TAL
Classification et prediction d'offres d'emploi

REGUIG Ghiles

April 27, 2017

Abstract

Dans le cadre de l'UE de Traitement Automatique du langage, on s'intéresse à la problèmatique de classification et prédiction de documents, en se basant sur l'aspect sémantique de ces dernier.s On se place dans le cadre de la recherche d'emploi et l'on désire prédire, selon un document décrivant l'emploi recherché (un CV ou une lettre de motivation) un ensemble d'annonces pouvant lui convenir.
Pour ce faire, on s'intéresse à diverses techniques:

1. en premier lieu on effectue une classification grâce à un algorithme de type *Latent Semantic Allocation* (LSA). On prendra ce modèle comme référence afin de tester les performances des modèles suivants.
2. ensuite on s'intéresse à des modèles de classification basés sur une représentation *word2vec* des documents et on se focalisera sur l'influence du traitement des termes complexes lors de la représentation de nos documents.

1 Introduction

Dans le domaine du traitement automatique de la langue, l'exploitation de la sémantique d'un document constitue un enjeu majeur dans divers domaines tels que la classification, la traduction ou l'extraction de connaissances. Pour ce faire, il est souvent nécessaire de savoir avec quelle granularité il faut aborder le problème, savoir s'il faut découper un document en lettres, en mots, en groupe de mots, en phrases voir en paragraphes afin de pouvoir extraire au mieux les caractéristiques qui nous intéressent.
Dans ce rapport, nous décrirons les différentes étapes d'une expérience visant à la classification de petites annonces extraites du web en fonction d'un CV ou d'une lettre de motivation en prenant en compte ou non les termes complexes.

2 Théorie

2.1 Représentation et Partitionnement de documents

La problèmatique que nous abordons, à savoir la recommandation d'offres d'emploi en fonction d'un CV, peut être vue comme une tâche de partitionnement non supervisé de données.
On entend par le terme partitionnement le regroupement de documents similaires (la notion de similarité étant à définir) afin d'obtenir des sous-ensembles homogènes et plus facilement utilisables que le groupe de documents initial.
Il existe déjà divers algorithmes permettant une classification non-supervisée de documents, notamment via le célèbre LSA, pour Latent Semantic Analysis, qui permet de représenter les documents selon les mots les plus discriminants qu'ils contiennent.
Dans notre cas, on s'intéresse à l'utilisation d'une représentation vectorielle de nos documents à partir de l'algorithme *word2vec* afin de savoir s'il est intéressant d'utiliser un tel espace dans la classification de documents. On s'intéressera aussi à la représentation de nos documents ansi qu'à l'influence de l'utilisation de termes complexes sur les performances de nos modèles.

Latent Semantic Analysis

L'algorithme *LSA* permet une classification des documents grâce à une représentation vectorielle des mots.
En premier lieu, les documents sont représentés par des matrices d'occurences de mots où, en général, les colonnes correspondent aux différents mots et les lignes correspondent aux différents documents. Afin d'assigner un poids à chaque mot, on le pondère selon l'inverse de sa fréquence dans les documents, autrement dit, on multiplie son nombre d'occurence par l'inverse de sa fréquence dans le corpus. Cela permet de donner un poids plus important aux mots qui n'apparaissent que dans peu de documents, que l'on utlisera pour caractériser des classes de documents.
Les dimensions de la matrice sont en général très grandes et creuses, c'est-à-dire que beaucoup de valeurs sont à 0 (le terme n'apparaît jamais dans le document). Afin de synthétiser la matrice, on a recours à une décomposition en valeurs singulières, ce qui permet de réduire le nombre de dimensions tout en faisant en sorte de préserver un maximum d'informations.

On peut alors calculer les similarités entre documents en comparant leurs représentations vectorielles réduites avec une mesure de telle que la similarité cosine. Cela permet, entre autres, de faire du clustering en rapprochant les documents avec de grandes similarités, il est alors possible d'utiliser presque n'importe quel algorithme de classification non supervisée afin d'obtenir nos clusters.

