Network Analysis for Information Retrieval

Alfathi Badia[1,2*] and Slimi Sarra[2,3†]

[1*]InfoStat sector, Icom, University Lyon2, 5 Av.Pierre Mendès, Bron, 69500, France.

[2]InfoStat sector, Icom, University Lyon2, 5 Av.Pierre Mendès, Bron, 69500, France.

*Corresponding author(s). E-mail(s): mbmabadie@gmail.com;
Contributing authors: sarraslimi0@gmail.com;
†These authors contributed equally to this work.

Abstract

This project focuses on analyzing a structured corpus through various functionalities including data loading, statistics display, corpus visualization, keyword-based search, clustering, and supervised classification. By leveraging structure and textual-based representations and machine learning techniques, we aim to gain insights into the corpus structure and content, enabling effective information retrieval and classification tasks.

Keywords: Structured corpus analysis, Data acquisition, Statistical analysis, Graph-based representation, Visualization, Keyword search, Clustering, Supervised classification, Information retrieval.

1 Introduction

In the realm of information retrieval and analysis, structured corpora pose both challenges and opportunities. This project endeavors to address these complexities by offering a comprehensive solution tailored towards efficient data acquisition, statistical analysis, and visualization. Beyond merely exploring the textual content, our approach delves deeper into the structural facets of the corpus, harnessing graph-based representations to unveil intricate relationships and patterns. Especially, this endeavor encompasses the implementation of a robust search engine, empowering users to query the corpus by keywords and retrieve relevant information. Moreover, we embark on

the journey of classification, leveraging advanced techniques to categorize documents based on their topics and content. Furthermore, employing clustering methodologies, we seek to uncover latent groupings within the corpus, shedding light on underlying themes and clusters of related documents.

2 Data comprehension and engineering

2.1 Data comprehension

The dataset comprises 999 rows, each representing a document. Each document is characterized by four columns:

- **ID**: This column contains unique identifiers for each document in the dataset. Each document is assigned a specific ID number.
- **Title**: The title column holds the titles of the documents. It provides a brief description or name for each document, giving readers an idea of its subject matter.
- **Content**: In this column, you'll find the main body of text for each document. It contains the actual content, such as paragraphs, or any textual information relevant to the document's topic.
- **Venue**: The venue column indicates where each document was published or presented. It could include information about journals, conferences, or any other platform where the document was made available to readers.

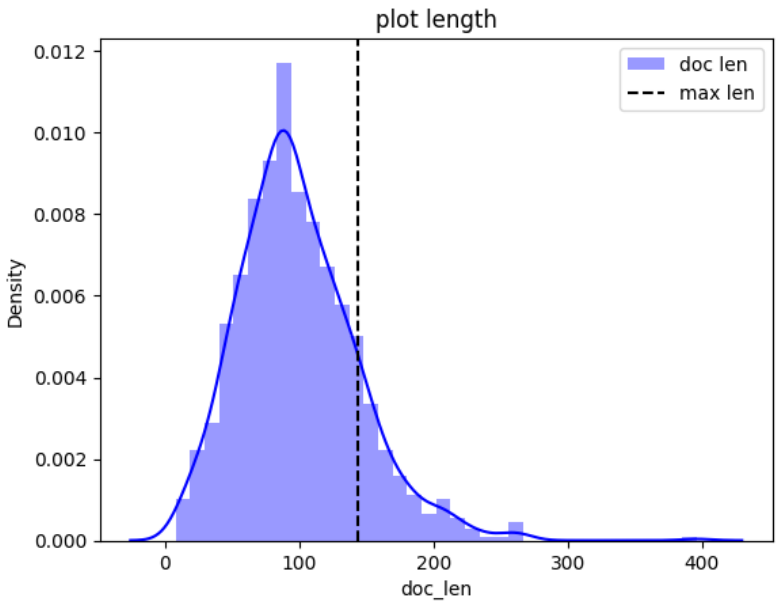

Fig. 1 Distribution of Documents length

The plot Figure 1 illustrates the distribution of document lengths within the dataset, revealing valuable insights into the data's characteristics. The majority of documents fall within the range of 80 to 110 words, as indicated by the peak of the distribution. This suggests a consistent length pattern among a significant portion of the documents. The mean document length is approximately 99 words, with a standard deviation of 44.54, indicating a moderate degree of variability in document lengths. The shortest document contains 8 words, while the longest extends to 396 words.

