Master 1 : Données, Apprentissage et Connaissances (DAC)

Rapport PLDAC

Analyse d'un corpus du CV, apprentissage de représentation sur des documents structurés et démêlage des facteurs explicatifs

Membres :

HADDADI Hacene

AKNOUCHE Anis

Encadrants :

GUIGUE Vincent

Année universitaire : 2020/2021

Table des matières

I) Introduction : 3

II) Partie I 3

A) Prise en main des données : 3

B) Pré-traitement des données et modèles : 8

1) Pré-traitement : 8

2) Modèles : 11

3) Méthodologie : 12

C) Compagne d'expérimentation : 13

i) Apprentissage supervisé : 13

ii) Apprentissage non supervisé : 19

III) Partie II : 23

1) Prise en main des données : 23

2) Méthodologie : 24

3) Campagne d'expérimentation : 25

4) Modélisation des profils : 30

IV) Conclusion : 35

I) **Introduction :**

De nos jours , de nombreuses entreprises reçoivent un grand nombre de CV en ligne. En outre, la numérisation de ces derniers a ouvert de nombreuses options pour l'industrie ainsi que pour les demandeurs d'emploi, le défi pour le fournisseur d'emploi est donc de séparer la qualité de la quantité.

La présélection de candidats prend beaucoup de temps et impliquent de nombreux efforts humains. Ainsi, en automatisant le système de classification des CV selon le domaine industriel le temps et les efforts peuvent être réduits dans le processus de présélection des CV tout en augmentant l'efficacité.

A cela s'ajoute la modélisation de l'expérience des candidats et la détection de carrière qu'une entreprise pourra utiliser pour cibler les profils recherchés et faire de la recommendation de jobs.

Dans la suite, on mettera en évidence la méthodologie suivie et les différentes techniques de machine learning associées au problème de classification de CV ainsi que pour la modélisation de l'expérience et la détection de carriere.

II) Partie I :

A) Prise en main des données :

Nous avons à disposition un fichier contenant 5000 CV, avant toute chose il faut comprendre et analyser la strcuture du fichier.

- Le fichier est sous forme d'une liste de CV, chaque CV est composé de jobs, des compétences, d'un cursus scolaire et du domaine industriel.

- Chaque jobs est définie par un nom, sa date de début et de fin, une déscriprtion des différentes tâches éffectués et de la compagnie qui a proposé ce job.

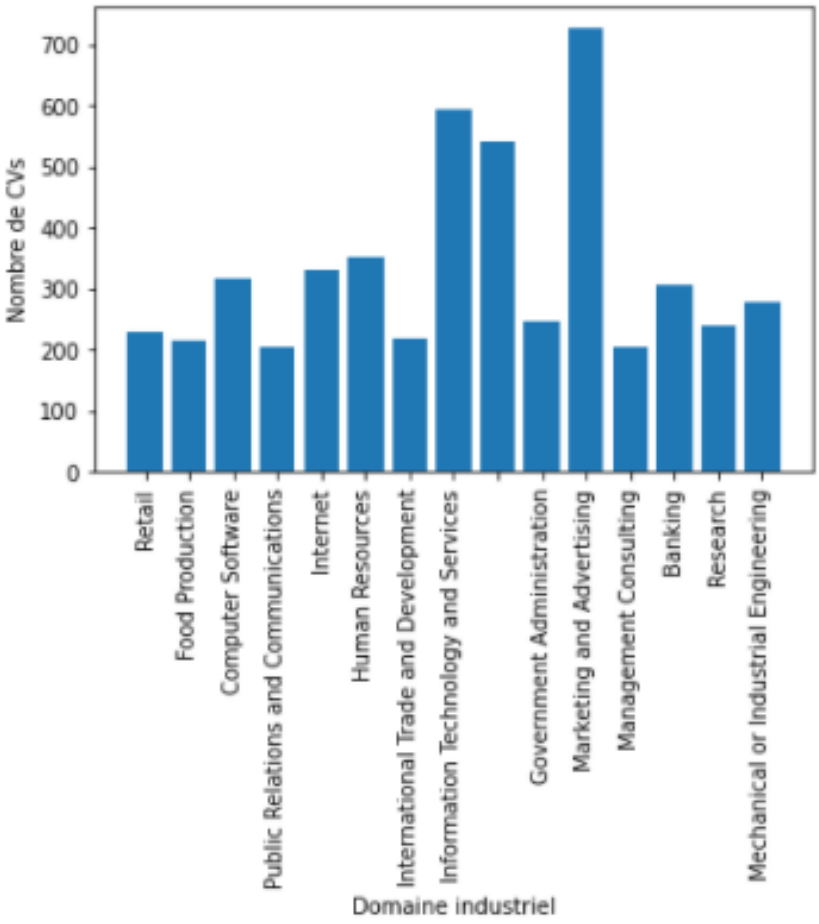
- Une compétence est une liste des différents outils et technologies maîtriser par l'individu.

- Le cursus scolaire est une liste des diplômes obtenus, avec l'année de début et de fin de la formation et l'institut dans lequel est effectué la formation.

- Le domaine industriel correspond au domaine dans lequel travaille l'individu.

i) **Nombre de CV par domaine :**

Nos données sont classées selon le domaine industriel, affichons les différents domaines que nous avons ainsi que la répartitions des CV sur ces derniers.



Remarque :

La première chose qu'on remarque avec la figure est la présence de nombreux CVs dont on ne possède pas l'étiquette 'industry', il faut donc les supprimer car ils n'apportent pas d'information pour la partie supervisée de la classification de documents.

On remaque aussi que les classes « Information Technology and Services » et « Marketing and Advertising » sont surreprésentées dans notre dataset, la distribution des autres classes quant à elles est assez unifome.