Word2Vec

Comme son nom l'indique, l'algorithme word2vec permet de représenter un mot en tant que vecteur. Pour avoir une représentation permettant de situer le mot dans l'espace selon sa sémantique, on lui assigne des coordonnées en fonction de son contexte (mots qui le suivent et qui le précédent).
L'algorithme nécessite de faire au moins 2 passes sur le corpus d'apprentissage. Durant la première passe, l'algorithme initialise son vocabulaire et donne à chaque mot une représentation aléatoire. Durant la seconde passe, l'algorithme modifie les coordonnées des mots en rapprochant ceux qui sont utilisés dans le même contexte tout en éloignant les autres, afin d'éviter que tous les mots ne convergent vers les mêmes coordonnées (notamment à cause des stopwords qui sont très utilisés quel que soit le contexte). On obtient alors un espace qui permet de capter le contexte d'utilisation d'un mot. On peut alors, grâce à ces valeurs numériques, calculer des distances ainsi que des similarités entre mots.

2.2 Utilisation de termes complexes

On qualifie de terme complexe (*Multiword Unit*, en anglais) un composé syntagmatique (ensemble de mots représentant un intéret sémantique en tant que groupe) tel que *droit privé*, *droit des affaires* ou encore *droit international*.
On se rend immédiatement compte de l'importance de ces termes lorsqu'il s'agit de comprendre une offre d'emploi ou un CV car les termes clés ne sont pas forcèment des termes simples . Ainsi dans le cas de la classification, on pourrait affiner le partitionnement en utilisant des termes très ciblés afin de rapprocher une demande d'une offre.

Extraction de termes complexes

Vue l'importance des termes complexes dans l'étude d'un document, on désire donc pouvoir les extraire et les traiter à part. Leur extraction est une tâche particulièrement délicate car il n'existe aucune technique permettant d'extraire à coup sûr l'ensemble des termes complexes d'un corpus. Les approches généralement étudiées sont soit linguistiques, soit statistiques, soit un hybride des deux.
Dans le cas linguistique, on s'intéresse à des patrons prédéfinis que l'on recherche au sein des étiquettes morpho-syntaxiques du corpus.
Dans le cas statistique, on se base, par exemple, sur la récurrence d'une suite de termes afin de repérer les candidats les plus susceptibles d'être des termes complexes.
Dans le cas d'une approche hybride, les deux techniques sont combinées, on peut, par exemple, après avoir extrait les suites de mots les plus récurrentes, les filtrer selon certains patrons morpho-syntaxiques.

LSA et termes complexes

Dans le cas d'une représentation LSA, on ne s'intéresse pas au pré-traitement des termes complexes. Ceci se justifie par la méthode de représentation utilisée. En effet, lors de la réduction de dimension, les informations contenues dans la matrice sont fortement synthétisées et il n'est pas rare que la représentation ne prenne plus seulement en compte les mots pris singulièrement, mais plutôt des groupes de mots jugés assez similaires.
C'est pour cette raison que l'on prend cet algorithme comme modèle de référence.

Word2Vec et termes complexes

L'algorithme word2vec, bien que très efficace en terme de projection de mot lorsqu'appris avec un corpus conséquent ne permet pas de projeter directement de termes complexes. On pourrait éventuellement utiliser la moyenne des représentations vectorielles de chaque mot composant le terme complexe, cependant cela ne serait efficace que si chaque occurence de chaque mot est apprise

dans le contexte du terme complexe. En effet, si les mots *droit* et *juridique* ne sont employés que l'un à la suite de l'autre dans le corpus d'apprentissage, on peut éventuellement supposer que la moyenne des deux donnerait une bonne représentation du terme complexe *droit juridique*.
Evidemment, ce type de cas ne se produit jamais, c'est pour cela que l'on optera pour une autre solution : commencer par repérer les termes complexes au sein de notre corpus, puis en faire une seule entité afin d'avoir une représentation non biaisée par les occurences des composantes du terme.

Représentation documentaire avec Word2Vec

L'algorithme de base de Word2Vec permet de représenter les mots dans un certain espace. Pour pouvoir représenter un document, on peut donc faire la moyenne de chaque terme le composant. Cependant, en prenant la totalité des termes du document pour le représenter, on risque en effet de prendre en compte beaucoup de bruit (*stop words* notamment). C'est pour cela que lors de cette étape, on effectuera un pré-traitement des termes (simples ou complexes) à utiliser pour représenter un document.