2.2 Data Engineering

In our data prepossessing phase, we employed NLTK to clean the textual content of the documents. The following steps were undertaken:

- **Removal of stopwords**: Commonly occurring words that do not carry significant meaning, such as "the", "is", and "and", were eliminated from the text to enhance the relevance of the remaining words.
- **Removal of extra whitespace and punctuation**: Extraneous whitespace and punctuation marks were stripped from the text to ensure consistency and improve readability.
- **Tokenization**: The cleaned text was tokenized, meaning it was split into individual words or tokens, to facilitate further analysis at the word level.
- **Lemmatization**: The NLTK WordNet Lemmatizer was applied to the tokenized text, reducing each word to its base or dictionary form. This helps in standardizing words and reducing the complexity of the vocabulary.

By performing these data engineering steps, we prepared the text data for subsequent analysis, ensuring that it is cleaned, standardized, and ready for further processing such as clustering and classification.

3 Search engine

The initial phase of our project focused on developing a search engine tailored for publications and documents. Users input their desired topic, and the search engine retrieves the top document with the highest similarity scores. This search engine aims to streamline information retrieval processes, providing users with quick and relevant access to documents aligned with their interests or queries.

3.1 Implementation

3.1.1 TF-IDF

Following the data cleaning process, which involved removing empty document rows, we proceeded to create a vocabulary set from the cleaned text data. Utilizing the Term Frequency-Inverse Document Frequency (TF-IDF) approach[1], we calculated the

[1]TF-IDF is a numerical statistic that reflects the importance of a term within a document relative to a collection of documents. It is computed as the product of two terms: Term Frequency (TF) and Inverse Document Frequency (IDF). TF measures the frequency of a term within a document, while IDF quantifies how rare a term is across all documents in the corpus. By combining these two terms, TF-IDF effectively

importance of each word in the documents relative to the entire corpus. Once the TF-IDF representation of the documents was obtained, we created vectors for the query or search keywords using the same TF-IDF transformation. Finally, we computed the cosine similarity between the query vector and the document vectors to identify the top documents closely related to the search query. This approach enables efficient and effective document retrieval, providing users with relevant documents ranked by their similarity to the search query.
Example case:
Number of top documents we need: 10 | Topic: 'Software Engineering and Systems'
The output >

Index	Title	Score
1	Towards Intelligent Distributed Computing: CE...	0.341740
309	Resource Specification Prototyping Human Inten...	0.311154
777	GPOP: Global File Popularity Measurement Unstr...	0.308590
867	Novel Single Fuzzy Approximation Based Adaptiv...	0.288796
585	Towards Systematic View Cybersecurity Ecology	0.287834
368	Hybrid Type II Fuzzy System and Data Mining Appr...	0.275043

Table 1 Top Documents for 'Software Engineering and Systems-TF-IDF'

3.1.2 Doc2Vec

To enhance our search capabilities, we explored an alternative method using the Doc2Vec model [2]. Specifically, we utilized the Doc2Vec model with the following specifications: dm=1, vector size=300, window=10, hs=0, min count=10, dbow words=1, and sample=1e-5. This model was trained on our document corpus to learn distributed representations of documents. Subsequently, we employed the trained model to infer document vectors for the entire corpus. By comparing these inferred document vectors, we identified the most similar document vector to the query vector, thereby retrieving the top documents closely related to the search topic.

First Example case
Number of top documents we need: 3 | Topic: 'Software Engineering and Systems'

Index	Title	Score
314	Template-Based Generation of Semantic Services	0.99
822	TAPAS: A Robotic Platform for Autonomous Navigation in Outdoor Environments	0.99
894	Using Dependency Analysis to Improve Question Classification	0.98

Table 2 Top Documents for the Topic 'Software Engineering and Systems' - Doc2Vec

highlights terms that are both frequent within a document and rare across the entire corpus, making it a powerful tool for information retrieval and document similarity computation.

[2]In Doc2Vec, each document is represented by a fixed-length vector, capturing semantic information about its content. By training on a large corpus of documents, Doc2Vec can generate vector representations that encode document semantics and facilitate various downstream tasks, including document similarity computation and information retrieval.

Second Example case
Number of top documents we need: 3 | Topic: 'Computer Vision'

Index	Title	Score
822	TAPAS: A Robotic Platform for Autonomous Navigation in Outdoor Environments	0.99
661	Deep Venous Thrombosis Identification from Analysis of Ultrasound Data	0.99
746	Global Regularity of the 2D Magnetic Micropolar Fluid Flows with Mixed Partial Viscosity	0.99

Table 3 Top Documents for the Topic 'Computer Vision' - Doc2Vec

4 Documents Classification

To assign a class to each article, we need to perform text classification. Considering that the articles do not have predefined classes, we have decided to pursue two main approaches to accomplish this task.
The first approach involves manually assigning classes to a subset of the data and then applying supervised classification. The second approach involves conducting unsupervised classification (clustering), which we will discuss in the next session.