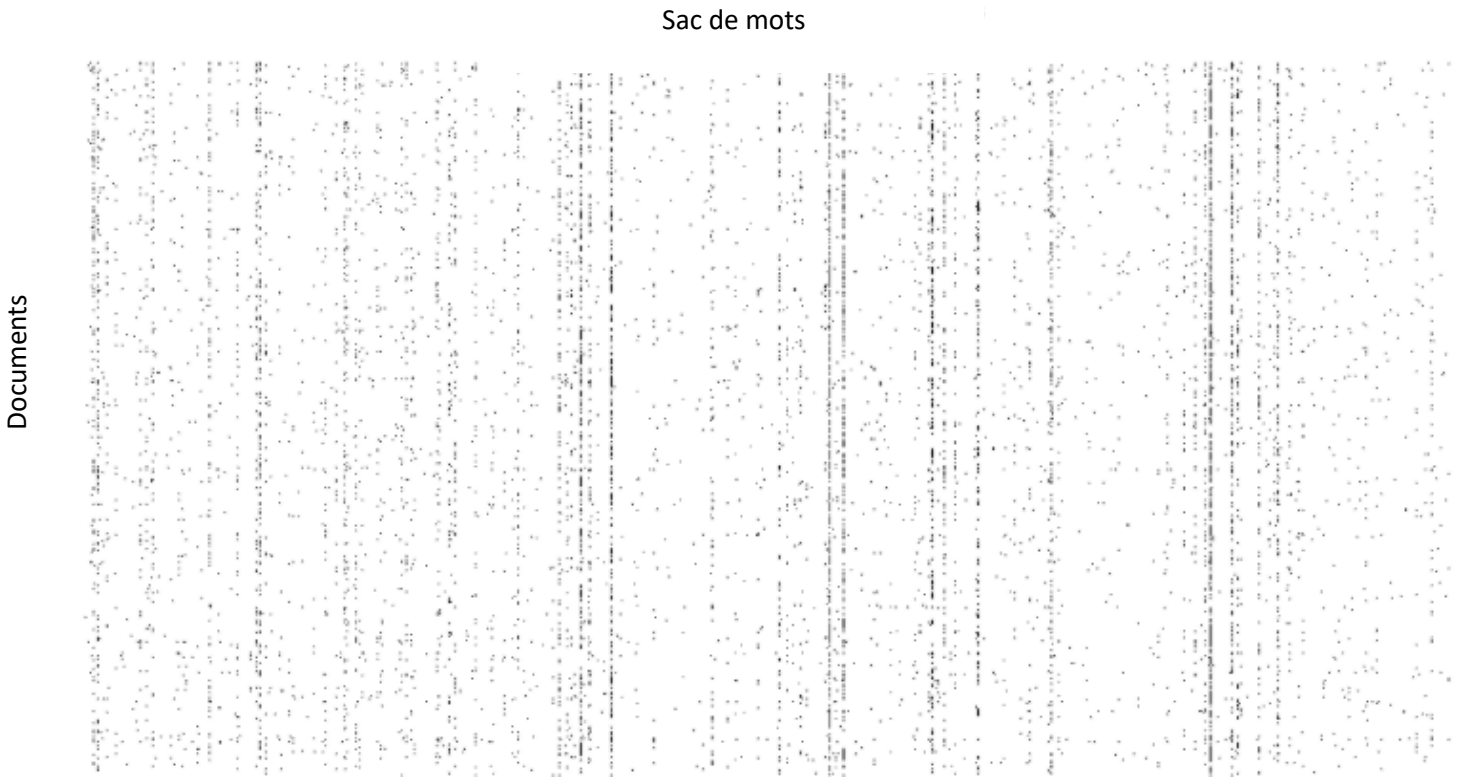
ii) <u>Mesure de la sparsité :</u>

L'approche classique pour travailler sur un problème de TAL et pourvoir faire de la classification dans notre cas, est de numériser nos documents, pour cela on définie un dataset avec comme dimension un sac de mots et pour chaque ligne un document. Chaque case de cette matrice correspond au nombre d'occurrence du mot dans le document.

Le problème de ces matrices est le fait qu'elles soient parcimonieuses, mesurer la sparsité permet donc de connaître le nombre de mots actif en moyenne par document.

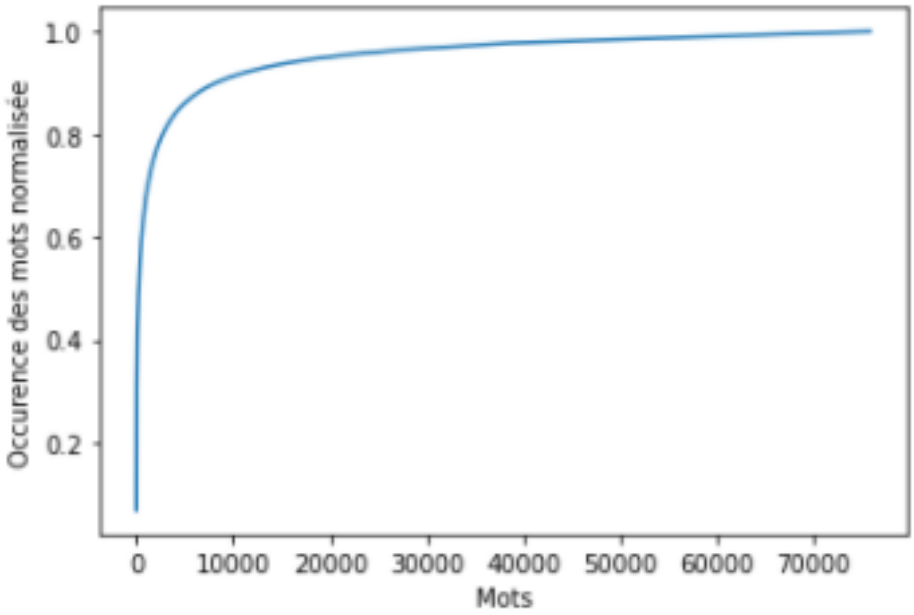
Ici la mesure est de : 208 mots par documents sur 75858 mots.

La figure suivante affichant notre matrice sparse met en évidence ce phénomène.



iii) <u>Occurrence des mots :</u>

Notre sac de mots à une taille de 75858 mots, on s'intéresse ici à leurs occurrences cumulées.



Remarque :

La figure ci-dessus met en évidence la loi de zipf, autrement dit, dans notre vocaublaire, nous avons très peu de mots qui reviennent beaucoup de fois, et un grand nombre de mots qui ne reviennent que peu de fois.

iv) Words Cloud :

Comme notre but dans un premier temps est de faire de la classification de documents, il serait intéressant d'afficher les words cloud pour visualiser les mots les plus fréquents des différentes classes.

Food Production

Government Administration

Human Resources

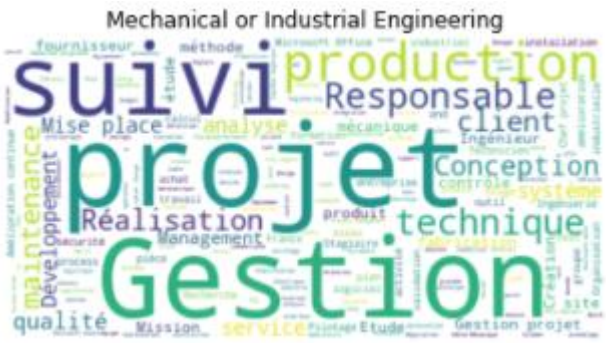
Information Technology and Services

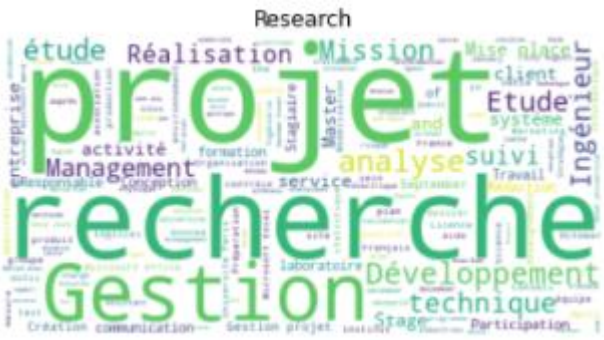
International Trade and Development

Internet

Management Consulting

Marketing and Advertising





B) Pré-traitement des données et modèles :

Le but de cette séction est de déterminer la méthodologie à suivre durant la campagne d'éxpérimentation.

La première partie de ce projet consiste à classifier des documents, nous devrons donc pré-traiter nos données et ensuite apprendre et tester des modèles dessus pour répondre à cette problématique.

Nous allons dans un premier temps présenter les pré-traitments que nous avons appliqué, et par la suite les différents modèles utilisés.

