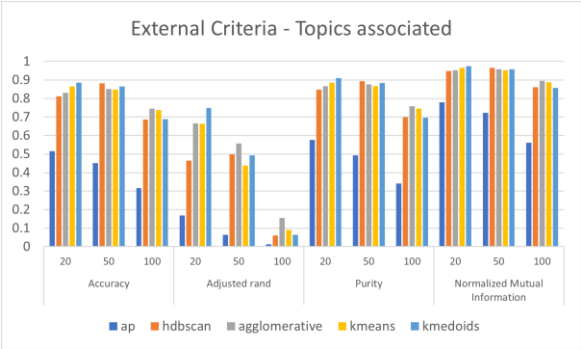
To analyze if the medoid adequately represents the remaining topics in a given cluster, we decided to use the median criteria. To do this, we gathered the top terms based on the TFIDF median values, and then found the topic index with the highest mean value for those top terms. This way we found the topic index that has the highest values for the most frequent terms, thus, it is the topic index that most adequately represents the remaining topics in the cluster. After, we compared the topic index found (i.e., based on the top terms) with the medoid index. To account for misleading evaluations, we removed the outliers. Furthermore, when there are only two topics in a cluster, the medoid could be any of those because the lowest mean distance would be the same and so we decided to choose the medoid index that matches the median index. We computed the accuracy for each approach by dividing the number of clusters that contain the most adequately represented medoid (i.e., medoid index and median index are the same) by the total number of clusters. The average accuracy for all approaches was 0.78, therefore, we can conclude that the medoids chosen adequately represent the remaining topics in the cluster.

2.7 Documents' organization

To analyze how the documents collection is organized we decided to evaluate the clustering approaches using external criteria. We computed 4 metrics: the accuracy, the adjusted rand index, the purity and the normalized mutual information. Before evaluating the clusters, we decided to remove all documents that were not associated with any topic and all documents associated with multiple topics. To evaluate the organization of documents and the performance of the clustering approaches, we decided to use the topics that the documents are associated with as ground truth. To do this, for each cluster, we checked if all documents belonged to the same topic, if so, their true label would be the cluster label, otherwise, their true label would be -2 which is a non-existing cluster label. Furthermore, we removed all outliers to fairly evaluate our clustering performances, because a cluster is classified as correct if all documents belong to the same topic, therefore, if the cluster is an outlier it would be classified as correct which in fact it is not entirely true.

Accuracy metric computes the fraction of right predictions. Adjusted rand index computes the similarity between the clustering and the ground truth considering all pairs of samples. If the score is 0 then the clustering and the ground truth do not agree on any pair of points and if the score is 1 the clustering and the ground truth are exactly the same. Purity computes the percentage of the total number of data points that were correctly classified. For each cluster, it assigns the most frequent class in the clustering and then divides the number of correctly assigned documents by the total number of samples. Normalized mutual information measures how much information is shared between the clustering and the ground truth.

2.8 Conclusions



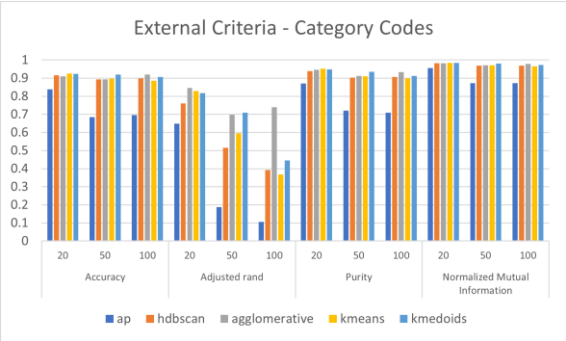
Based on the results we gathered, AP is the clustering approach that performs worst and HDBSCAN, Agglomerative and K-medoids are the approaches that perform best. A possible reason behind AP's low performance is that this approach is robust to outliers, thus, when evaluating, the sample used will always be higher than the sample used for the remaining models and so AP will be more prone to error. For each subset and all models, the average accuracy is 0.78 for the 20 subset, 0.78 for the 50 subset and 0.64 for the 100 subset, as shown in the **Figure above**.

