Classification des publications de recherche

Rapport de synthèse

Projet n°48

Antoine Delporte
Thibaut Mallet
David Thery-Bulha

Table des matières

I- Présentation du projet

1) Problématique

Chaque année, des dizaines d'articles issus de la recherche scientifique à Télécom SudParis sont publiés. Les thèmes abordés sont divers et évoluent avec les tendances d'années en années.

Le but de ce projet de développement est de trouver un algorithme de fouille de texte qui permette, à partir des données disponibles sur ces publications de faire ressortir les grands thèmes abordés. Les différentes publications seront ainsi rassemblées en clusters voire sous-clusters représentant ces domaines et sous-domaines de la recherche dans l'école.

Les encadrants sont Bruno DEFUDE, Directeur adjoint de la Recherche et Bernadette DORIZZI, Directrice de la recherche de Télécom SudParis.

Le but final est d'arriver à visualiser ce découpage au travers de graphiques interactifs permettant de mieux communiquer autour des thématiques de recherche dans l'école sans avoir à faire le travail manuellement.

L'algorithme ne doit donc pas être dépendant d'un corpus de publications mais doit pouvoir s'adapter aux évolutions afin que la création de cette visualisation puisse se faire et se mettre à jour facilement.

Enfin, notre projet s'inscrit dans une démarche de recherche ; c'est-à-dire que notre objectif est d'explorer les possibilités qu'offrent les différents algorithmes de classification pour améliorer la visibilité des publications de Télécom SudParis. Notre but n'est donc pas d'écrire un programme complet, facile d'utilisation, avec une interface graphique, etc.

2) Solutions employées

Afin de créer ces groupes de documents nous nous baserons sur des corpus de publications issues des enseignants-chercheurs de Télécom SudParis. Ceux-ci contiennent à minima le titre de la publication et les auteurs. Il est ainsi possible de remonter aux groupes de recherche correspondant. Pour d'autres documents nous pouvons disposer d'une liste de mots-clés. Toute ces informations nous permettrons de créer des ensembles de publications grâce à des algorithmes de classification déjà existants. Nous utiliserons, dans le cadre de ce projet, deux de ces algorithmes : algorithme de K-moyennes et algorithme de classification ascendante hiérarchique.

Dans un premier temps il faudra donc mettre en place ces algorithmes, définir leur pertinence et évaluer le poids des différents arguments afin d'obtenir un résultat parlant et réaliste.

Il faudra ensuite le mettre en place de manière générique en faisant le lien avec une méthode de visualisation des données qu'il faudra définir.

II- Aspects techniques

1) Source des données

Nous avons travaillé sur deux bases de données différentes. La première contenait une liste de 4449 publications avec pour chacune le titre et les auteurs comme données intéressantes. Toute ces publications n'étaient pas pertinentes. Nous avons choisi de travailler uniquement sur celles qui correspondaient à un article paru dans une revue ou à une conférence. En enlevant les éventuels doublons nous avons 1344 publications. La base de données contient en outre des informations permettant de relier un auteur à une équipe de recherche ce qui nous sera utile pour la suite.

La seconde base de données est plus modeste : 191 publications seulement mais la plupart est accompagnée d'une information supplémentaire : des mots clés ajoutés par les auteurs eux-mêmes.

Plusieurs problèmes se posent déjà quant à ces données :

- Elles proviennent de deux sources différentes et ont donc un format différent pour lequel il a fallu s'adapter.
- Elles sont parfois incomplètes (pas d'auteur, pas de mot clés…)
- Le format d'écriture d'un nom est parfois différent

{\bf{Bardel, No{\'e}mie}} and {\bf{Desbouvries, Fran{\c{c}}ois}}

Texte tel que présent dans la base de données

Une fois ces problèmes résolus il a été possible dans la majorité des cas d'associer une publication avec un ou plusieurs enseignants-chercheurs de l'école et par extension à une ou plusieurs équipes de recherche.

2) Prétraitement

Nous avons décidé de considérer une publication comme une liste de mots intéressants à partir desquels on peut faire des rapprochements avec d'autres publications. Ainsi le titre et les mots-clés (si présents) ont été considérés de la même manière. Nous reviendrons par la suite sur l'apport de ces mots-clés.