1) Pré-traitement :

Dans tout problème de TAL, le pré-traitement des données est une des tâches les plus importantes car c'est ce qui permet en général d'obtenir de bons résultats.

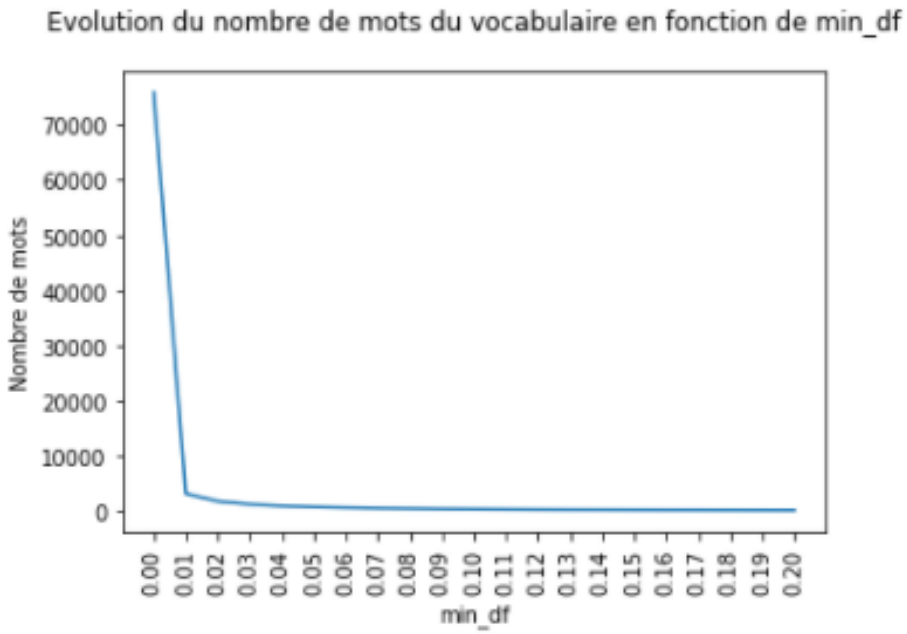
La plupart des méthodes de pré-traitement ont pour but de faire varier la taille de notre vocabulaire, essayons donc de voir l'évolution de ce dernier pour quelques-unes de ces méthodes.

Rappelons que sans pré-traitement nous avons une matrice sparse avec :

- 5000 lignes (documents)
- De dimension 75858 (taille du vocabulaire initiale)

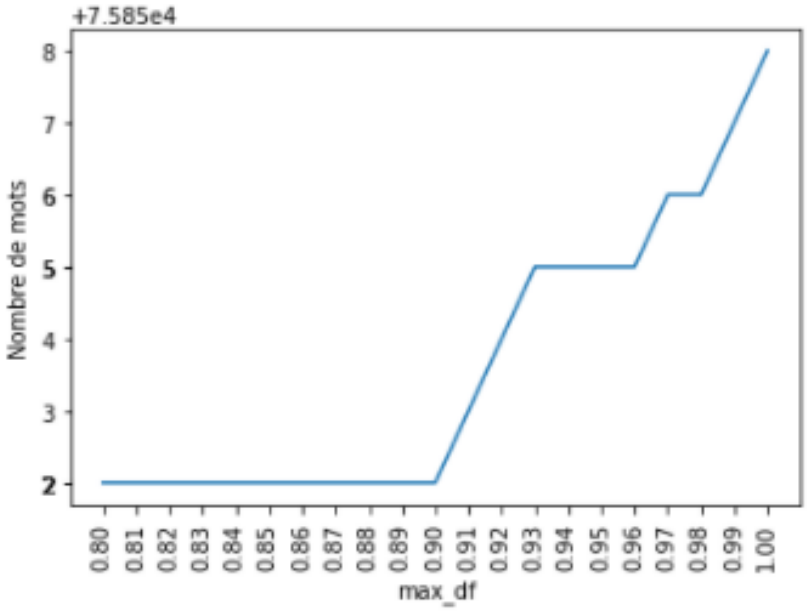
i) **<u>Suppression des mots rares :</u>**

Nous savons que les mots rares sont inclus dans les mots avec une fréquence documentaire faible, faisons donc varier le seuil de fréquence documentaire minimum admis pour les mots et affichons l'évolution de la taille du vocabulaire.



ii) **Suppression des mots avec une grande fréquence documentaire :**

Nous faisons varier le seuil de fréquence documentaire maximum admis pour les mots et affichons l'évolution de la taille du vocabulaire



Remarque :

Les deux figures ci-dessus confirment bien le fait qu'on ne possède que très peu de mots très fréquents, et énormément de mots peu fréquents, de petits changements dans min_df impliquent une grande réduction de la dimension du dataset et à l'inverse une plus grande varaition de max_df induit un changement négligeable.

iii) **Stop words, Stemming , Lowercase, token-pattern et n-gram:**

- Les strop words sont parmis les mots les plus présents dans les doccuments . Souvant, ils n'apportent pas d'informations, mais ca peut s'avérer utile de les laisser dans certain cas.

- Le stemming permet de trouver les radicaux des mots de manière brut, ce qui permet de regrouper les mots et dépasser les barière du langage tel que le féminin, le pluriel… Mais appliquer le stemming n'est pas forcément synonyme de meilleurs résultats car ca induit aussi une perte d'information.

- Nos machines sont sensibles à la casse, par exemple si l'on rencontre dans nos documents les mots « Bonjour » et « bonjour », ils seront interprété comme deux mots différents, en passant tous les mots à la minuscule on résout ce problème, mais dans certain cas il est intéressant de laisser cette distinction.

- Token-pattern indique le pattern que doivent respecter les mots de notre vocabulaire, il nous permettera donc de supprimer la ponctuation, les mots qui ne sont pas en caractère latin mais aussi les chiffres.

-On peut aussi réfléchir à décomposer nos document en n-gram, ce qui veut dire qu'au lieu de prendre comme token un mot, on prend une séquence de n mots successif dans le document, mais cela implique un cout plus important en mémoire et en temps de calcul.

Observons la variations de la dimension de notre vocabulaire en fonction de ces différents pré-traitements :

	stemming	Stop-words	Token-pattern	Lowercase
Sans	75858	75858	75858	75858
Avec	43160	75645	71824	58131

iv) **Matrice TF-IDF :**

Il peut être intéressant de passer de la matrice comptant le nombre d'occurrences des mots vers la matrice tf-idf, et ce car cette dernière permet de donner un score présentant l'importance d'un mot en prenant en compte sa fréquence docuementaire et son nombre d'occurrence dans le document, cette méthode permet donc de mieux saisir les mots discriminants.

Remarque :

Une réduction trop grande de la dimension de notre dataset peut conduire à une importante perte d'information, dans l'autre cas, une dimension trop grande implique de plus long calculs et un risque de sur-apprentissage. Dans les deux cas on a une dégradation des performences de nos modèles, il faut donc trouver la meilleure combinaison de pré-tratiements de sorte à maximiser ces performences.

