Analysis of Information Networks Project

Mathis DOUSSE and Inès FAIVRE

Université Lumière Lyon 2

février 2024 - mars 2024

Abstract

L'objectif principal de ce projet est de développer une solution d'analyse d'un corpus structuré qui comporte plusieurs fonctionnalités telles que le chargement rapide des données et l'affichage de quelques statistiques, la visualisation du corpus pour donner une idée de la structure des données, l'inclusion d'un petit moteur de recherche permettant de faire des requêtes par mots clefs, une nouvelle structuration des données à l'aide de techniques de clustering et la classification supervisée des données en prenant en compte la structure et l'information textuelle.

Keywords : Network analysis, search engine, centrality measures, vectorization, embedding, clustering, classification.

1 Introduction

Dans cette étude, nous allons tenter de développer une solution d'analyse d'un corpus structuré en graphe qui comporte plusieurs fonctionnalités. Pour ce faire, nous devions dans un premier temps charger des données qui seront tirées du Citation Network Dataset et décrites dans la section 'Description du jeu de données'. Puis nous afficherons quelques statistiques concernant notre graphe d'articles et d'auteurs. Ensuite, nous pré-traiterons nos données afin de les rendre expoitables pour l'étude, que ce soit au niveau du contenu des articles qu'en termes de graphe. Enfin, nous nous attarderons sur les différentes fonctionnalités que nous pouvons mettre en place sur notre graphe. En allant de l'inclusion d'un petit moteur de recherche permettant de faire des requêtes par mots clefs à une nouvelle structuration des données à l'aide de techniques de clustering et à la classification supervisée des données en prenant en compte la structure et l'information textuelle.

C'est pourquoi, dans un premier temps, nous présenterons une description détaillée des données. Puis, nous mettrons en lumière la méthodologie adoptée pour tester les différentes fonctionnalités que nous avons implémenté afin de résoudre notre étude. Durant votre lecture, vous trouverez des résultats sous la forme de tableaux de graphes ou de textes tous provenant du code accessible au lien suivant : https://github.com/FeckNeck/Network-Analysis-for-Information-Retrieval

2 Description du jeu de données

Le jeu de données considéré est tiré du Citation Network Dataset qui rassemble plusieurs millions d'articles : https://www.aminer.org/citation. Nous utiliserons en particulier un extrait de la version 10 qui comprend plus de 3 millions d'articles et 25 millions de citations, extraits le 27 octobre 2017. Cet extrait, fourni sous forme de 4 fichiers d'export .json, comporte diverses articles publiés de 1990 à 2017 et ne conserve que quelques informations : identifiant, titre, résumé (s'il existe), auteur, lieu de publication, année, nombre de citations. Il est alors possible de faire un graphe pour les articles contenant toutes ses "métadonnées" et également sur les auteurs. Nous avons fait le choix de modéliser les deux pour la suite.

Nous aurions souhaité garder l'intégralité des données présentes dans les fichiers fournis mais celles-ci s'avèraient trop volumineuses pour notre machine et les différents besoins que nous avions. C'est pourquoi nous avons réduit notre jeu de données au fichier .json numéro 3 contenant tout de même 79

007 articles avec 178 268 auteurs différents. Nous avons désiré voir la proportion d'articles par rapport à leur année de parution:

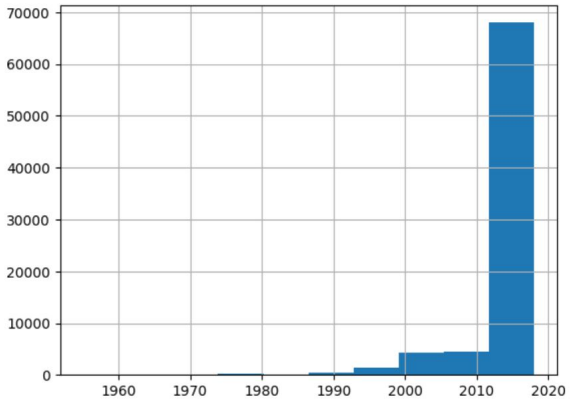

Figure 1: Nombre d'articles de notre jeu de données parus par année.