La première étape du traitement fut de ne garder que les publications en anglais. Cela représente la majorité de nos données et travailler avec plusieurs langues aurait été impossible. Pour cela nous avons utilisé une bibliothèque de détection de langage crée par Shuyo Nakatani (https://github.com/shuyo/language-detection/blob/wiki/ProjectHome.md).

Ensuite l'étape principale consiste à lemmatiser les mots. C'est-à-dire à ne garder que la racine d'un mot (enlever les conjugaisons d'un verbe, les marques de pluriel etc..). Il existe de nombreuses bibliothèques qui font cela. Après en avoir testé plusieurs, notre choix s'est porté sur Stanford Core NLP (http://stanfordnlp.github.io/CoreNLP/), une suite d'outils de traitement du langage naturel crée par l'université de Stanford, pour sa facilité d'intégration et son efficacité. Elle permet aussi une grande liberté de personnalisation du traitement que nous n'avons pas utilisé.

La dernière étape consiste enfin à filtrer les mots indésirables, vides de sens. Il peut s'agir des *stopwords* communs dans une langue (*a, the, about, begin, did* …), de caractères indésirables (ponctuation) ou encore de mots courants dans notre contexte (*system*, *environment*, *model*…).

Une fois cette dernière étape effectuée nous pouvons associer une à une les publications avec leurs auteurs (en ne gardant que les auteurs de l'école) et leurs équipes de recherche.

Nous avons décidé de stocker ces résultats dans une nouvelle base de données contenant les titres prétraités et les références vers les autres bases afin d'éviter de refaire ce travail à chaque fois.

3) Algorithmique

Nous allons maintenant décrire le fonctionnement des algorithmes de classification pour comprendre comment nous sommes passés d'une liste de titres issus du prétraitement à une visualisation des classes de manière interactive.

Il existe de nombreux algorithmes de ce type mais notre choix s'est porté sur les deux suivants : l'algorithme de K-moyennes et l'algorithme de classification ascendante hiérarchique. Ils ont tous les deux des avantages et des inconvénients qui seront décrits précisément dans les parties correspondantes.

a) Algorithme de K-moyennes

Cet algorithme permet de diviser un ensemble de données en K partitions avec K défini manuellement. Le principe est simple, il s'agit de choisir K objets dans notre ensemble (aléatoirement ou non) formant ainsi K clusters (classes), ensuite on (ré)affecte chaque objet de notre ensemble au cluster Ci de centre Mi tel que la distance entre cet objet et Mi soit minimale, puis on recalcule le centre Mi du cluster (barycentre) et on recommence à partir de la deuxième étape tant qu'on fait des affectations. On voit que l'algorithme s'exécute simplement tant qu'on arrive à définir mathématiquement notre espace d'objets. En effet, étant donné que nous travaillons sur du texte (des titres), il est plus difficile de calculer des distances entre objets, il a donc fallu construire un espace mathématique à partir des données prises en entrée (la liste des titres). Pour cela, nous avons commencé par créer un dictionnaire de tous les mots utilisés dans l'ensemble des titres pour créer une base vectorielle ; ainsi, nous avons ensuite pu récupérer les coordonnées de chaque titre dans cette base vectorielle. Pour illustrer le principe employé, voici un petit exemple :

- Admettons que notre dictionnaire de mots est le suivant : [bonjour, cassiopée, chaud, fait, il, ordinateur, projet, réseau, temps, train, voiture]
- On souhaite vectoriser le titre « Bonjour, il fait chaud »
- On obtient alors le vecteur [1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0].

Maintenant que l'on a des vecteurs numériques, on peut utiliser les outils mathématiques et donc calculer la distance entre deux vecteurs. Nous avons essayé avec deux fonctions de distance : la distance euclidienne et la distance cosinus. La distance euclidienne est l'une des plus simple, elle s'écrit de la manière suivante : $d(X,Y) = \sqrt{(x_i - y_i)^2}\ avec\ x_i\ et\ y_i\ coordonnées\ de\ X\ et\ Y$. Et la distance cosinus s'écrit $d(X,Y) = \left|\frac{X.Y}{\|X\|*\|Y\|} - 1\right|$. Nous avons décidé d'utiliser la distance cosinus car elle permet de mieux discriminer nos vecteurs et donc d'avoir un meilleur clustering ; cette distance est très utilisée dans le cas de comparaisons de textes.