2) Modèles :

Nous avons à disposition des modèles connus comme étant robuste pour les problèmes de TAL, et ceux utilisés dans le cadre de ce projet sont :

- **Multnomial Naive Bayes :**

  MultinomialNB implémente l'algorithme de Bayes de façon naïve pour des données distribuées de manière multinomiale

- **Régréssion logistique :**

  La régression logistique, malgré son nom, est un modèle linéaire de classification plutôt que de régression. Dans ce modèle, les probabilités décrivant les résultats possibles d'un seul essai sont modélisées à l'aide d'une fonction logistique.

- **Support Vector Machines (SVM) :**

  Les SVM sont un ensemble de techniques d'apprentissage supervisé destinées à résoudre des problèmes de discrimination et de régression. Les SVM sont une généralisation des classifieurs linéaires.

- **Latent Dirichlet Allocation (LDA) (Apprentissage non supervisé):**

  LDA est un modèle probabiliste génératif pour les collections de données discrètes telles que les corpus de textes. Il s'agit également d'un modèle thématique utilisé pour découvrir des thèmes abstraits à partir d'une collection de documents.

3) <u>Méthodologie</u> :

Dans un premier temps, nous allons tester une approche supervisée, nous savons que dans nos données nous disposons de 560 CV non étiquetées, or pour la classification de documents en supervisé il est primordiale d'avoir les labels, on doit donc les supprimer de notre dataset . Nos données sont aussi déséquillibrées (deux classes sont surreprésentées), nous allons donc travailler sur nos données avec et sans équillibrage pour comparer les performances.

Nous savons que pour tout problème de classification, nous devons avoir une baseline qui définit les performances minimales que nos modèles doivent dépasser, il faut donc choisir la meilleure baseline possible en fonction de nos données (classes majoritaire, stochastique …).

Une fois la baseline définie, nous allons tester les différents modèles présentés ci-dessus.

Chaque modèle possède des hyper-paramètres que le data scientist doit définir lui-même et une des façons de faire est le GridSearch, un optimisateur de parmètres intégré à sklearn qui prend en paramètres un classifieur et différentes valeurs de ses hyper-paramètres, et renvoi la combinaison qui maximise les perfomrances de la validation croisée.

L'idéal dans notre cas serait de lancer ce GridSearch sur chaque combinaison de pré-traitements possible et ce pour tous les modèles, mais le temps de calcul nécéssaire ne nous permet pas d'opter pour cette option, à la place nous allons prendre en considération uniquement le MultinomialNB car c'est un classifieur plutôt bien adapté pour les données textuelles mais surtout son temps d'apprentissage et de test nécéssaire est relativement faible comparé aux autres classifieurs, nous allons ensuite lancer son GridSearch avec nos données **non traitées** pour obtenir les meilleurs hyper-paramètres et pour cette configuration, nous allons tester

toutes les combinaisons de pré-traitments et prendre celle qui maximise nos performances selon la validation corisée.

Une fois la combinaison de pré-traitements optimale obtenu avec le MultinomialNB, nous pouvons lancer les GridSearch des autres modèles uniquement sur cette dernière, on pourra ensuite comparer les résultats des différents modèles.

Dans un second temps, nous allons tester l'approche non supervisée, cette dernière est plus délicate à mettre en oeuvre car nous n'exploitant pas les labels, il faut donc se reposer sur des analyses qualitatives et quantitatives pour juger les performances.

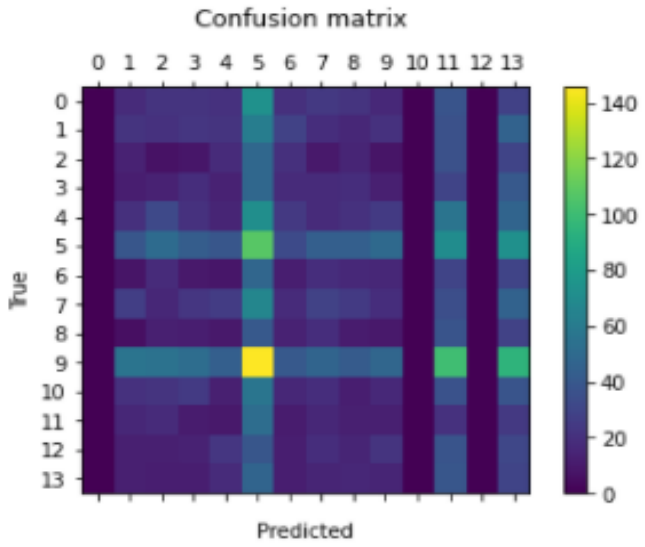
C) Compagne d'expérimentation :

i) Apprentissage supervisé :

1) Baseline :

Prenons dans un premier temps une baseline aléatoire qui prédit selon une probabilité uniforme :

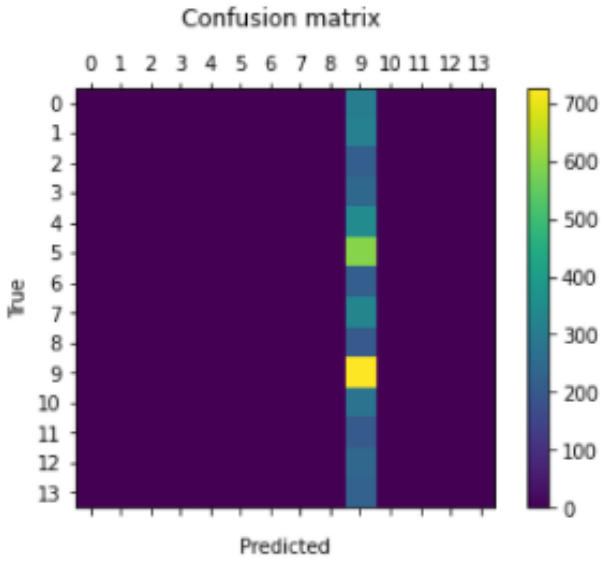
```
Accuracy (Baseline) : 7.22 %
```



On peut voir qu'on obtient des scores très faibles.

Testons maintenant une baseline qui classe tous les documents dans la classe majoritaire :

```
Accuracy (Baseline) : 16.30 %
```



On obtient de meilleurs résultats que la précedente baseline, on prendra donc cette dernière comme modèle de réference.

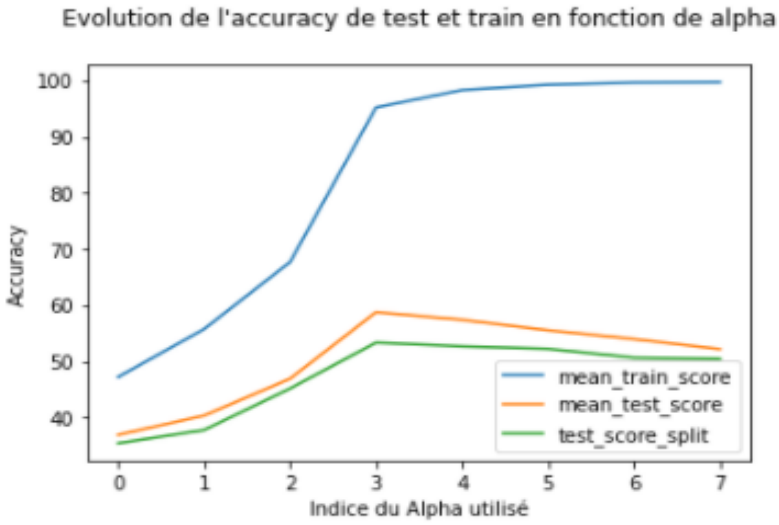
2) Multinomial Naive Bayes :

i) GridSearch sur les paramètres du classifieur :

On lance un gridSearch sur le multinomialNB.