To understand further the wrong classifications, we decided to analyze, for each wrong cluster, if all documents had at least one category code in common.

Approaches	**20_subset**	**50_subset**	**100_subset**
K-means	0	0.042	0.089
K-medoids	0	0.0059	0.0063
AP	0.086	0.294	0.29
Agglomerative	0	0	0.020
HDBSCAN	0.0703	0.04704	0.045

The **table above** shows, for all wrong classified clusters, the percentage of clusters that do not have at least one category code in common. Based on our analysis, we noticed that, even though the clusters have been wrongly classified in terms of associated topics, most wrong clusters contain documents that have at least one category code in common. We also decided to classify clusters solely based on documents having at least one category code in common. Again, AP performed the worst but the accuracy for all approaches increased significantly with an average of 0.90 for the 20 subset, 0.87 for the 50 subset and 0.86 for the 100 subset, as shown in the **Figure below**. The reason behind this is that now documents have a higher chance to be classified correctly because documents have many codes associated and codes in common even though they are not associated to the same topic. We can conclude

that the documents collection is organized based on the topics they are associated with and based on their category codes.



3. SUPERVISED APPROACH:

For this section of the project, we had to create a classification model. We started by implementing the three main functionalities and from there we answered each one of the questions to explore.

3.1 Performance of Classifier vs Boolean Model

Before we start comparing the performance of our IR System and the performance of the Baseline IR system, from the first delivery, let us explain how we implemented the new IR System.

3.1.1 Classification Models

We decided to create two different classification models: Naive Bayes and Random Forest.

In the *training* functionality, we created the model and then trained it on the train data (*Model.fit(X,Y)*). In the *classify* functionality we returned the output of the prediction done on the test data, by the model created and trained from the previous functionality. Finally, in the *evaluate*, we calculated the accuracy, error rate, precision, recall and f-score of the predicted labels (that we get from invoquing *classify)* for all topics.

In addition, in the training functionality for Random Forest we used *RandomizedSearchCV*, in order to get the best parameters, instead of just using the default values, like we did for the Naive Bayes model.

3.1.2 Performance

To compare the performance of our IR system with the baseline IR system (Boolean Model) we calculated the average precision and bpref of the models for the first 20 topics. The dataset we used includes all documents that are relevant to the first 20 topics.

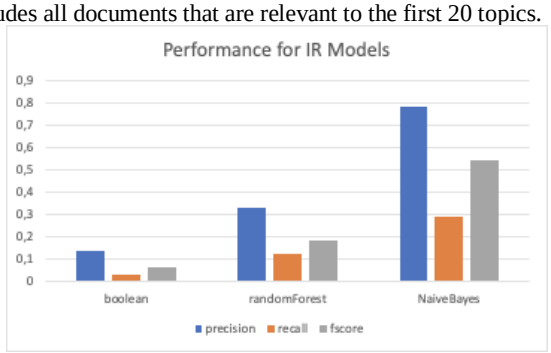


Figure 1. Performance of the Boolean Model, Random Forest and Naive Bayes

As can be seen in **figure 1**, both Random Forest and Naïve Bayes have a higher performance than the Boolean Model. This means incorporating relevant feedback aids the IR system. Our hypothesis on why that happens is that we are using supervised learning (classifiers) instead of unsupervised learning (Boolean). We are allowing our model to collect data and learn to correctly classify that data, so that it can then predict labels for new data. In short, we are optimizing the performance by using experience.

3.2 Classifiers Performance Across Topics

To compare the performance of our models across topics we calculated the performance of our models for each topic, using the dataset described in the previous subsection.