4.1 Manual Labeling

We have a total of 1000 articles, of which 180 articles are missing the 'venue' feature in the data. Therefore, we only need to process the 820 articles that have the 'venue' feature.
Since the articles do not have classes, we have opted to manually assign classes to 217 articles from the entirety of our database, which contains 1000 articles, in order to classify the remaining 603 articles. The question arises: how were the classes chosen? The first step was to resort to Google Scholar to identify possible article categories. We found 55 different subcategories, but this still constitutes a large number, and given the small size of our database, we cannot work with such a large number of classes. Therefore, to reduce this number, we asked ChatGPT to group these subcategories into 7 classes, resulting in the following proposed classes:

- Theoretical and Fundamental Computer Science
- Artificial Intelligence and Machine Learning
- Information and Communication Technologies (ICT)
- Computer Vision and Image Processing
- Robotics and Embedded Systems
- Cyberscience and Social Networks
- Software Engineering and Systems

To avoid manually labeling 217 articles, we opted to use an API from OpenAI within Excel through a VBA script. This API allows us to provide a prompt containing the content of the "Venue" feature and ask which class best matches the venue.

```
Sub ClassifyArticleVenue()
    Dim http As Object, URL As String, response As String, apiKey As String
    Set http = CreateObject("WinHttp.WinHttpRequest.5.1")
    apiKey = "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"
    URL = "https://api.openai.com/v1/chat/completions"


    For Each cell In Range("D2:D1000")
        If cell.Value = "" Then GoTo NextCell

        Dim requestBody As String

        Dim prompt As String
        prompt = "Voici une liste de classes: Theoretical and Fundamental Computer Science, Artificial Intelligence and Machine Learning,
                "Étant donné la venue """ & cell.Value & """, quelle classe lui correspond le mieux?"

        requestBody = "{""model"": ""text-davinci-003"", ""prompt"": """ & prompt & """, ""temperature"": 0.7, ""max_tokens"": 60}"

        http.Open "POST", URL, False
        http.setRequestHeader "Content-Type", "application/json"
        http.setRequestHeader "Authorization", "Bearer " & apiKey
        http.setTimeouts 5000, 20000, 30000, 30000
        http.Send (requestBody)

        response = http.responseText
        Dim json As Object
        Set json = JsonConverter.ParseJson(response)

        Dim classe As String
        classe = json("choices")(1)("text")


        cell.Offset(0, 1).Value = classe

NextCell:
    Next cell
End Sub
```

Fig. 2 VBA Script for labeling

However, this approach was unsuccessful because the OpenAI API is paid, and as a result, we did not have access to this API.
Consequently, we resorted to manually assigning classes by asking ChatGPT-4 each time which class matches the venue, and then adding ChatGPT's response manually to our Excel file.

After following this approach, the 217 articles were labeled with the seven different classes as shown in the figure 3

4.2 Applying Classification models

Before applying the classification models, the labeled dataset needs to be divided into 3 sets: a training set (151 observations), a test set (42 observations), and a validation set (24 observations).

In this section, we tested and evaluated several algorithms, which are as follows:

- **Gradient Boosting:** We trained an ensemble model of 100 decision trees (estimators) available in the « scikit-learn » library.
- **Decision Tree :** We considered applying the « DecisionTreeClassifier() » available in the « scikit-learn » library.
- **Random Forest :** We trained the « RandomForestClassifier() » model available in the « scikit-learn » library.
- **K Neighbors :** We trained the linear model « KNeighborsClassifier() » model available in the « scikit-learn » library.