L'hyperparamètre du multinomialNB est alpha, on teste les valeurs suivantes :

```
[2, 1.5, 1.3, 1, 0.1, 0.01, 0.001, 0.0001]
```



Le meilleur modèle est obtenue pour un alpha = 1, pour les performances suivantes :

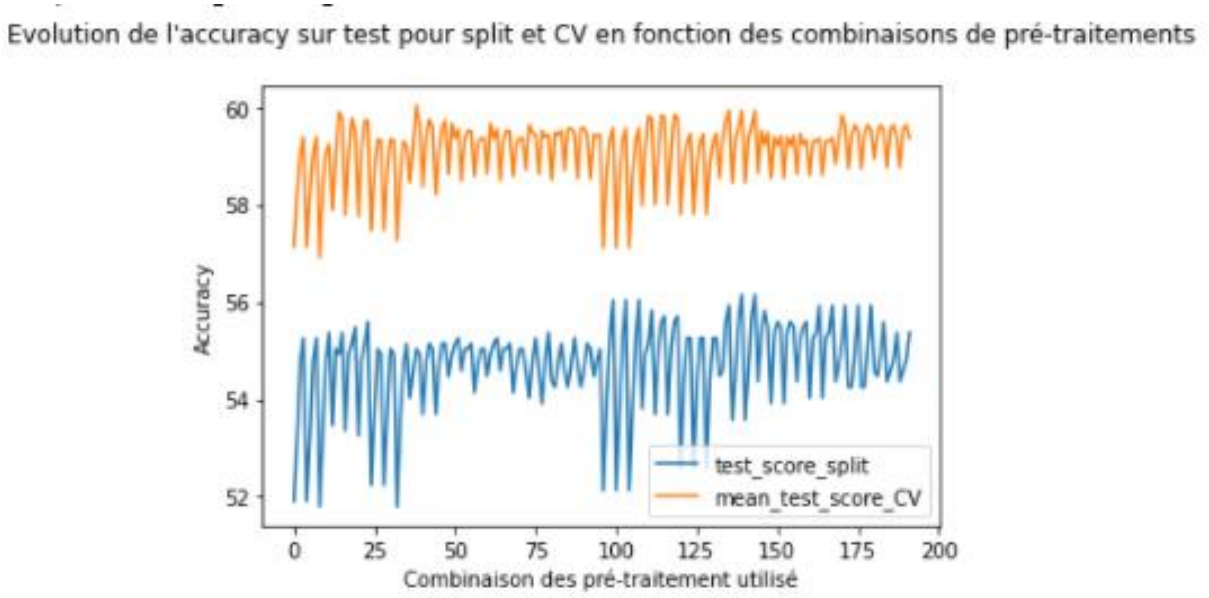
Acurracy_test_CV	Accuracy_test_split
58%	53%

- **Train_test_split (80% train et 20% test) :**
  On obtient une accuracy de 53% en test

- **Cross validation (CV=5) :**
  On obtient une accuracy de 58 % en test

ii) <u>Combinaison des pré-traitements de données:</u>

Nous allons ici travailler avec le meilleur MultinomialNB obtenu ci-dessus et trouver la meilleure combinaison de pré-traitements.



La figure met en évidence l'évolution de l'accuracy sur les données test en split simple (80% train 20% test) et en validation croisée (CV=5) et ce en fonction des différentes combinaisons de pré-traitements.

Combinaison optimale :

La combinaison optimale obtenu est :

- Utilisation de la matrice des occurrences.
- Suppression des stopwrods.
- Suprresion de la ponctuation, des chiffres et des mots en caractères non latin.
- Mise en minuscule des mots de notre corpus.
- Suppression des mots avec un DF ≤ 5 et DF ≥ 0.9
- Bow sous forme d'uni-gramme

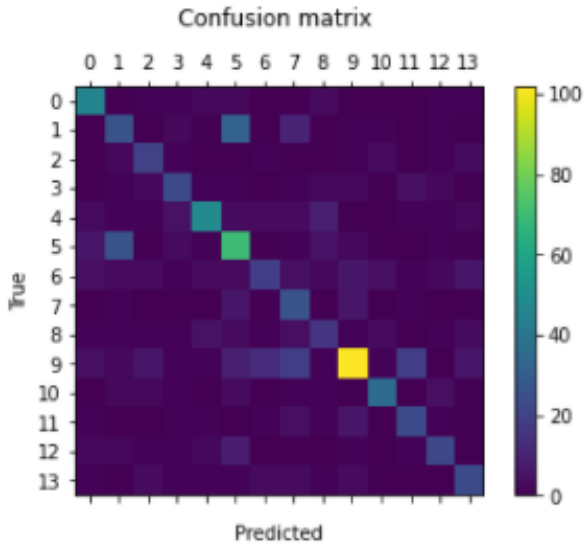
Stopwords	Stemming	LowerCase	Token-pattern	n-gram	min_df	max_df
True	False	True	True	(1,1)	5	0.9

Après ces pré-traitements, notre bag of words à une taille de : 11873 mots.

Résultats pour le meilleur MultinomialNB :

Une fois les meilleurs hyper-paramètres et combinaison de pré-traitement obtenu, nous pouvons afficher les résultats de notre modèles.

Accuracy_train_CV	Acurracy_test_CV	Accuracy_test_split
78%	60%	56%



Classification report :

	precision	recall	f1-score	support
0	0.68	0.81	0.74	57
1	0.39	0.36	0.37	73
2	0.50	0.62	0.56	32
3	0.55	0.59	0.57	39
4	0.74	0.60	0.67	81
5	0.50	0.61	0.55	115
6	0.46	0.31	0.37	61
7	0.34	0.65	0.45	40
8	0.39	0.39	0.39	41
9	0.78	0.55	0.65	184
10	0.75	0.72	0.73	50
11	0.46	0.59	0.52	39
12	0.59	0.55	0.57	40
13	0.52	0.57	0.55	40
accuracy			0.56	892
macro avg	0.55	0.57	0.55	892
weighted avg	0.59	0.56	0.56	892

Remarque :

On analysant la matrice de confusion ainsi que les différents f1-scores des classes, nous observons une certaine difficulté rencontré par notre modèle pour distinguer certains domaines entre eux, par exemple les classes 5 et 1 qui correspondent respectivement à *« Computer Software »* et *« Information Technology and Services »*.

Affichons les mots les plus fréquents pour ces deux classes.

On remarque que les deux classes ont beaucoup de mots en commun, et donc le problème rencontré par notre modèle dans la distinction de ces dernières se situe au niveau du vocabulaire utilisé dans les CV des deux domaines indusriels, on peut donc faire l'hypothèse qu'ils possèdent un jargon similaire ce qui induit une complication au niveau de la classification.

Pour y remédier, nous pouvons réfléchir à mettre en œuvre d'autres métriques qui prennent en considération ce phénomène.