3 Pratique

3.1 Corpus

Annonces

Pour mener nos expériences, on se dote d'un corpus d'offres d'emplois recueillies par webscrapping sur le site www.monster.fr. On recueille en tout 6975 annonces, inégalement réparties entre les différentes catégories du site comme le montre la Table 1.
Les données ayant été recueillies directement, sans vérification faites sur leur contenu, on obtient donc un corpus bruité avec des annonces d'autres langues (anglais, allemand) ainsi que des structures irrégulières. De plus, il n'est pas exclu d'avoir des fautes d'orthographe, ce qui induira un biais au niveau de l'apprentissage de notre modèle. Il n'existe aucune garantie non plus que les annonces postées dans une certaines catégories appartiennent réellement à cette catégorie.
Le corpus dont on dispose étant assez petit, on comparera un espace appris seulement avec les annonces avec un espace déjà appris.

3.2 LSA

Le premier modèle auquel on s'intéresse est un modèle de clustering basé sur l'algorithme LSA. Ce modèle sera utilisé comme référence à surpasser par word2vec.
L'implémentation est faite en Python grâce à la bibliothèque sklearn.

3.2.1 Pré-traitement des données

Afin de faciliter le traitement de nos textes, on commence par les traiter afin d'avoir une représentation plus simple. En premier lieu, les annonces sont passées en minuscule, bien que les majuscules puissent permettre de reconnaître les noms d'entreprises, par exemple, ces derniers n'apportent pas réellement d'information lorsqu'il s'agit de rapprocher une annonce d'un CV.
Le découpage est fait par mot et on représente le corpus par une matrice tf-idf. Les mots apparaissant dans moins de 2 documents sont éliminés, cependant, on ne fixe pas de borne supérieure. Les stopwords (mots revenant souvent et n'apportant pas d'information, considérés comme du bruit) donnés par la bibliothèque nltk sont également supprimés.
Le choix est fait de ne pas faire de stemming (racinisation, réduction des mots à leur radical) afin d'éviter les confusions. Par exemple, le mot *développeur* peut être assez représentatif du domaine de l'informatique, s'il est racinisé, il sera confondu avec toutes les variantes du verbe *développer*, même si l'on pourrait résoudre le problème via le biais d'un étiquetage morpho-syntaxique, on fait le choix de ne pas y avoir recours.

3.2.2 Réduction de dimension

En prenant en entrée la matrice obtenue par le pré-traitement, on effectue une décomposition en valeur singulière (SVD) en fixant le nombre de dimensions à garder à 100. Cette valeur a été fixée

selon les informations obtenues sur la page web de la bibliothèque sklearn.

3.2.3 Partitionnement des données

Afin d'effectuer le clustering des données, on opte pour l'algorithme DBSCAN. Ce choix est fait car l'algorithme ne nécessite pas de spécifier le nombre de clusters, on pourra donc se servir de cette valeur pour l'évaluation du modèle. De plus, on utilise une distance cosine qui permet de rapprocher deux documents dont les dimensions réduites sont proches.

3.3 Word2Vec

Dans le cas de la représentation *word2vec*, on construit plusieurs espaces de représentation différents, afin, notamment, de vérifier l'impact du traitement des termes complexes.
L'implémentation est faite en Python grâce aux librairies gensim pour la représentation *word2vec* et sklearn pour le clustering avec *DBSCAN*.

3.3.1 Pré-traitement des données

Pour une représentation de type *word2vec*, le pré-traitement fait est beaucoup plus léger. Le corpus est encore une fois passé en minuscule, puis on effectue un découpage en phrases, en se basant sur les ponctuations et les retours à la ligne. Enfin, on élimine les ponctuations restantes et on obtient un découpage du corpus en phrases, sous forme de liste de mots.
Encore une fois, on fait le choix de ne pas faire de lemmatisation pour les mêmes raisons que pour l'algorithme *LSA*.

3.3.2 Espace basé sur le corpus avec termes simples

On construit un premier espace en prenant les mots tels quels, sans se soucier des termes complexes. Le corpus étant assez petit, la représentation est faite en seulement 100 dimensions.