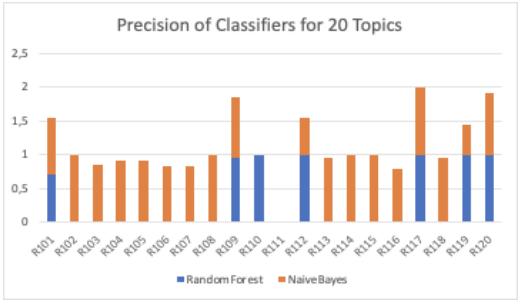


Figure 2. Performance of RF and NB for 20 topics

As can be seen in **figure 2**, we don't have a uniform performance across topics. There seems to be topics that are substantially harder to classify. Both Random Forest and Naive Bayes had zero precision for the topic R111, though Random Forest had zero precision on many other topics as well.

By looking only at Random Forest, we can see that it either had really good precisions (almost one) or really bad ones (0 precision). This is because for the topics with zero precision, there are probably a lot of documents with features that weren't present in the training data, and so the model did not learn how to correctly classify those documents.

Naive Bayes only had a bad precision for the topic R11, because it seems that this topic is complicated and probably very similar to documents not relevant to topic R11.

3.3 Random Forest vs. Naive Bayes

To compare the different classification algorithms (Naive Bayes, Random Forest) and the different parameterizations for Random Forest, we decided to calculate the precision, bpref and fscore using the dataset described in section 3.1.2

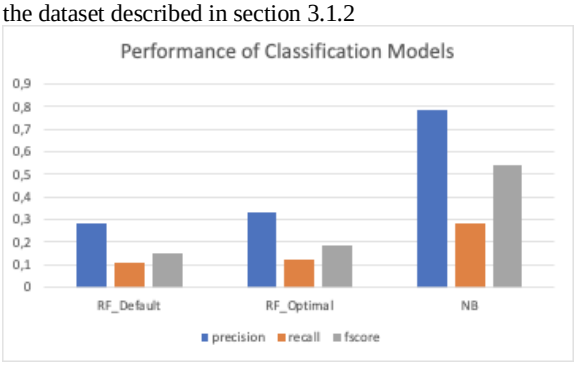


Figure 3. Performance of Naive Bayes and Random Forest (Default and Optimal)

As can be seen in **figure 3**, Naive Bayes has a much higher performance than Random Forest. Our hypothesis on why Random Forest is worse than Naive Bayes, is that Random Forest doesn't have the ability to extrapolate. Since our training data is not that large, the information from the test data could be out of the bounds from our training data. For instance, there could be terms in the test data not present in the train data that are relevant to the topic of that document.

Comparing RF_Default with RF_optimal we can conclude that finding the best parameters increases, even if not significantly, the performance of the system.

3.4 Extension Towards Ranking

Our classifier was only able to classify if a certain document was relevant or not to a topic. It wasn't able to rank documents. To extend our IR system to perform ranking, we used the probabilistic output (predict_prob) in our classifier, so we could rank the documents according to those probabilities. Then, we compared the performance (precision, bpref) of our ranking model with the

baseline ranking models (bm25 and space vector model), again using the dataset described in section 3.1.2.

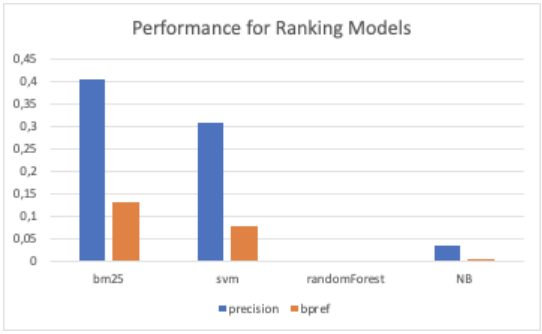


Figure 4. Performance of the following ranking models: BM25, SVM, RF, NB

As we can see in **figure 4**, using the classifiers for ranking does not help the IR system. With Random Forest we get a precision of 0, a low value was expected since just as a classifier, it already had a small precision, For Naive Bayes we have a precision of 0.03 and a bpref of 0.05. So also, very low results. We hypothesized that the reason why extending classification to ranking had such bad results, is that in cases where the classifiers are wrong, when they classify non relevant documents as relevant, they are giving a higher probability to those documents when compared to the real relevant ones. This causes the ranking to be incorrect, resulting in the low values we see.