Nous avons opté pour la métrique « top-n », qui dans notre cas considère qu'un document est bien classifié si notre modèle nous renvoie sa véritable classe dans les *n* classes plus probables.

On obtient donc les résultats suivant pour un split classique des données :

	Top 1	Top 2	Top 3
MultinomialNB	56%	76%	84%

Nous remarquons une augmentation conséquente des performances en test.

Equilibrage des données :

Pour l'équilibrage de nos données, nous avons opté pour un sous-échantillionnage des classes surreprésentées, nous obtenons ainsi une amélioration de 4% en accuracy en test pour la validation croisée, nous pouvons donc conlcure que le gain en performance ne justifie pas la perte d'information causée par le sous-apprentissage.

Performances des différents modèles :

Les meilleurs hyper-paramètres trouvés pour nos modèles sont :

- **Svm** : C=0.001 , kernel='linear'
- **Logistic regression** : C=0.01, penalty='l2' et solver='lbfgs'

	Accuracy train CV	Accuracy test CV	Accuracy test split
MultinomialNB	78%	60%	56%
LogisticRegression	89%	59%	55%
SVM	89%	60%	55%

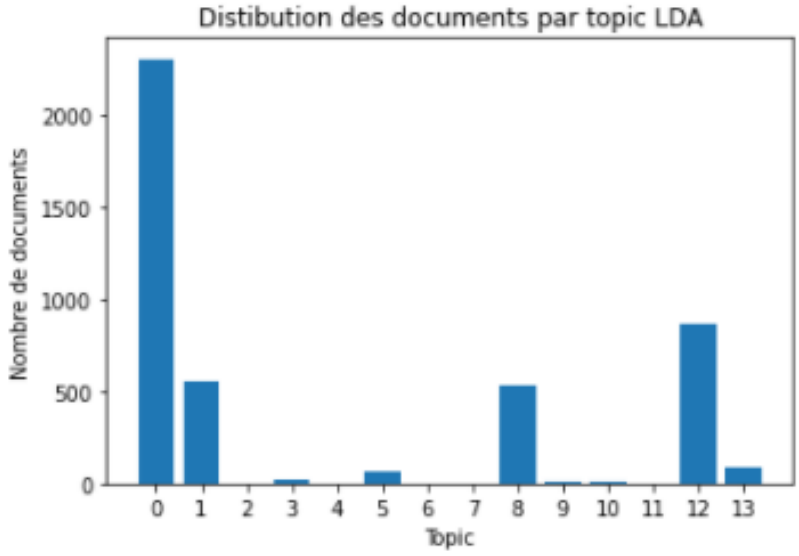
	Accuracy Top 1	Accuracy Top 2	Accuracy Top 3
MultinomialNB	56%	76%	84%
LogisticRegression	54%	73%	82%
SVM	55%	74%	83%

ii) Apprentissage non supervisé :

Nous allons tester dans cette partie une approche non suppervisée pour essayer de classer nos documents.

1) LDA :

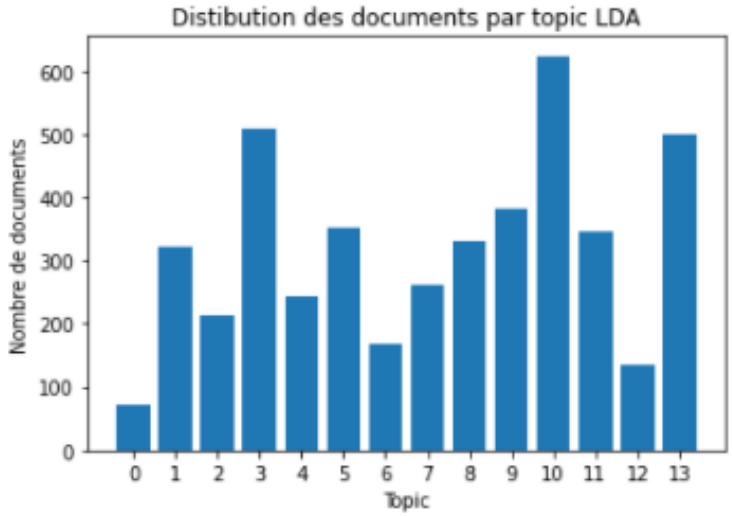
Nous allons dans un premier temps l'appliquer sur nos documents *non prétraités*.



La principale chose qu'on remarque est que la distribution des documents sur les topics est largement différente de celle de notre dataset.

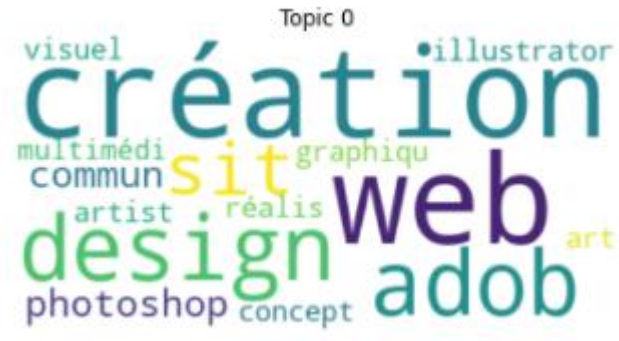
Essayons maintenant la combinaison suivante de pré-traitements :

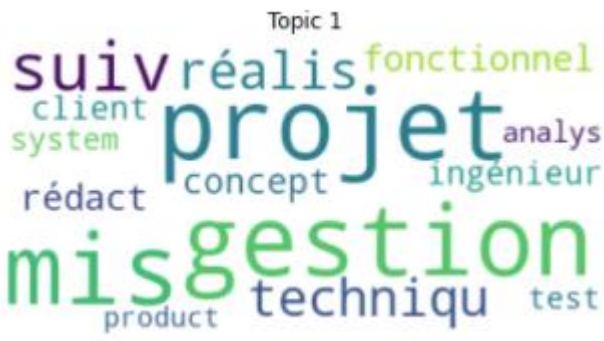
Stopwords	Stemming	Lowercase	Token-pattern	N-gram	Min_df	Max_df
True	True	True	True	(1,1)	5	0.95

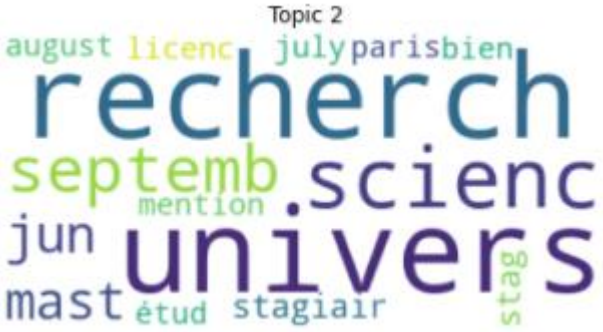


On remarque cette fois ci une distribution plus proche de celle de notre dataset.

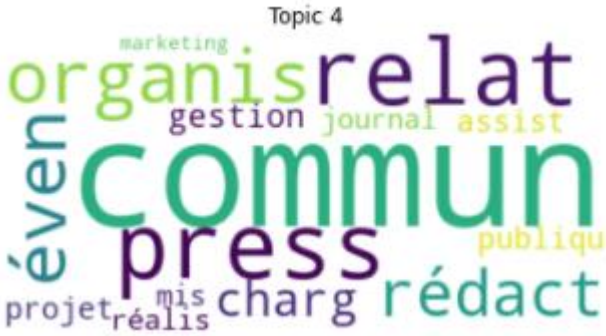
Affichons les 15 mots les plus discriminents par topic avec des wordclouds.