On voit que la grande majorité d'entre-eux ont été publié en 2017. Au sein des différents auteurs des articles, en moyenne chacun a publié 2 articles avec un record de 92 articles parus pour Wei Wang. Pour ce qui est des abstracts, leur longueur moyenne est de 1060 caractères avec une pointe maximum à 7139 caractères et minimum à 60. Cette information peut s'avérer utile dans la partie embedding des données puisque des méthodes telles que BertTokenizer ne permettent de tokenizer qu'un certain nombre de caractères par texte (512 tokens exactement).

3 Méthodologie

3.1 Pré-traitements

Dans cette partie, nous aborderons les pré-traitements faits sur les textes mais également sur nos graphes. Pour les textes, nous avons réalisé une fonction 'preprocess' contenant un ensemble de pré-traitements :

1. Ensemble du texte en minuscule
2. Regex pour retirer les chiffres et les caractères spéciaux
3. Mise en forme des mots sous forme de tokens
4. Retrait des 'stop words', les mots 'outils'
5. Lemmatisation des mots (lui renvoyer sa forme neutre canonique, son lemme)
6. Stemmatisation des mots (racinisiation, conserver que le radical des mots)

que nous appliquons à l'ensemble des abstracts que contiennent nos données. De plus, comme expliqué précédemment, nous avons réduit le nombre de données que nous avons pris en entrée. Ainsi, s'est présenté à nous un problème de graphe connecté pour les articles. En effet, ne prenant qu'un fichier sur l'ensemble de 4, il est possible que les articles de ce fichier ne soient pas forcément reliés (par le biais des citations d'auteurs) à d'autres. C'est pourquoi nous avons remplacé les références vides par des références existantes dans le corpus.

3.2 Prise en compte de la structure du corpus

Pour la prise en compte de la structure du corpus, comme dit précédemment, nous avons deux graphes, un portant sur les auteurs et un sur les articles écrits par ces auteurs. Le graphe des auteurs est relié par les auteurs qui ont collaboré ensemble tandis que le graphe des articles est relié par les auteurs

qui sont referencés dans l'article. Nous avons souhaité faire quelques statistiques sur nos graphes en prenant en compte par exemple le degré total de notre graphe ou bien calculer des mesures de centralité. Hors pour cela, il était nécessaire que notre graphe soit totalement connecté. Pour cela, nous avons récupéré dans nos graphes respectifs les plus grand sous-graphes connectés. Grâce à cela, nous avons pu faire nos mesures de centralité. On en a sélectionné quatre différentes :

- Degree centrality : basée sur le degré des noeuds
- Closeness centrality : basée sur la somme des chemins les plus courts vers tous les autres nœuds
- Katz centrality : basée sur l'influence d'un nœud au sein d'un réseau en mesurant le nombre de voisins immédiats ainsi que tous les autres nœuds du réseau qui se connectent à ce noeud. Les connexions établies avec des voisins éloignés sont pénalisées par un facteur d'atténuation
- Betweenness centrality : basée sur le nombre de fois qu'un nœud se trouve sur le chemin le plus court entre d'autres nœuds

Et voici les résultats obtenus sur ces deux graphes :