3.5 Behavior of the IR with additional features

To see if incorporating additional features aid the IR system, we decided to apply PCA, since it reduces our feature space. Before comparing the performances of the models using PCA against the models without PCA, we decided to find the best number of components.

	Accuracy	Precision	Recall	F1
Naive Bayes with PCA	0.89	0.44	0.47	0.33
Random Forest with PCA	0.95	0.55	0.28	0.33
Naive Bayes	0.96	0.84	0.32	0.44
Random Forest	0.95	0.46	0.22	0.25

We ran our program for a different PCA number of components. We used a collection consisting of all the documents that were relevant to the first 10 topics. After analyzing the results, we found that 20 is the optimal value for both Naive Bayes and Random Forest.

After finding the optimal number of components we compared the performance of our models with and without PCA. While we were able to see a slight improvement in the Random Forest model, the performance of our IR system decreased on the Naive Bayes model, especially on the precision score. This is probably due to the fact that one or more least important terms (least important according to the PCA model) were actually improving Naive Bayes accuracy on our test subset.

4. GRAPH RANKING APPROACH: DOCUMENT CENTRALITY

For this section, we had to develop a program that used a PageRank-based approach for aiding the IR System. We started by implementing the two main functionalities and from there we answered each one of the questions to explore.

4.1 Performance of Baseline IR System Vs Graph-enriched IR System

Before we start comparing the performance of our Graph-enriched IR System and the performance of the Baseline IR system, from the first delivery, let us explain how we implemented the new IR System.

4.1.1 Graph-enriched IR System

Our IR System uses a graph ranking method. The ranking is done by the functionality *undirected_page_rank*, which is responsible for creating a graph (using the other functionality *build_graph*) and ranking the documents (nodes of the graph).

Let's start by discussing the *build_graph* functionality. In this functionality, first we represented the documents in vectors using the *TFIDF vectorizer*. Then, we created a graph (using the *networkx*), and we calculated the distance score between all pairs of documents, using the module *pairwaise_distances*. As we calculated each score, we added each node (that corresponds to a document), the edges between the documents whose distance score was less than the threshold, and their corresponding weights (scores). Finally, we returned the created graph.

From *pairwaise_distances* we used the following metrics: 1) *Euclidean*, since it gives us the distance between the two vectors (documents); 2) *Manhattan*, that gives us the distance between the vectors in a grid-like path. With high dimensionality spaces it is preferable to apply the L1 norm (Manhattan distance). This is due to the curse of dimensionality; 3) and finally *cosine*, which looks at the angle between two vectors, not taking into regard their weights or magnitude. If we consider that we are working with documents with uneven length, then for example, just because the term "Community" appears more in document 1 them in document 2 it does not mean that document 1 is more related to the topic "Community". It could simply be that document 1 is longer. So not taking into regard the magnitude of vectors would be beneficial.

For the metrics *Manhattan* and *Euclidean* we applied min max normalization on the distances scores before adding them as the weights of the edges.

Now let's discuss the *undirected_page_rank* functionality. We started by doing a pre-selection of *p* documents so we could build a graph for those documents. We developed multiple pre-selection methods using the different IR models developed in the first delivery (Boolean Model, Space Vector Model and finally BM25 Model). Then, after doing the pre-selection, we used the returned *p* documents and called *build_graph* to have a graph for those selected documents. We ranked the documents by applying the extended page rank algorithm, using *netwrorkx.pageRank*, with the weights being the weights of the edges of the graph and the priors being one of the following: uniform distribution, BM25, RRF, degree centrality, closeness centrality or betweenness centrality. We thought it would be interesting to see how using different ranking models or using centrality metrics would impact the performance of the system. We did not use the indegree and outdegree centrality since our built graph is undirected, so we simply duplicate all edges to make it a directional graph.