Topic 2
august licenc july parisbien
recherch
septemb scienc
mention
jun univers
mast étud stagiair
stag


Topic 3
ressourc social humain
entretien mis
gestion
suiv recrut manag
rh format
charg
administr projet
personnel


Topic 4
marketing
organis relat
gestion journal assist
éven commun
publiqu
press
projet mis charg rédact
réalis


Topic 5
manag financ
financi banqu
directeur
commercial direct
client
gestion
respons projet group
analys
business servic


Topic 6
produit stag
réalis
product ingénieur
analys
mission mis qualit
recherch
projet
mécan
entrepris
développ étud


Topic 7
sécur ingénieur
qualit
respons
industriel
manag amélior
servic
mainten
gestion
réalis
mis projet product septemb


Topic 8
client digital commun
social développ gestion
strategy
marketing
projet manag
medi
franc strateg business
international


Topic 9
campagn
création web
mis gestion
plac
commun
suiv
digital strateg sit
marketing
client projet social

Analyse des wordclouds :

En visualisant les différents wordclouds, on peut faire le rapprochement entre les topics et les domaines industriels suivants :

- Topic 2 -> Research
- Topic 3 -> Human Ressources
- Topic 5 -> Banking

Mais pour les autres topics, il est difficile de faire le rapprochement avec des domaines industriels car comme vu précédemment, certains domaines possèdent un jargon similaire ce qui rend la tâche compliquée.

On peut donc conclure que l'approche non supervisé n'apporte pas de solution à notre problème de confusion.

Conclusion :

Suite à cette partie expérimentale, nous pouvons déduire que le meilleur modèle pour répondre à notre problématique est le multinomial naive bayes, mais les performances obtenues restent assez faibles et ce à cause de la confusion entre les différentes classes causée par le jargon utilisé dans nos CV, pour cela on préfèrera donc l'utilisation de la métrique top-n qui nous permet de mieux cerner cette confusion.

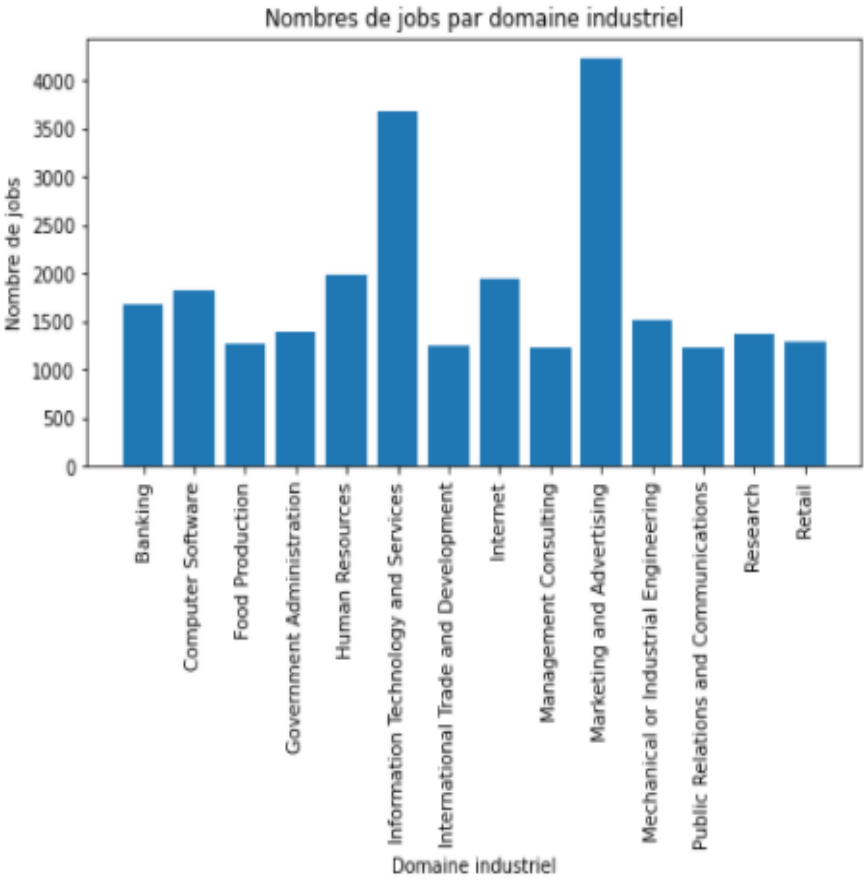
III) Partie II :

Le but de cette partie est de modéliser l'évolution de carrière, et ainsi déduire un ensemble de profiles représentant cette évolution. Pour ce faire nous avons à disposition dans les différents CV des jobs avec leurs descriptions qui nous permettront de répondre à cette problématique.

La principale difficulté de cette partie est le fait que l'on ne possède pas d'étiquettes pour nos jobs, nous devrons donc trouver un moyen de faire face à ce problème.

1) Prise en main des données :

Nous avons un total de 25813 jobs dans nos données, affichons leurs distributions par domaine industriel.



Remarque :

On peut remarquer que la distribution des jobs suit celle de nos domaines industriels.

2) **Méthodologie :**

Pour modéliser une évolution nous devons introduire la notion d'expérience dans nos données, nous supposerons donc que nos jobs peuvent être classés en trois niveaux d'expertises : débutant, intermédiaire et avancé.

Nous savons aussi que dans une carrière professionnelle une propriété est vérifiée, il s'agit de la continuité de l'évolution, ainsi un individu ne peut qu'avancer dans l'expérience acquise (pas de passage d'un niveau supérieur à un niveau inférieur) et s'il commence avec un job débutant et atteint un job expert, il est obligé de passer par le niveau intermédiaire.

Une des façons trouvée lors de nos recherches pour répondre à cette tâche est l'utilisation des chaînes de markov cachées (HMM), en effet si l'on considère que nos observation sont les jobs et que les états cachés les niveaux d'expériences, ces dernières permettent d'avoir une matrice de transition entre les états, ce qui nous permet donc de facilement vérifié la propriété de la continuité de l'évolution en mettant à zéro les probabilités de passage d'un niveau supérieur à un niveau inférieur et celle de débutant vers avancé.

Nous allons opter pour une solution plus simple car le but des HMM est de faire respecter cette propriété or nous pouvons le faire autrement.

Sachant que dans un CV les jobs suivent un ordre chronologique (le premier étant le plus récent), nous pouvons proposer un étiquetage naïf qui consiste à diviser les différents jobs d'un CV en trois groupes de plus ou moins même taille, celui contenant les plus récents se voit attribuer le niveau « avancé », les plus anciens « débutant » et le reste le niveau « intermédiaire ».