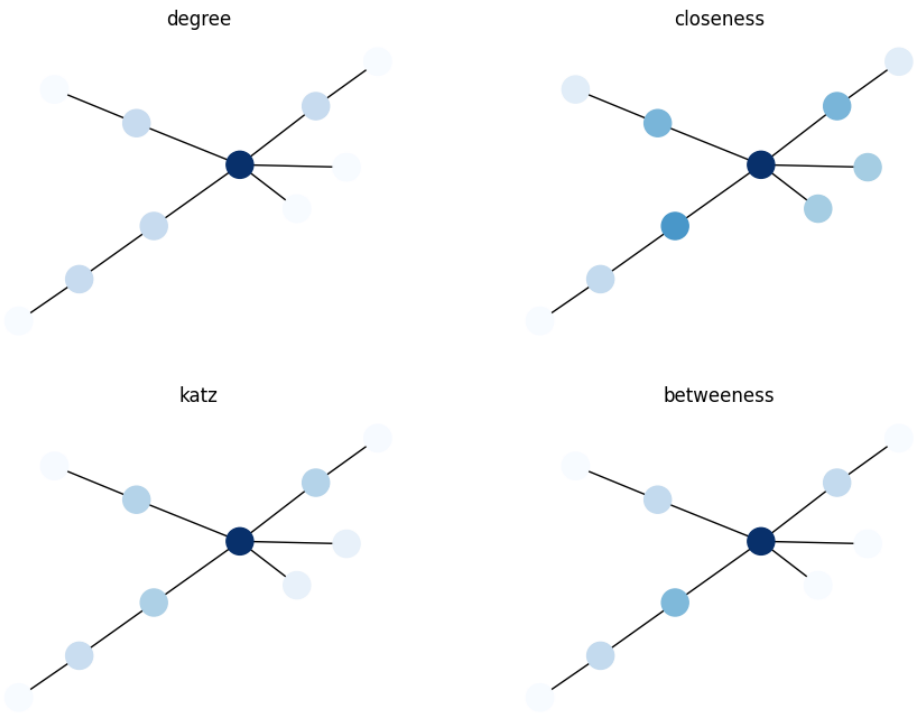

Figure 2: Différentes mesures de centralités sur le graphe des articles.

Le graphe des articles le plus connecté que nous avons réussi à avoir comporte 10 articles. Les mesures de centralité les plus hautes seront représentées par un bleu foncé tandis que les mesures faibles seront de ton bleu très clair voire gris pâle. Visuellement, on peut d'ores-et-déjà répérer un noeud central. Comme nous pouvions nous y attendre, sur l'ensemble des mesures, ce noeud a la mesure de centralité la plus grande. Pour la degree centrality, la majorité des noeuds sont de même couleur puisque le degré se base sur le nombre d'arcs (ou d'arêtes) reliant ce noeud. Ainsi chaque noeud est relié par deux arcs hormis le noeud central (5 arcs) et les noeuds aux extrémités (1 arc). Les autres mesures vont donner plus de poids aux noeuds proches du noeud central, à part les deux noeuds uniquement reliés au noeud central.

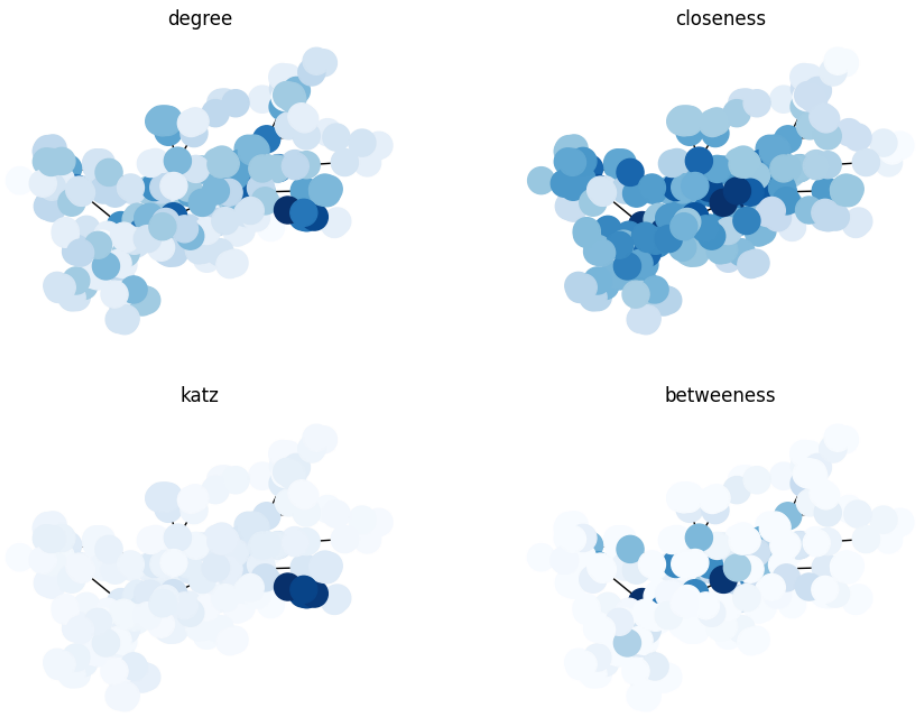

Figure 3: Différentes mesures de centralités sur le graphe des auteurs.