4.1.2 Performance

In order to compare the performance of our IR System with the baseline system, we ran multiple experiments using: the different pre-selection methods we implemented; different priors for the page rank algorithm; different similarity metrics for building the graph, and finally different threshold values for building the graph. Each experiment is structured in the following format: selection_priors_similarity_threshold.

Since the number of experiences was high, we decided to use a subset of all the documents to be our collection. We used a dataset with all the documents that are relevant for the first 50 topics.

After running all the experiments, we decided to compare the performance of our IR system with the baseline IR by analyzing the precision (to evaluate if the systems return relevant documents) and bpref values (to evaluate if the system prefers relevant documents to irrelevant ones). In the first delivery we saw that BM25 ranking model outperformed Space Vector Model and the Boolean Model. To see if using PageRank improved the ranking performance, we decided to compare the performance of each ranking model with the performance of the PageRank model using multiple combinations with the pre-selection method being the baseline ranking model.

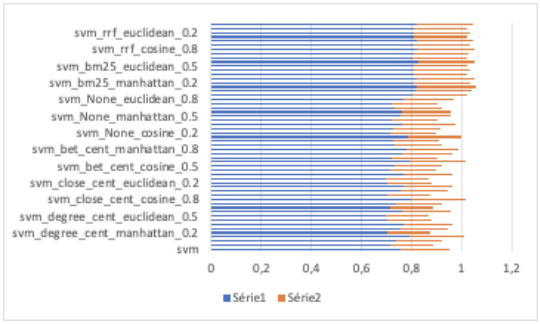


Figure 5. Performance using Space Vector Model as Pre-Selection

In **figure 5**, we can see that the space vector model achieves a precision of 0.73 and bpref of 0.2. Our best result from the graph-enriched IR system is 0.83 precision and 0.23 bpref when using the Euclidean distance metric and threshold value of 0.8 for building the graph and using bm25 scores for the priors of the PageRank algorithm. In the first delivery we saw that the BM25 Model outperformed the Space Vector Model, so it makes sense that the performance of our system using BM25 as the pre-selection method increases the performance even more. As we can see by comparing the values of figure 6 with the performance value of svm of figure 1.

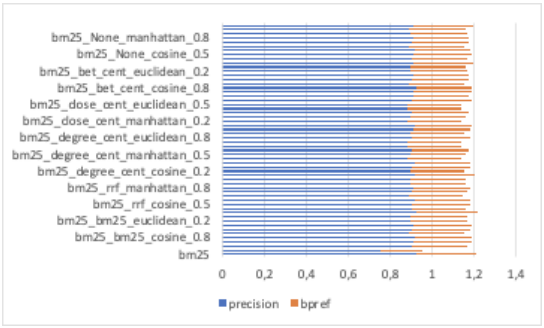


Figure 6. Performance using BM25 Model as Pre-Selection

In **figure 6**, we can see that the BM25 model achieves the best performance. So, it seems that in contrast to the Space Vector Model, the ranking performance does not improve when the selected documents are ranked by the PageRank algorithm instead of BM25.

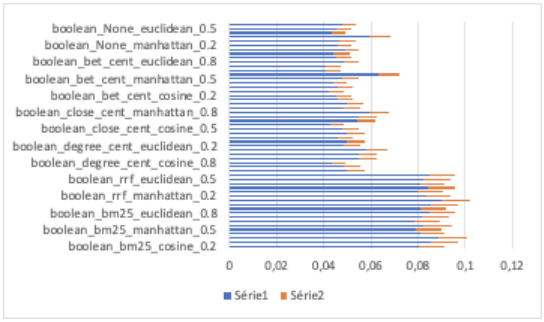


Figure 7. Performance using Boolean Model as Pre-Selection

In regard to the performance of our IR system when the pre-selection method used is the Boolean model, the system achieves the worst performance.