Concernant la programmation de cet algorithme, nous avons commencé par l'implémenter nous-même mais nous n'arrivions pas à gérer la condition d'arrêt de la boucle ; nous avons donc cherché et trouvé un code JAVA implémentant déjà cet algorithme (code d'Orhan Demirel).

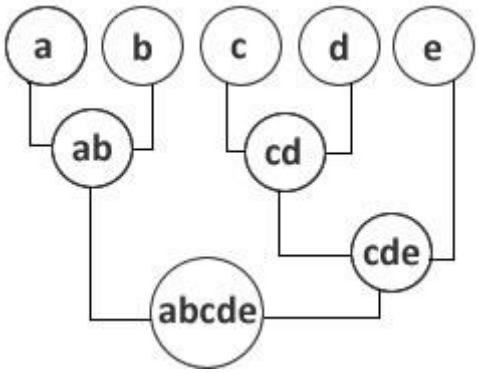
b) Algorithme de classification ascendante hiérarchique

Cet algorithme consiste à regrouper deux à deux les objets les plus proches, il faut donc définir à l'avance une matrice de distances entre chaque objet, la distance cosinus dans notre cas.

L'algorithme fonctionne de la manière suivante :

- Initialisation :
  - Les clusters initiaux sont les n singletons correspondants aux n titres.
  - On calcule la matrice des distances des objets deux à deux.
- Boucle :
  - On regroupe les deux éléments (objets ou groupe d'objets) les plus proches.
  - On met à jour la matrice des distances en remplaçant les deux éléments regroupés par le nouveau et en recalculant sa distance avec les autres clusters.

On obtient à la suite de cet algorithme un dendrogramme, voici à quoi un dendrogramme ressemble :



Pour programmer cette classification, nous avons utilisé une bibliothèque JAVA (https://github.com/lbehnke/hierarchical-clustering-java) pour gagner du temps, ainsi que la même méthode expliquée dans la partie précédente pour vectoriser nos données et calculer la matrice de distances.

c) Comparaison des deux algorithmes

	K-moyennes	**Classification ascendante hiérarchique**
Avantages	- Les objets peuvent être réaffectés dans une autre classe dans une itération. - Efficace et converge rapidement.	- Le nombre de classes est déterminé automatiquement. - Possibilité de choisir une fonction distance

Inconvénients	- Le nombre de classes est choisi manuellement, donc pas optimal.	- Converge plus lentement que k-moyennes.

4) Visualisation

Une autre grande partie de notre travail fut la visualisation. Ce n'est pas la plus compliquée mais c'est celle qui a le plus d'impact pour quelqu'un d'extérieur. Nous avons donc décidé d'utiliser principalement des bibliothèques JavaScript afin d'obtenir rapidement des visuels agréables et informatifs.

Dans une première approche, nous avons voulu utiliser des nuages de mots. Cette solution est très simple à mettre en place et fournit un résultat très parlant. Nous avons d'abord utilisé cette technique pour avoir un aperçu avant *clusterisation* de l'ensemble des mots-clés générés par le prétraitement ou par équipe de recherche. Puis nous l'avons utilisé après *clusterisation* afin d'avoir rapidement un aperçu du résultat et de pouvoir comparer notre méthode en changeant certains paramètres de manière rapide. Pour cela nous avons utilisé la bibliothèque Wordcloud2.js (https://github.com/timdream/wordcloud2.js) dont nous avons personnalisé l'interface afin de pouvoir jouer dynamiquement avec la visualisation.

Voici un exemple de nuage de mot représentant un cluster

Pour notre visualisation principale nous avons décidé d'utiliser la bibliothèque D3.js (https://d3js.org/). Celle-ci fournit des visuels de base très puissants, personnalisables à l'envie (avec des scripts JavaScript ou PHP). La visualisation la plus intéressante est celle présentant les clusters sous forme de cercles plus ou moins grands en fonction du nombre de titres et sur lesquels on peut zoomer afin de voir les sous-clusters. Nous avons rajouté la possibilité d'afficher de manière dynamique des informations détaillées sur chaque cluster avec des statistiques et une liste des publications originales (titre, auteurs, équipe de recherche) constituant le cluster.