En ce qui concerne le graphe des auteurs, notre graphe entièrement connecté est beaucoup plus conséquent que celui des articles. On voit que pour la degree et la closeness centrality, nous avons des mesures de centralité très variées avec certains noeuds qui ressortent. Pour la Katz et la betweeness centrality, beaucoup de noeuds ont été pénalisés et ont des mesures de centralité très faibles. On distingue tout de même quelques noeuds avec des mesures fortes.

3.3 Moteur de recherche

Afin de mettre en place un petit moteur de recherche permettant de faire des requêtes par mots clefs, nous avons essayé plusieurs méthodes d'embedding pour resprésenter nos abstracts. Tout d'abord, nous avons expérimenté avec des méthodes simples telles que le TFxIDF. C'est une représentation qui en plus de prendre en compte le nombre d'occurrences (TF), prend en compte la rareté d'un mot dans le corpus. Cette représentation est dite creuse car elle se réfère à un vocabulaire fixe où la majorité des mots ne sont pas présents dans nos données. C'est pourquoi, pour contrer cela, d'autres méthodes dites pleines existes telles que les approches de plongement. On peut citer Doc2Vec qui est une extension des approches Word2Vec dans lesquelles on ajoute un "token" associé à chaque document (ici à l'abstract). Mais également Sentence-Bert ou la librairie Spacy. Nous avons essayé l'intégralité des méthodes cités ci-dessus et nous aurions aimé faire plus mais malheureusement, ayant fait notre projet en local, il n'était possible de faire fonctionner l'intégralité des méthodes d'embedding. Une fois les méthodes fixées, nous avons chercher à retourner les doccuments les plus similaires à la recherche demandée. Pour cela, nous avons utilisé des méthodes de similarité de type cosinus avec les différents embeddings. Enfin, pour une meilleure expérience utilisateur, nous avons proposé un petit système qui permet de sélectionner le mot que l'on recherche, le type d'embedding que l'on souhaite utiliser et le nombre de documents que l'on souhaite retourner.

3.4 Classification supervisée

Pour la partie classification, nous travaillerons sur le graphe des articles. Il a d'abord fallu créer des labels pour chacun de nos articles (nos noeuds). Pour ce faire, nous avons récupéré grâce à la variable 'venue' le lieu de publication de l'article. Cette variable nous permet ainsi d'avoir la conférence ou le

journal correspondant et ainsi avoir une brève idée du type de contenu de l'article. Par exemple, un article paru dans la conférence 'Conference on Computer Vision and Pattern Recognition' aura de très forte chance de parler de Computer Vision. Ainsi, nous avons récupéré les mots clefs des différents lieux de publication en retirant les mots en lien avec le lieu comme conference, international, research, etc. Puis nous avons créé les huit catégories suivantes :

- Artificial Itelligence : ["artifici","intellig", "theori", "electron"],
- Computer Vision : ["comput", "imag","ieee", "manag", "design", "vision", "recognit", "pattern"],
- Data : ["data", "analysi", "transact"],
- Human Computer Interaction : ["communic","robot", "scienc", "human", "sensor"],
- Machine Learning : ["model","mathemat", "language", "logic", "algorithm", "learn"],
- Networks : ["network", "system", "process", "secur","wireless","cybernet"],
- Software Engineering : ["engin", "technolog", "applic", "softwar", "program", "informat"]
- Other

Voici la répartition des différents articles par catégories que nous avons créé :