4.1.3 Conclusions

When using the Boolean Model for the pre-selection, the system achieved the worst performance. We created the following hypotheses to explain this situation: The PageRank algorithm consists of ranking nodes (in these context documents) having in mind the quality and number of links to each node. The Boolean models simply returns documents, for a specific topic, whose terms match the ones of that topic. In the first delivery we saw that this model didn't achieve a high performance. This is related to the fact that it doesn't calculate any similarity or distance metric between the topic and the documents. Thus, allowing the model to return documents that aren't relevant, and so the graph will have a higher number of "bad" nodes that could potentially be connected to other nodes that correspond to relevant documents, injuring the performance of the PageRank algorithm.

When using the Space Vector Model for the pre-selection, we saw that our system outperformed the performance of the ranking Space Vector Model. Even in cases where the priors of the PageRank are the uniform distribution ("none" in the label of the experience) the precision and bpref are still higher. We created the following hypotheses to explain this situation: Ranking the documents having in mind how similar or how not similar they are to the topic, instead of looking at each document equally (uniform distribution), helps the ranking, and that's why when the threshold increases (higher distance, more edges, and so more information), the precision and bpref increase as well.

When using the BM25 Model for the pre-selection, we saw that our system did not increase the performance of the ranking BM25 Model. So, ranking documents having in mind their similarity to the topic did not help the system, no matter the metric used. Still the difference of the performance was small. We created the following hypotheses to explain this situation: The experiments were run for 15 documents, so since bm25 always had a high precision, it might have returned documents all with very similar similarity scores, and so PageRank does not improve from using such values. The fact that bm25 is better than the rest, shows that bm25 scores are enough to rank documents. The fact that PageRank does not have better results than BM25 even with its scores, shows that by focusing on the quality and unit of links of each node it worsens the performance.

4.2 Ranking Performance for different θ values

From observing the previous figures of the previous subsection, we can see that for the same similarity metric, pre-selection method

and priors, the precision/bpref values tend to increase with the threshold.

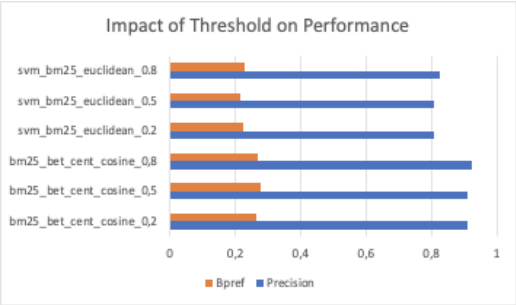


Figure 8. Performance when varying threshold value

In **figure 8** we can see some of those cases. This makes sense, since by increasing the threshold we are allowing more edges to be added to the graph, and due to this the PageRank algorithm will have more links to consider. So, if a document has a link with low values, the ranking it will give the document will be smaller than if the document was only linked to very high similar documents. This reflects the importance of not only knowing which documents are more similar but also which aren't.

4.3 Graph and Centrality Score Documents

To study the graph centrality scores of the documents in a graph, we built a graph using cosine distance metric and threshold value of 0.8 for the documents pre-selected using the BM25 model (with p=15 documents). Then, for each document in the graph we got the centrality score. *Networkx* has many different centrality measures, we decided to use the ones we mentioned in section 3.1.1.

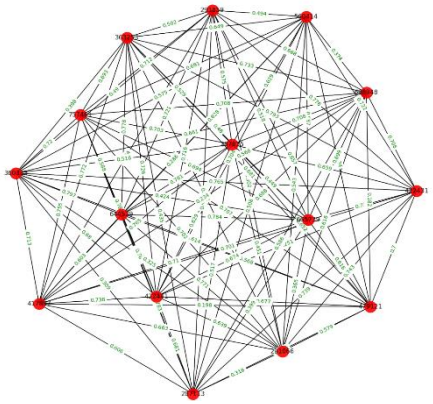


Figure 9. Graph representation

Let's start with the degree centrality scores: Most documents in the graph had a degree centrality of 1. Document 422431 was the one with the lowest degree centrality (0.86). This is because document 422431 is the node with lower degree (12), while all other documents who had a degree centrality of 1 have a degree of 14. So, degree centrality is simply related to the degree of the documents (nodes).