Lorsque les clusters sont calculés, il ne reste qu'à générer le visuel. Pour cela, notre programme génère un fichier JSON qui contient toutes les données (identifiants des titres, taille du cluster, etc.). Ce fichier JSON est donné à un script Javascript qui s'occupe de créer le visuel interactif.

Afin de coller un peu mieux avec notre second type de *clustering* (hiérarchique), nous avons utilisé un deuxième type de visualisation permettant de mieux cerner les résultats de cette classification. Malheureusement nous n'avons pas trouvé le moyen de récupérer facilement et automatiquement les données après ce *clustering* hiérarchique et donc de les inclure dans la visualisation. Cela nous aurait demandé plus de temps.

5) Tests

Il est à noter qu'une grande partie de notre travail a consisté à faire des tests et comparaisons afin d'arriver au meilleur résultat possible.

Tout d'abord il a fallu choisir les bibliothèques avec soin comme nous l'avons vu précédemment. Ensuite il a fallu tester l'impact des nombreux paramètres sur les résultats obtenus après *clusterisation.* Par exemple l'inclusion des mots clés a-t-elle un impact au final ? Une des grandes difficultés est justement de caractériser « un bon résultat » car cela est très subjectif et porte très souvent sur le sens des mots. Dans notre exemple nous avons constaté que l'inclusion des mots clés rendait les clusters plus cohérents et que les publications de ceux-ci portaient plus souvent sur des sujets proches. Toutefois la différence n'était pas si significative, cela est peut-être dû à la faible quantité de l'échantillon de publications avec des mots clés. Nous avons néanmoins décidé de les garder afin d'utiliser les mots supplémentaires qu'ils peuvent apporter.

III- Management et Risques lors du déroulement du projet

1) Management et rôles des membres d'équipe

Tout d'abord, de manière générale, notre projet s'est déroulé sans trop de problèmes de gestion. Cela est notamment dû à notre gestion régulière des objectifs en comparaison avec un planning initial et les conseils de nos encadrants, ainsi que notre mise à jour des fonctions et algorithmes à utiliser pour optimiser nos résultats.

Dès le début du projet, nous avions pris l'habitude de réaliser des comptes rendus de chaque réunion avec nos encadrants résumant les points principaux à développer, les algorithmes et fonctions à étudier et/ou tester pour les semaines à venir, et enfin les techniques de visualisations des clusters ainsi que leurs conseils pour obtenir des visualisations cohérentes des résultats afin de faire apparaître le travail restant et les modifications à apporter au travail déjà effectué.

En ce qui concerne les responsabilités au sein de l'équipe, nous n'avions pas véritablement de rôle fixé au sein de ce projet : Nous nous répartissions les tâches de manière assez variée pour ce qui est de la recherche des solutions à aborder. Si au sein du groupe David proposait de manière plus spontanée des répartitions du travail au sein du groupe et les solutions à implémenter en java ainsi que les solutions de planification après chaque réunion (en générale hebdomadaire), nous avions tout de même chacun tendance à proposer des arguments de planification. Egalement de manière générale, Antoine se chargeait de réaliser des recherches sur les algorithmes à utiliser et proposait des alternatives aux solutions déjà mises en place et Thibaut avait d'avantage tendance à résoudre les problèmes liés au développement des fonctions et algorithmes rencontrés durant tout le projet.

2) Planification

Pour ce qui est de la planification et des responsabilités, notre projet s'est déroulé de la manière suivante :

a) Réunions

Nous avions une série de 13 réunions (en générale hebdomadaires) entre le 14 janvier et le 28 mai et nous avons commencé la définition du projet début février. Pour ce qui est des différents livrables (poster, revue 1 et 2, planning et synthèse) nous prévoyions généralement une semaine avant le deadline pour la répartition des tâches et la rédaction.

b) Préparation

Il nous a fallu environ un mois pour réaliser correctement une liste de *stopwords* (même si lors de certaines réunions en Avril et Mai nos encadrants nous ont tout de même conseillé de la modifier afin d'obtenir un *clustering* plus pertinent, en effet nos nuages de mots présentaient des défauts de visualisation et des mots récurrents). Nous avons également rencontré des défauts de lemmatisation des mots (i.e. regroupement des synonymes lors de l'application des algorithmes k-moyenne et hiérarchique) après la période de préparation, ce qui gênait la cohérence dans la visualisation.