3.3.3 Espace basé sur le corpus avec termes complexes

Avant de pouvoir entrer les termes complexes dans notre modèle, il faut tout d'abord se poser la question de leur extraction. On commence par utiliser les fonctionnalités offertes par la bibliothèque gensim qui repère les termes complexes par co-occurences, à savoir en comptant le nombre de bi-gram récurrents. Il est possible de passer aux tri-grams en appliquant récursivement le même algorithme.
La fonction s'appliquant directement aux listes de mots à entrer dans le modèle *word2vec*, on obtient les phrases découpées telles que les bi-grams sont réunis en une seule entité. Par exemple, chaque co-occurence des mots *data* et *science* est remplacée par *data_science*.

3.3.4 Espace basé sur wikipédia avec termes simples

A titre de comparaison, on utilise aussi un modèle à 500 dimensions déjà entraîné sur la version française de wikipédia. Ce dernier ne prend en compte que les termes simples.
Les structures de phrases de wikipédia étant assez régulières et le corpus étant conséquent, on suppose que l'espace de représentation est déjà correctement optimisé.
Comme il n'est pas possible de modifier le vocabulaire d'un espace déjà entraîné, on prend le modèle tel qu'il est.
Lien de téléchargement : https://zenodo.org/record/162792

3.3.5 Représentation des documents dans un espace *word2vec*

Avant d'avoir recours au clustering, il est nécessaire de se demander quelle est la meilleure manière de représenter une offre d'emploi.
Dans notre cas, nous nous intéresserons à 2 cas de figures :

1. une représentation "fréquentielle" faite en sommant les projections des mots de l'annonce après avoir éliminé les stop words. Dans ce cas, les mots pleins récurrents sont pris en compte autant de fois qu'ils apparaissent.

2. une représentation "présentielle" qui ne prend en compte la projection d'un mot plein que s'il est présent dans le texte. Pour ce faire, on modèlise notre texte par un set de mots, ce qui permet d'éliminer les doublons.

3.3.6 Partitionnement des données

Tout comme pour *LSA*, on effectue un clustering via l'algorithme DBSCAN de la bibliothèque sklearn en prenant en compte une distance cosine.

Catégorie	Nombre d'annonces
Architecture, Creation et spectacle	142
Edition et Ecriture	315
Ingénierie	399
Marketing	272
Ressources Humaines	390
Services Administratifs	392
BTP et second oeuvre	371
Informatique et Technologies	399
Logistique et Réparation	288
Recherche et Analyses	173
Sécurité	24
Commercial/Vente	392
Gestion de projet/Programme	392
Juridique	191
Qualité/Inspection	397
Santé	398
Stratégie et Management	392
Comptabilité et Finance	153
Installation et Réparation	394
Production et Opérations	395
Restauration et Hôtellerie	296
Services Clientèle	388
Total	6975

Table 1: Répartition des annonces selon les catégories.

3.4 Expérimentations

Nous commençons tout d'abord par construire les différents espaces word2vec que nous utiliserons pour nos expériences. On en considère 4 :

UnigramAnnonces espace construit à base d'unigrammes du corpus d'annonces.

BigramAnnonces espace construit à partir des unigrammes et des bigrammes des annonces. Les bigrammes sont déduits grâce aux co-occurences dont on fixe le nombre minimal à 5.

TrigramAnnonces espace construit à partir des unigrammes, bigrammes et trigrammes des annonces. Ces derniers sont extraits en utilisant la même fonction qui permet de passer d'unigramme à bigrammes en prenant en entrée les bigrammes.

UnigramWiki espace contenant uniquement des unigrammes déjà entraîné récupéré sur https://zenodo.org/record/162792

La prise en compte des termes complexes influe sur la taille du vocabulaire considéré par chaque modèle. On consigne dans la Table 2 la taille du vocabulaire de chaque espace.

3.4.1 Clustering et évaluation

Afin d'avoir une idée de l'efficacité des différents algorithmes, on se réfère à un clustering de type DBSCAN, qui permet de créer des partitions en fonction de la distance des points dans l'espace. De plus, l'algorithme permet aussi de classer des points comme étant du bruit, ce qui permet de pallier aux faiblesses de notre corpus.
Les annonces ayant été recueillies selon 22 catégories, on souhaiterait que le partitionnement se fasse avec au moins autant de cluster. Dans l'idéal, le nombre de clusters devrait être plus élevé afin de séparer le plus finement possible les annonces.
Une fois l'algorithme DBSCAN utilisé, on vérifie la pureté des clusters selon les catégories des annonces qui les composent. Il est à noter aussi qu'il existe un cluster de bruit (noté cluster -1) contenant tous les points que le classifieur considère comme étant trop loin des autres pour faire partie d'un quelconque cluster.
Etant donné que l'on désire trouver un clustering le plus fin possible, nous faisons varier la distance