Moving on to the closeness centrality: Like before most documents in the graph have a closeness centrality of 1, with document 422431 having the lowest (0.875). This document is the one with the highest average of the shortest path length from him to all other nodes in the graph. Which is why its closeness centrality is the lowest, since closeness centrality just shows how close a node is to all other nodes in the graph.

Finally, let's see the betweenness centrality: Most documents have a betweenness centrality of only 0.0018, and again document 422431 has the lowest centrality score (0). The centrality measure reflects how much a given document (node) is in-between others. By looking at the graph, we can see that it's almost a complete graph. Due to this, from a certain node u, we can reach another node v, not needing to pass through other nodes (direct link). That's why the betweenness centralities are so low, even reaching zero for the document with less links.

4.4 Priors Performance

To analyze the impact of using different priors in the PageRank algorithm we decided to compare the performance using all the different priors mentioned in section 3.1.1.

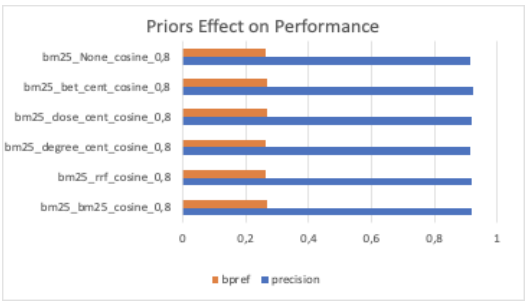


Figure 10. Performance using Cosine Distances

In figure 10, we can see that when using cosine and a threshold 0.8, there isn't a very significant difference from using one method over the other. Still using BM25 scores in the priors gives normally the best result. Followed by the RRF scores. When using the different distance metrics (other than cosine) we get similar results (see figures 11 and 12). This also happens for the different pre-selection methods (Boolean and svm). We hypothesized that the reason why using bm25 scores instead of the uniform distribution, helps the performance of our IR system, is related to the fact that the PageRank algorithm, with bm25 as priors, gives more importance to the nodes that are more similar to the topic in question, and doesn't treat every document(node) equally. Resulting in a higher ranking for the documents with higher priors.

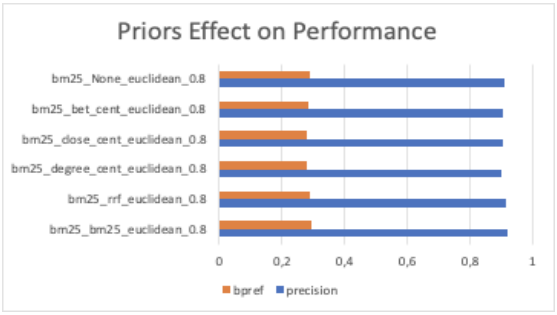


Figure 11. Performance using Euclidean Distances

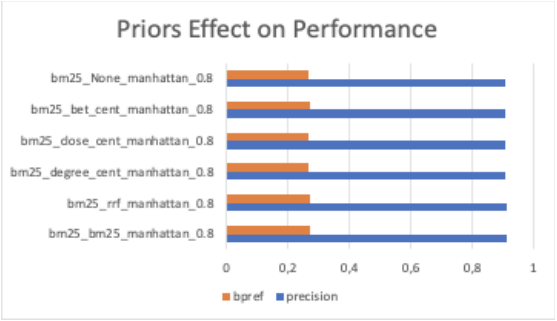


Figure 12. Performance using Manhattan Distance