Cette partie de préparation a été en grande partie réalisée par David.

c) Clustering

Entre début février et mi-mars, Nous avons travaillé sur le *clustering* de k-moyenne (Thibaut s'est chargé de toute l'implémentation et nous avons chacun étudié les avantages des différentes distances utilisables pour cet algorithme afin de réaliser un *clustering* bien cohérent).

Entre mi-mars et fin avril, Antoine a travaillé sur l'autre algorithme de *clustering* disponible (le hiérarchique) afin d'essayer de faire une comparaison de visualisation avec k-moyenne.

En ce qui concerne la visualisation, Antoine et Thibaut se sont chargés des nuages de mots (tests et préparation d'un fichier Javascript) et David et Thibaut se sont occupés de la génération d'un fichier JSON de visualisation des clusters.

d) Finalisation

La finalisation nous aura pris environ un mois, entre le transfert des données prétraitées dans une nouvelle base de données, les comparaisons et améliorations de visualisation, les statistiques de clusters, et la capitalisation sur le résultat.

3) Analyse a posteriori du déroulement du projet

Ce qui s'est bien passé : Tout d'abord, nos échanges avec les encadrants ont toujours été très constructifs et nous ont amené à nous poser les bonnes questions tout au long du projet, que ce soit sur les techniques de *clustering* ou sur la visualisation de nos résultats.

Ensuite, notre organisation était en général très bonne et la répartition des tâches a toujours été sans mésententes au sein du groupe.

Ce qui s'est mal passé : Nous avons eu quelques difficultés avec le prétraitement de texte et surtout sur la gestion de la liste des *stopwords*, ce qui avait des conséquences sur la visualisation des clusters et également des nuages de mots. Par ailleurs pour le choix des distances lors de l'implémentation de l'algorithme de *clustering* de k-moyenne, nous avons hésité 2 à 3 semaines sur la meilleure distance à utiliser pour obtenir des clusters cohérents. Enfin, en ce qui concerne nos objectifs, nous avons eu du mal à terminer toute la partie de la visualisation puisqu'il nous manque à l'heure actuelle la traduction du dendrogramme de *clustering* hiérarchique en un arbre pour la comparaison avec k-moyenne.

4) Plan de management des risques

Notre But principal n'est pas de faire un programme fonctionnel clé en main mais d'explorer les possibilités offertes par le *machine learning* pour améliorer la visibilité des publications

a) Objectifs de la conduite du projet

Notre conduite de projet a deux objectifs principaux : Le respect de la qualité (performances, spécifications) et le respect des délais fixés par notre planning (le respect des coûts n'étant pas pris en compte ici). Par cette étude, nous cherchons à déterminer quand et comment chaque risque est susceptible de se matérialiser, puis il nous faut qualifier sa gravité, sa probabilité, le type d'impact, la détectabilité puis la criticité du risque (probabilité*gravité).

b) Identification des risques projet

Le principal risque projet pour nous était bien sûr le respect des délais fixés par le planning (**RISQUE 1**) : Nous devions veiller à exécuter chaque étape suivant les prévisions tout en prenant en compte les indications et conseils de nos encadrants afin d'obtenir une bonne qualité dans les résultats. Nous devions parfois négocier des marges supplémentaires auprès de nos encadrants pour finir le travail correctement (en effet, dès le début nous avons mis du temps pour apprendre le

fonctionnement des algorithmes de classification). Par ailleurs, ce respect des délais passait nécessairement par une clarification permanente des rôles et des responsabilités (**RISQUE 2**) sur les tâches à accomplir.

Il était nécessaire de porter une attention particulière lors de la récupération des titres depuis les bases de données (**RISQUE 3**), la création d'un dictionnaire exhaustif de tous les mots utilisés dans le lot de données (**RISQUE 4**) et l'analyse des résultats obtenus après application des algorithmes (**RISQUE 5**).

Lors de la recherche des méthodes et du prétraitement de texte, il est assez difficile de pouvoir établir une liste exhaustive de *stopwords* (**RISQUE 6**). En effet cela est primordial afin d'optimiser l'affichage des clusters. Toujours pour l'affichage, il nous fallait vérifier que la lemmatisation des mots s'exécute correctement (**RISQUE 7**) (cf « data » et « datum »).