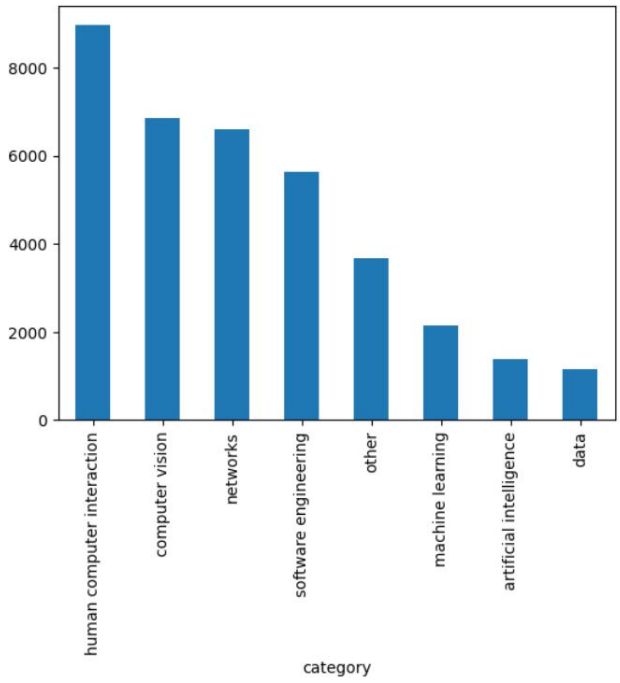

Figure 4: Nombre d'articles de notre jeu de données par catégories générées.

Nous souhaitions que la catégories 'Other' ne soit pas trop volumineuse par rapport au reste et ce n'est pas le cas ici. Une fois nos catégories créées et affectées à nos articles, nous avons choisi de les expérimenter avec dans un premier temps un embedding de type TFxIDF et trois modèles que sont : la régression logistique, les SVM et un perceptron muti-couches. Ensuite, nous avons souhaité pousser la classification en utilisant l'embedding de S-BERT mais cela s'est avéré très coûteux à faire tourner. Enfin, afin de ne pas seulement prendre en compte l'embedding des abstracts, nous avons souhaité intégrer la représentation de nos données qui est sous forme de graphes. Pour ce faire, nous nous sommes intéressés au GNN (Graph Neural Network) qui vont permettre d'utiliser des réseaux de neurones pour traiter des graphes. Dans notre cas, nous avons souhaité tester à la fois les GCN (Graph

Convolutionnal Network) qui vont permettre de prendre en compte l'importance des voisins des noeuds par rapport au nombre de couches établies dans le réseau. Mais pour aller plus loin, également les GAT (Graph Attention Transformers), non plus basés sur des racines d'approches spectrales mais sur un principe d'attention similaire aux transformeurs. Les GAT vont essayer de refléter une similarité entre noeuds grâce aux features.

3.5 Clustering

Enfin, dans la partie Clustering, l'idée était de tester un (ou plusieurs) algorithmes de clustering classiques, comme KMeans, le clustering spectral ou Louvain. Nons avons alors décidé d'utiliser les différents embeddings de la partie Classification (TFxIDF, Sentence-BERT et Doc2Vec) et d'essayer d'obtenir des groupes de thématique similaires à ceux émis par nos labels de classification. Pour évaluer nos résultats et nos performances, nous nous sommes servis de deux mesures de similarité entre clusters qui sont :

- L'ARI (Adjusted Rand Index Score)
- L'AMI (Adjusted Mutual Info Score)

L'ARI va considérer toutes les paires d'échantillons en comptant les paires attribuées dans le même cluster ou dans des clusters différents des clusterings prédits et réels. Il est compris entre 1 et -0,5 pour les clusters particulièrement discordants. L'AMI est un ajustement du score d'information mutuelle (IM). Il tient compte du fait que l'IM est généralement plus élevé pour deux clusters comportant un plus grand nombre de clusters, indépendamment du fait qu'il y ait réellement plus d'informations partagées. Ensuite, avec nos différents embeddings, nous sommes partis sur du KMeans et nous avons souhaité tester le Clusering Spectral ainsi que les blocs models (en utilisant AgglomerativeClustering de scikit-learn). Cependant, ces deux méthodes sont efficaces dans le cas où le graphe est entièrement connecté, ce qui n'est pas le cas ici. De plus, ce sont des méthodes très coûteuses que nous avons tenté de faire tourner en local, sur l'ensemble de nos données. Nous avons été obligé de réduire la quantité pour obtenir certains des résultats notamment avec le Spectral Clustering. Nous aurions aimé tester Louvain mais lors de l'installation de la librairie, une erreur a été retournée car la version de python que nous utilisons n'est pas assez récente. Cependant, il ne nous est pas possible de l'augmenter car torch ne fonctionne pas dans une version plus récente de python (3.12).