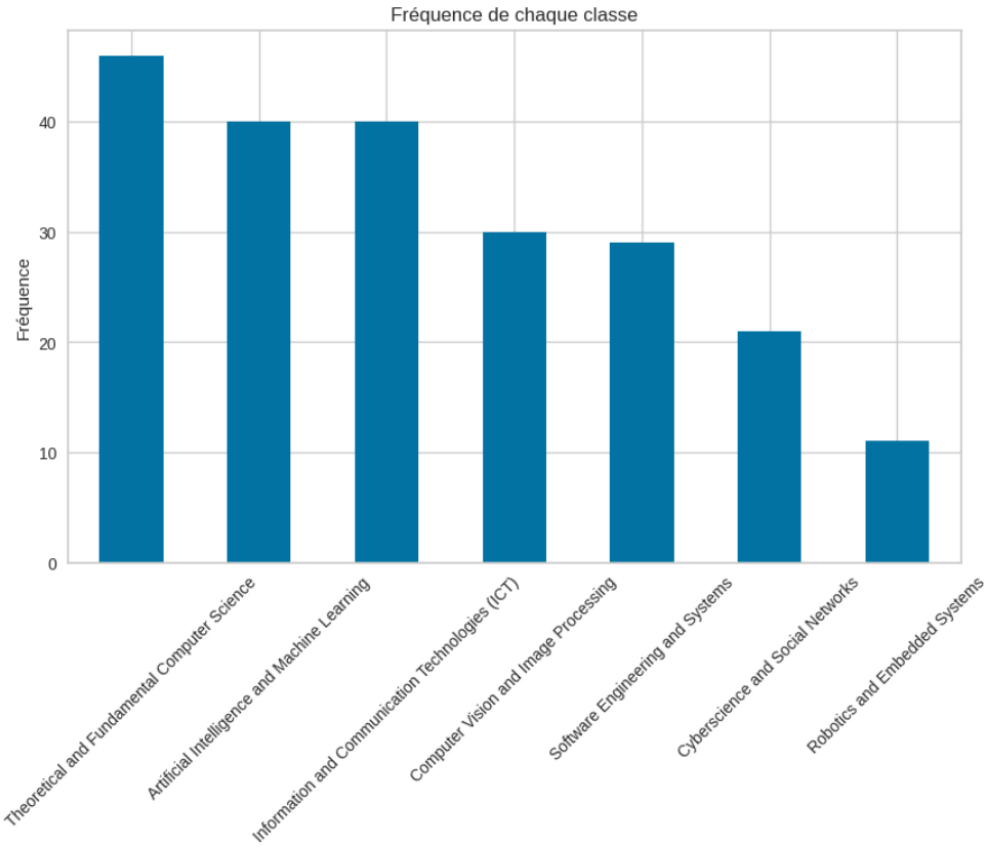

Fig. 3 Class's Distributions

Model	Acc	f1(Class1)	f1(Class2)	f1(Class3)	f1(Class4)	f1(Class5)	f1(Class6)	f1(Class7)
Gradient Boosting	0.71	0.86	0.86	0.40	0.67	1.00	0.75	0.60
Decision Tree	0.75	0.86	0.86	0.40	0.86	1.00	0.75	0.67
Random Forest	0.75	1.00	1.00	0.33	0.75	1.00	0.75	0.60
K Neighbors	0.71	0.86	1.00	0.33	0.75	0.00	0.75	0.75

Table 4 Evaluation results of Supervised Classification

Interpretation: The performance of all the methods used is very similar.
All methods encounter challenges in modeling classe 3 (Cyberscience and Social Networks), except K neighbors method which encounter challenges in modeling the class 5 also (Robotics and embedded systems).
This issue may arise because our dataset is imbalanced.
Since the Decision Tree model achieved the highest accuracy and the best F1-score

values, indicating superior modeling of different classes, we decided to use this model to predict the entire dataset. After applying this model to assign classes to our 603 articles, the distribution of classes was as follows:

- Theoretical and Fundamental Computer Science: 249
- Information and Communication Technologies (ICT): 153
- Software Engineering and Systems: 70
- Artificial Intelligence and Machine Learning: 64
- Computer Vision and Image Processing: 38
- Cyberscience and Social Networks: 25
- Robotics and Embedded Systems: 4

5 Documents clustering

5.1 KMeans

After preprocessing and cleaning the data, we applied the TfidfVectorizer for feature extraction. Subsequently, we utilized the KMeans algorithm. The figure illustrates the relationship between the number of clusters (K) and the inertia. Notably, the plot 4 indicates that K=4 represents the optimal number of clusters.

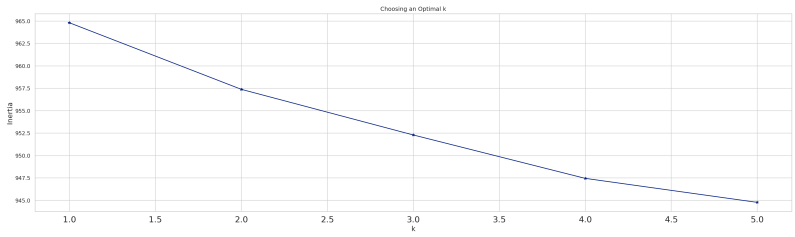

Fig. 4 Selecting the Optimal k

After applying KMeans with parameters set as max iter=200, clusters=4, and init=10, we employed PCA to reduce the dimensions to 2 after assigning the clusters. The result is a representation 6 of four topic clusters. However, it's evident that there is some degree of overlap between the clusters, likely due to the high similarity of topics within the tech field.