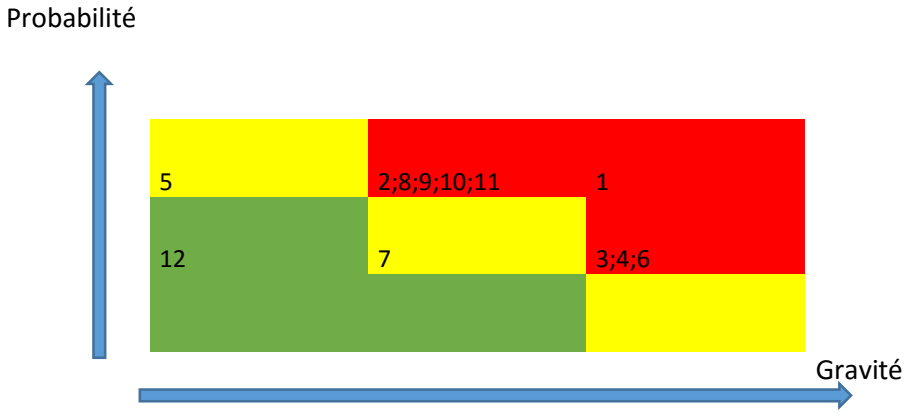
Puis, lors des deux approches *clustering* nous devions choisir correctement une API afin d'optimiser les résultats (**RISQUE 8**). Le choix du programme à utiliser pour les nuages de mots constituait le même risque d'optimisation des résultats (**RISQUE 9**). De même, la décision quant au meilleur des deux algorithmes (k-moyenne et hiérarchique) doit se faire sur plusieurs critères (**RISQUE 10**) (notamment l'obtention de clusters bien convexes et cohérents, et également correctement séparés les uns des autres). Pour cela, nous avons d'ailleurs longtemps hésité quant à la distance à utiliser (**RISQUE 11**) (cosinus ou euclidienne) pour finalement opter pour la distance cosinus.

Par ailleurs, il y a également l'aspect aléatoire des résultats des algorithmes qu'il faut prendre en compte (**RISQUE 12**), et cela nous nous en sommes aperçu quelques mois après le début du projet. Cela ajoute une part significative de travail pour la fiabilité que nous n'avions pas pris en compte à la base dans nos délais.

Finalement, à chaque nouvelle phase du projet, nous avons pu observer que de nouveaux risques peuvent apparaître tout au long de la vie du projet : il était donc nécessaire d'établir une surveillance des risques ou plan de management des risques.

c) Evaluation et hiérarchisation des risques projet

Le meilleur outil d'évaluation est la grille de criticité : ci-dessous une grille avec dans chaque case le ou les numéros de risques correspondants.



On voit clairement que le point le plus critique de notre projet est le respect des délais : en effet notre travail évolue en permanence à chaque phase du projet et donc les marges varient en permanence ce qui implique le besoin de surveiller les risques.

Un dernier point qui nous aura permis de gagner beaucoup de temps durant l'implémentation des algorithmes est le choix du langage de programmation (ici java) que nous connaissions déjà et dont l'utilisation nous aura posé peu de problèmes techniques. De plus, nous avons trouvé certaines fonctions assez connues (comme les algorithme *complete linkage* ou *single linkage*) déjà implémentées en Java sur *github.com* et/ou dont les modifications à effectuer étaient peu importantes.

IV- Conclusion

Nous avons donc atteint notre objectif au cours de ce projet qui était de parvenir à une visualisation des publications, leur donner du sens et permettre une éventuelle exploitation. Toutefois ce résultat n'est arrivé qu'à la fin et il a donc été difficile tout au long du projet de pouvoir exploiter véritablement nos résultats, les comparer et leur donner un sens.

Entre différents types d'algorithmes par exemple il est possible de comparer le nombre de clusters, le nombre d'auteurs et d'équipes différents pour chacun mais il est difficile de corréler ces nombres à un bon résultat ou pas.