4 Résultats

Après avoir présenté les différents méthodes utilisées dans l'ensemble des parties ci-dessus, nous allons maintenant nous pencher sur les résultats. Pour la partie moteur de recherche, nous obtenons globalement des résultats satisfaisants mais regardons plus en détails :

```
Document  4173  :  On the feasibility of an embedded machine learning processor for intrusion detection  ( 72.0 %)
Document  44550  :  Infrastructure for Usable Machine Learning: The Stanford DAWN Project  ( 64.0 %)
Document  27692  :  Machine learning and systems for the next frontier in formal verification  ( 63.0 %)
Document  28972  :  The 2003 learning classifier systems bibliography  ( 63.0 %)
Document  27044  :  Hyperparameter optimization to improve bug prediction accuracy  ( 62.0 %)
Document  2985  :  Machine Learning Meets Databases  ( 61.0 %)
Document  34438  :  In vitro molecular machine learning algorithm via symmetric internal loops of DNA  ( 61.0 %)
Document  42712  :  The optimal crowd learning machine  ( 59.0 %)
Document  30622  :  Design and implementation of low-level machine learning API and API server  ( 56.99999999999999 %)
Document  12814  :  Ontology knowledge-based framework for machine learning concept  ( 56.99999999999999 %)
```

Figure 5: Résultats pour la recherche avec TFxIDF

En utilisant TFxIDF pour rechercher les documents les plus proches de la recherche 'machine learning', nous obtenons des taux de similarité entre 55 et 72%. En étudiant les titres, on peut voir que les documents semblent bien en lien avec les mots clés de la recherche.

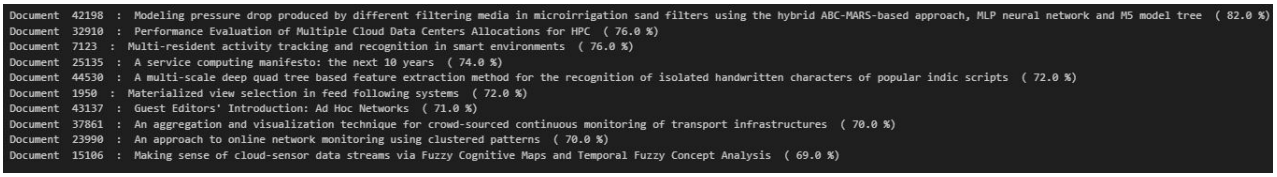

Figure 6: Résultats pour la recherche avec l'embedding de Doc2Vec

Cette fois, en se basant sur l'approche de Doc2Vec pour rechercher les articles similaires à la recherche du mot clé 'data', nous obtenons de meilleurs résultats qu'avec TFxIDF. L'article le plus proche obtient un taux de similarité de 82% contre 62% pour le moins similaire (sur une recherche de 10 articles).

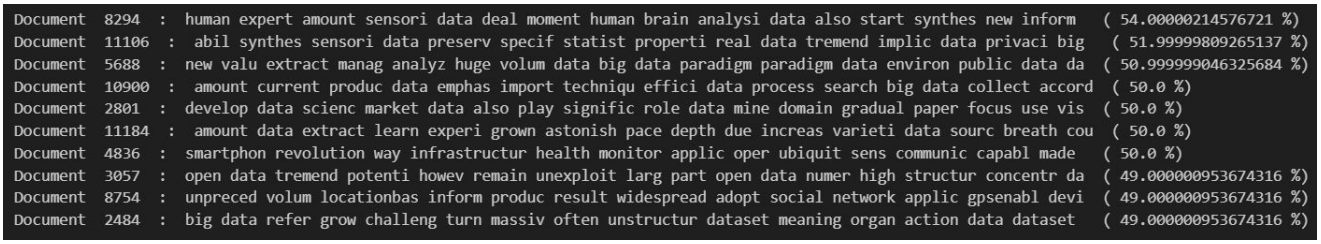

Figure 7: Résultats pour la recherche avec l'embedding de S-BERT

Enfin, avec l'embedding "Sentence-Transformers" de BERT pour la même recherche qu'avec doc2Vec (mot clé 'data'), nous obtenons des résultats assez mauvais par rapport au reste. Cela est sûrement dû au fait que nous utilisons qu'un quart des données pour vectorizer nos abstracts. L'action de le faire sur la totalité est malheureusement trop coûteuse, ce qui affecte les résultats.