maximale prise en compte par le classifieur afin de considérer deux exemples comme étant voisins. Les résultats des différentes configurations de DBSCAN testées sont consignées dans les Tables 3 et 4.
Enfin, dans le but de mesurer l'homogénéité des clusters, on calcule la pureté de chaque groupe. Pour ce faire, on prend le nombre d'annonces correspondant à la catégorie la plus représentée dans la partition que l'on normalise par le nombre total d'annonces du cluster.

Modèle	Taille du Vocabulaire
UnigramAnnonces	14 132
BigramAnnonces	19 474
TrigramAnnonces	20 659
UnigramWiki	475 475

Table 2: Taille du vocabulaire en fonction du modèle *word2vec*

Modèle	**epsilon**									
	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
LSA										
Nombre de clusters	56	64	74	87	102	115	124	149	162	151
Taille du cluster de bruit	6393	6273	6146	5988	5745	5494	5206	4835	4330	3634
Pureté moyenne des clusters	0.973	0.967	0.941	0.939	0.907	0.891	0.854	0.831	0.817	0.801
Variance de la pureté des clusters	0.012	0.012	0.024	0.022	0.026	0.033	0.045	0.050	0.053	0.055
UnigramAnnonces										
Nombre de clusters	75	110	26	14	7	5	3	2	2	2
Taille du cluster de bruit	6176	3825	857	246	103	46	27	13	10	8
Pureté moyenne des clusters	0.942	0.845	0.906	0.885	0.815	0.748	0.495	0.057	0.057	0.057
Variance de la pureté des clusters	0.018	0.046	0.048	0.062	0.116	0.160	0.192	0.0	0.0	0.0
BigramAnnonces										
Nombre de clusters	71	90	23	13	9	5	3	3	2	2
Taille du cluster de bruit	6191	3230	625	180	67	26	12	10	7	6
Purete moyenne des clusters	0.943	0.860	0.905	0.856	0.874	0.748	0.529	0.529	0.057	0.057
Variance de la purete des clusters	0.022	0.045	0.055	0.096	0.096	0.160	0.222	0.222	0.222	0.0
TrigramAnnonces										
Nombre de clusters	71	83	26	15	8	4	3	3	3	2
Taille du cluster de bruit	6179	3196	634	193	69	29	12	10	7	5
Purete moyenne des clusters	0.934	0.855	0.916	0.914	0.856	0.664	0.529	0.529	0.529	0.057
Variance de la purete des clusters	0.024	0.048	0.049	0.059	0.107	0.184	0.222	0.222	0.222	0.0
UnigramWiki										
Nombre de clusters	44	56	67	95	82	56	36	28	16	13
Taille du cluster de bruit	6550	6447	6295	5868	4838	3414	2243	1444	984	686
Purete moyenne des clusters	0.991	0.990	0.959	0.886	0.916	0.891	0.897	0.871	0.882	0.862
Variance de la purete des clusters	0.002	0.002	0.011	0.036	0.029	0.041	0.055	0.064	0.072	0.090

Table 3: Résultats de l'algorithme DBSCAN sur chaque modèle avec représentation fréquentielle