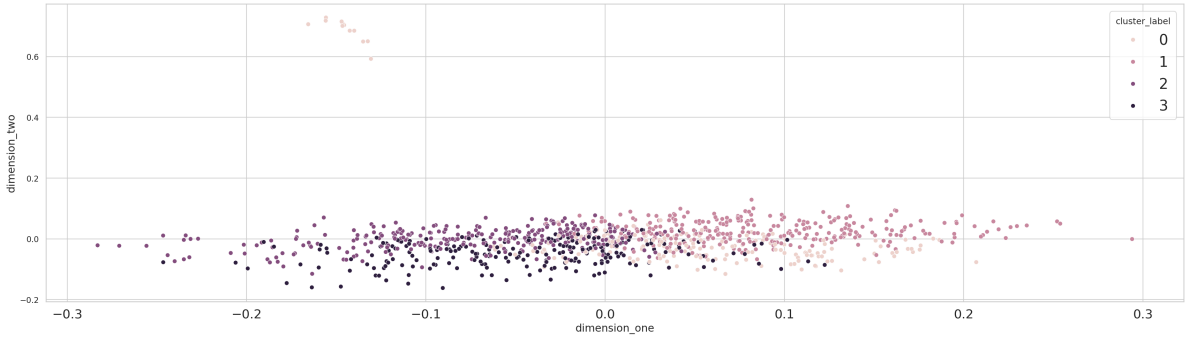

Fig. 5 Clusters representation

5.2 GCN

After assigning corresponding classes to our articles and grouping them together based on their degree of similarity, we also considered modeling the network of articles.
In this section, we have included the database containing the relationships between articles. Therefore, for each row of the dataset, we have the article and which article it has cited.
So, based on this relationship between the articles, we were able to construct the edges of our graph and considered that the nodes refer to the articles, specifically their titles. We trained a Graph Convolutional Network model using Adam as the optimizer and CrossEntropyLoss as the loss function. Afterwards, we opted for the graph visualization as depicted in the figure 7

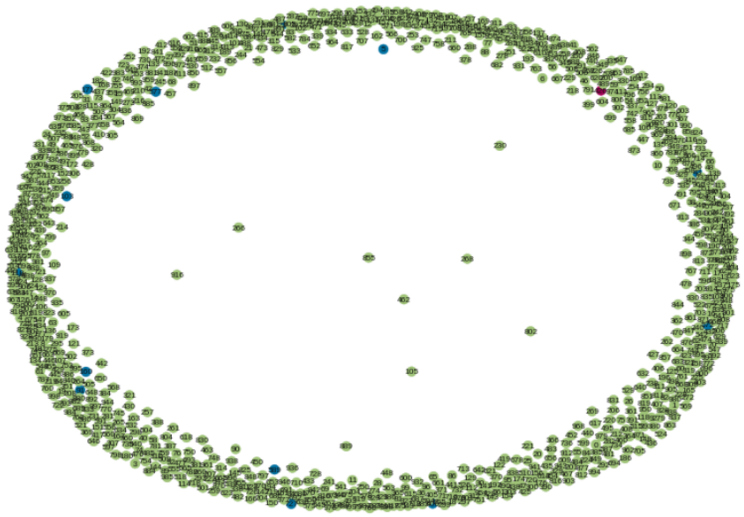

Fig. 6 Graph of GCN predictions

Interpretation: The GCN model did not adequately represent the relationships between the documents, especially considering that the model generated a loss of 0.6. Perhaps a more thorough data preparation process is required to achieve better results.

6 Challenges & Lessons Learned:

1. **T**he selection of classes and the manual assignment of classes to articles was a sensitive and laborious task, especially since the use of titles and content was prohibited. Therefore, relying solely on the venue for classification posed a real challenge.
2. **D**ealing with imbalanced data posed a significant challenge, making it difficult to achieve satisfactory performance for classes with low frequency.
3. **E**mphasizing the importance of confusion matrices, we found that assessing accuracy and F1 score alone was insufficient. Examining metrics for each class prompted further improvement of our approach.

7 Conclusion

In conclusion, our project has tackled the challenges of structured corpus analysis for information retrieval. By employing techniques like TF-IDF and Doc2Vec, we obtained valuable insights into document distribution and semantics. The development of a search engine enabled efficient retrieval of relevant documents based on user queries. Using clustering and classification is important for understanding corpus structure and content categorization. Moving forward, further refinements like using Graph-based networks can enhance scalability and performance, making our approach applicable in various research contexts.