Pour la partie sur la classification, nous avons proposé un récapitulatif sous forme de tableau avec les différentes méthodes utilisées, le type d'embedding et le score en accuracy :

Méthode	Embedding	Accuracy
Logistic Regression	TFxIDF	37%
SVM	TFxIDF	39%
MLP	TFxIDF	32%
GCN	-	**55%**
GAT	-	40%
GATv2	-	42%

Table 1: Résultats de la classification

On peut ainsi voir que les résultats que nous avons pu obtenir avec les diverses méthodes, sont vraiment très médiocres. On obtient de très légers meilleurs résultats avec les méthodes de type GNN. Nous n'avons pas augmenté le nombre de couches (seulement 2) car le graphe étant déjà très peu connecté, il ne nous semblait pas intéressant d'augmenter le nombre de voisins à prendre en compte. Ces résultats sont principalement dû au fait que le temps de calcul était trop volumineux dans certains cas. Mais peut-être également dû à nos choix pour classer nos articles dans les différentes thématiques ou à la prédominance de la classe 'Human Computer Interaction' par rapport aux autres.

Et cela se ressent également dans la partie Clustering. Le même problème de temps de calcul se présente également. Nous avons réussi à mettre en place KMeans ainsi que le Spectral Clustering

avec TFxIDF. Puis KMeans avec Doc2Vec et l'embedding d'un quart de nos abstracts avec S-BERT et le Spectral Clustering. Nous avons fixé k à 8 car c'est le nombre de catégories que nous avons créées précédemment. Malgré toutes ces méthodes nous n'avons pas parvenu à avoir des résultats satisfaisants. Nous avons remarqué que la majorité des clusters sont assignés au même thématique (voir 8), ce sont les classes majoritaires et probablement celles qui ont le plus d'articles reliés entre eux.

```
Cluster  0  :  human computer interaction
Cluster  1  :  computer vision
Cluster  2  :  networks
Cluster  3  :  human computer interaction
Cluster  4  :  networks
Cluster  5  :  computer vision
Cluster  6  :  human computer interaction
Cluster  7  :  human computer interaction
```

Figure 8: Catégorie la plus représentée dans chaque cluster

5 Conclusion

Pour conclure, nous sommes satisfaits du travail que nous avons réalisé sur ce projet. Il nous a permis de développer une solution d'analyse d'un corpus structuré en mettant un point important sur les graphes d'informations et en particulier sur les GNN qui étaient la grande nouveauté.

Néanmoins, nous avons rencontrés plusieurs difficultés dans la réalisation du projet. Premièrement, le jeu de données était à l'origine décomposé en 4 fichiers très volumineux ($\sim$ 1giga). Charger la globalité du jeu de données rendaient nos calculs beaucoup trop longs, nous avons alors décidé d'utiliser le plus petit fichier des 4. Cependant, les références des documents n'étaient pas forcément inclusent dans ce même fichier, nous avons donc décidé de remplacer les réferences par des existantes dans les documents, comme expliqué précédemment. Le choix des références s'est fait aléatoirement ce qui a engendré un graphe qui est très peu connecté. L'ideal aurait été de pouvoir charger l'intégralité des données pour avoir un graphe plus représentatif. De plus, certaines opérations que nous avons réalisées telles que la vectorisation de données avec BERT ou encore la classification avec les SVM étaient très longues ($\sim$ 15min) ce qui ralentissait le développement. Enfin, nous aurions aimé réaliser une interface graphique afin d'avoir un projet plus interactif, notamment pour le moteur de recherche ou pour les graphes. Par manque de temps et pour les raisons évoqués précédemment, nous avons abandonné cette idée. En ce qui concerne les évolutions possibles, il aurait été préférable de sauvegarder les données vectorisées (notamment celles de BERT) dans un fichier .csv ou une base de données vectorielle. Cela nous aurait permis de gagner du temps dans le développement du projet et aurait rendu la réalisation de l'interface graphique plus enviseagable. Il aurait été alors possible de simplement charger nos documents vectorisés plutot que de re-calculer leurs embeddings à chaque exécution. Dans la meme idée, nous aurions aussi pu utiliser LangChain et LangServe en ce qui concerne le moteur de recherche.