Modèle	**epsilon**									
	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
UnigramAnnonces										
Nombre de clusters	88	55	20	8	3	3	3	2	2	2
Taille du cluster de bruit	5934	2295	488	160	65	29	16	8	6	5
Purete des moyennes des clusters	0.913	0.864	0.858	0.851	0.529	0.529	0.529	0.057	0.057	0.057
Variance de la purete des clusters	0.032	0.046	0.078	0.110	0.222	0.222	0.222	0.0	0.0	0.0
BigramAnnonces										
Nombre de clusters	94	42	16	10	7	4	3	3	2	2
Taille du cluster de bruit	5866	1620	364	127	43	15	9	6	4	4
Purete moyenne des clusters	0.889	0.880	0.847	0.895	0.843	0.686	0.529	0.529	0.057	0.057
Variance de la purete des clusters	0.036	0.060	0.076	0.087	0.123	0.197	0.222	0.222	0.0	0.0
TrigramAnnonces										
Nombre de clusters	84	41	19	10	8	3	3	3	3	2
Taille du cluster de bruit	5897	1604	357	121	47	15	10	5	4	4
Purete moyenne des clusters	0.884	0.839	0.872	0.895	0.865	0.529	0.529	0.529	0.529	0.057
Variance de la purete des clusters	0.043	0.068	0.068	0.087	0.109	0.222	0.222	0.222	0.222	0.000
UnigramWiki										
Nombre de clusters	45	53	76	91	74	47	26	18	15	11
Taille du cluster de bruit	6539	6449	6234	5607	4281	2905	1898	1256	834	574
Purete moyenne des clusters	0.990	0.990	0.928	0.888	0.904	0.885	0.923	0.863	0.881	0.833
Variance de la purete des clusters	0.002	0.002	0.022	0.037	0.035	0.048	0.046	0.076	0.079	0.104

Table 4: Résultats de l'algorithme DBSCAN sur chaque modèle avec représentation présentielle

3.5 Analyses

Le clustering par DBSCAN vise à évaluer nos espaces. En effet, si l'algorithme arrive à découvrir assez de clusters distincts avec un taux de pureté assez élevé et un bruit assez bas, on peut se dire que le modèle de représentation peut convenir les représentations dans l'espace des annonces provenant de la même catégorie sont proches.

En observant les Tables 3 et 4, on constate que seule la représentation fréquentielle donne des partitionnements avec un nombre de groupes intéressants (autour de 22). Notamment pour une valeur epsilon de 0.03, on remarque que les espaces basés sur le corpus d'offres d'emploi donnent un nombre de clusters correspondant à nos attentes. En examinant de plus près cette colonne, on constate aussi que la prise en compte de termes complexes a permis de limiter les annonces considérées comme du bruit, ce qui veut dire que dans notre espace de représentation, les points isolés des autres sont beaucoup moins présents.
On observe aussi une variation du nombre de cluster lors du passage d'unigramme à bigramme qui passe de 26 à 23, cependant ce dernier revient à 26 lorsque l'on prend en compte les trigrammes.

Il est à noter aussi que les espaces construits par les annonces donnent largement de meilleurs résultats que la représentation LSA (qui est en 100 dimensions aussi) et celle basée sur le modèle entraîné sur Wikipédia (500 dimensions).

Choix de la représentation

Après lecture des données recueillies via DBSCAN, on peut déduire que l'espace de représentation le plus approprié à notre corpus est un word2vec fréquentiel prenant en compte les trigrammes.

4 Application

Une fois l'espace de représentation fixé, on peut à présent mettre en oeuvre un système de recommandation d'offre d'emploi basé sur un CV.
L'application prend en entrée le texte d'un CV et construit un dossier contenant les 10 offres d'emploi (tirées du corpus) les plus intéressantes. Ces dernières sont sélectionnées selon leur similarité cosine avec le CV. Le découpage en termes de ce dernier est fait par le Phraser construit par le corpus lors de l'apprentissage de l'espace. De cette manière, les termes complexes pris en compte sont les mêmes que ceux du corpus.

Utilisation

L'application est en fait un script Python exécutable depuis le terminal linux. Pour ce faire, il suffit de spécifier au script "app.py" le chemin vers le fichier contenant le CV.
L'application renvoie les 10 annonces maximisant la similarité cosine avec le document en entrée en les copiant dans un dossier spécifique.

Tests de prédiction

Afin de réellement pouvoir évaluer l'efficacité de l'application, il faudrait organiser une évaluation en prenant de vrais CV puis en faisant évaluer les prédictions de notre algorithme par les propriétaires des CV considérés.

5 Conclusion

En conclusion, on peut affirmer que le traitement de termes complexes dans un espace *word2vec* permet bel et bien de différencier plus nettement les documents. Cela se déduit depuis les résultats de l'algorithme DBSCAN qui arrive à trouver de meilleurs clusters.
Cependant, dans le cas de traitement de CV, il serait plus pertinent de séparer les différentes parties de ces derniers en compétences et expériences par exemple. En faisant cela, on pourrait avoir une meilleure représentation de nos documents et les rapprocher selon des caractéristiques spécifiques.