Un autre problème est la perte de sens des mots. En effet, sorti de son contexte un mot peut avoir plusieurs sens ou plusieurs mots peuvent désigner le même concept et il est alors très difficile pour un algorithme de saisir toute les nuances et subtilités d'une langue même dans un contexte qui est le nôtre, assez différent du langage naturel. De plus nous sommes obligés d'utiliser des informations *a posteriori* sur les résultats (enlever un mot- clé trop courant afin d'avoir une meilleur séparation).

Cela montre que ce domaine de travail pose de nombreuses questions intéressantes et problématiques que nous avons pu seulement abordé au cours de ce projet et où de nombreux travaux peuvent encore être fait afin d'améliorer ce qui est possible de faire avec du *machine learning* et des données textuelles.

ANNEXE : Documentation technique (simple)

Note préliminaire :

Le programme conçu en Java n'est pas fait pour être utilisé comme tel, il représente un brouillon de nos travaux et montre ce qu'il est possible de faire. De nombreux points ont été retravaillés au cours du projet et sont parfois inutiles. Pour plus d'informations, se référer aux commentaires du programme.

Base de données :

Le programme est fait pour utiliser un certain type de base de données dont les champs sont calqués sur le format *BibTex* (format générique pour les bases bibliographiques). En cas d'utilisation d'une autre base de données, le programme est à adapter.

Dépendances :

En plus des bases de données mentionnées le programme requiert deux bibliothèques :

- Stanford Core NLP : http://stanfordnlp.github.io/CoreNLP/
- Language Detection : https://github.com/shuyo/language-detection

Programme Java :

Le programme est divisé en 3 Projets : *Text*, *Clustering* et *Visualisation* s'occupant chacun de la partie correspondante.

Text permet de récupérer les informations d'une des bases de données et de les transférer sous forme d'une classe Vecteurs contenant pour chaque publication un numéro d'identification (celui de la base de données), une liste de mots prétraités, une liste d'auteur et une liste d'équipes de recherche.

Elle permet en outre la sauvegarde de ces données sous forme texte (à modifier selon ce qu'on veut sauvegarder, l'implémentation reste basique) ou la création d'une base de données intermédiaire contenant ces mots prétraités.

L'utilisation basique pour construire l'ensemble des mots et les récupérer est (celle-ci peut différer en fonction des options et de la base de données utilisée) :

```
DB myDB = new DB("nomDeLaBase");
Vecteurs v = new Vecteurs(myDB);
List<List<String>> titles = v.getVectors();
myDB.close();
```

Une classe Publi permet d'utiliser la base de données intermédiaire mais celle-ci reste basique et certaines fonctions ne sont pas (ou pas bien) implémentées. Des exemples sont fournis dans le code.

```
Publi p = new Publi();
List<String> titles = p.getWords(1); // 1,2 ou 3
en fonction de la base de donnée à utiliser
```

(Vecteur retourne pour une publication les mots sous forme de List<String> alors que Publi retourne sous forme d'une seule String ou les mots sont séparés par ' ;')

Clustering permet de calculer les clusters à l'aide des différents algorithmes de classification précédemment présentés.

On commence d'abord par créer notre objet qui va permettre de faire toutes les opérations sur les clusters :

```
Clusters clusters = new Clusters(25, "FichierTexteContenantLesTitres", true,
new File("RépertoireOuSontEnregistrésLesClusters"));
```

On génère le dictionnaire de tous les mots employés dans les titres, puis on vectorise les données :

```
clusters.createDictionary();
clusters.vectorizeData();
```

On crée les clusters en choisissant la fonction de distance que l'on désire utiliser puis on les sauvegarde et on crée des sous-clusters (facultatif) :

```
clusters.create(ChoixDeLaDistanceUtilisée);
clusters.save();
Clusters.subClusterize("data/clusters/", 20, 1);
```

Enfin, on génère le visuel en créer un fichier JSON :

```
Clusters.generateVisual("RépertoireDEntrée", "RépertoireDeSortie", 1, v);
```

Visualisation a pour tâche de générer une page html permettant de visualiser les résultats après clusterisation sous forme de nuage de mots. Le reste des visualisations, pour gagner du temps a été implémenté dans la projet *Clustering.*

L'utilisation se fait via la classe Wordlcloud après avoir généré les clusters :

```
Wordcloud w = new Wordcloud(clusters.getResultClusters(),
"RepertoireOuOnVeutLaPageHTML");
```