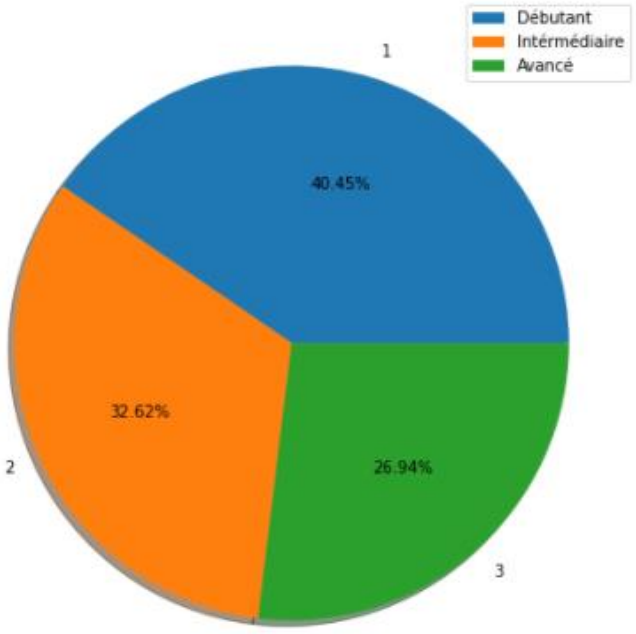
Une fois l'étiquetage terminé, nous pouvons approcher notre tâche avec un apprentissage supervisé de nos données, pour cela nous allons utiliser le meilleur modèle trouvé lors de la première partie avec les pré-traitements optimaux associés sur un dataset qui contient cette fois un job par ligne et comme dimensions notre sac de mots, nous avons fais ce choix car comme précisé ci-dessus, notre but est la modélisation et non le bonne classification des jobs, donc le modèle choisis n'apporte pas de réelle différence.

Pour faire respecter la continuité d'évolution d'une carrière dans notre cas, nous appliquerons un lissage sur les prédictions de notre modèle.

Pour le lissage, nous avons une fois encore adopté une stratégie simple, il suffira de remplacer les éléments qui causent problème par un élément capable de faire respecter la propriété, nous choisirons ces derniers de façon heuristique.

3) Campagne d'expérimentation :

Nous allons afficher ci-dessous la distribution des niveaux d'expérience dans nos jobs après l'étiquetage naïf, rappelons que nous avons à disposition 25813 jobs.



Affichons les différents word cloud des niveaux d'expertise :

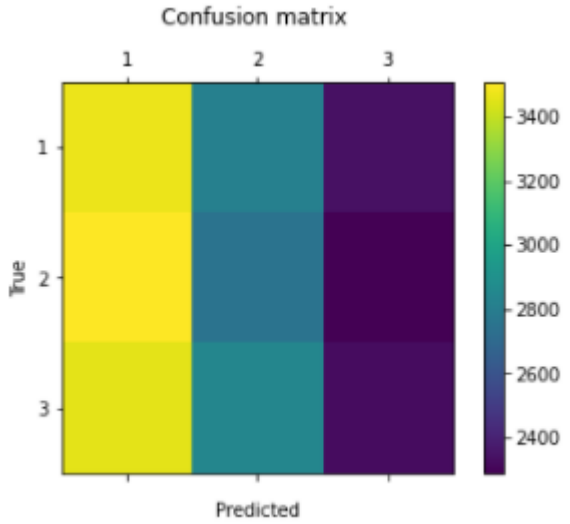
Remarque :

On peut remarquer que les mots les plus fréquents dans les trois classes sont quasiment les mêmes, on peut donc d'ores et déjà supposer que notre modèle rencontrera des difficultés pour faire la distinction entre les niveaux.

i) **Baseline** :

Prenons une baseline qui prédit les différentes classes selon une probabilité uniforme :

On obtient une accuracy de : 33.6 %



1 -> Niveau débutant

2 -> Niveau intermédiaire

3 -> Niveau avancé

ii) **Multinomial naive bayes :**

Rappelons que le meilleur paramètre alpha trouvé est 1 et que la combinaison optimale de pré-traitements obtenu est :

- Utilisation de la matrice des occurrences.
- Suppression des stopwrods.
- Suprresion de la ponctuation, des chiffres et des mots en caractères non latin.
- Mise en minuscule des mots de notre corpus.
- Suppression des mots avec un DF ≤ 5 et DF ≥ 0.9
- Bow sous forme d'uni-gramme

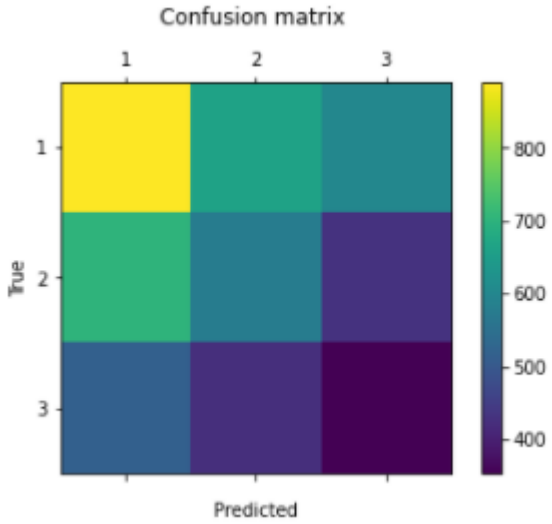
Stopwords	Stemming	LowerCase	Token-pattern	n-gram	min_df	max_df
True	False	True	True	(1,1)	5	0.9

Après ces pré-traitements, nous avons une matrice sparse avec :

- 25813 lignes (jobs)
- 10471 mots (taille du Bow)

Résultats :

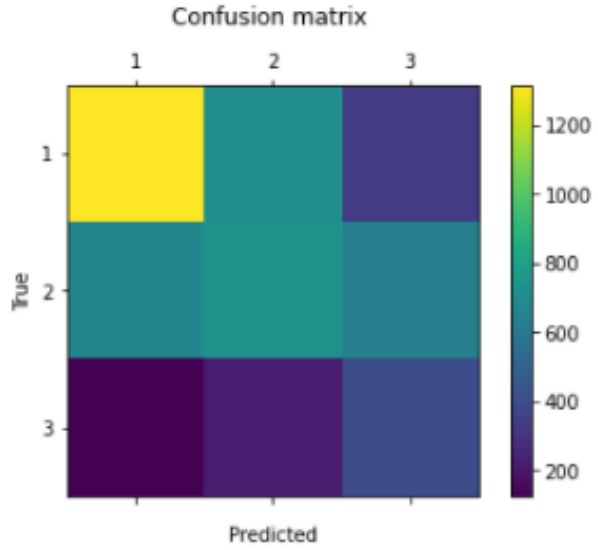
L'accuracy sur les données test de notre modèle est de : 35.25%



	precision	recall	f1-score	support
1	0.41	0.42	0.42	2111
2	0.34	0.35	0.34	1667
3	0.27	0.25	0.26	1387
accuracy			0.35	5165
macro avg	0.34	0.34	0.34	5165
weighted avg	0.35	0.35	0.35	5165

Nous allons maintenant procéder au lissage des prédictions, on obtient les performances suivantes :

L'accuracy est maintenant de : 47.60 %



	precision	recall	f1-score	support
1	0.56	0.62	0.59	2111
2	0.36	0.44	0.40	1667
3	0.54	0.29	0.38	1387
accuracy			0.48	5165
macro avg	0.49	0.45	0.46	5165
weighted avg	0.49	0.48	0.47	5165

Remarque :

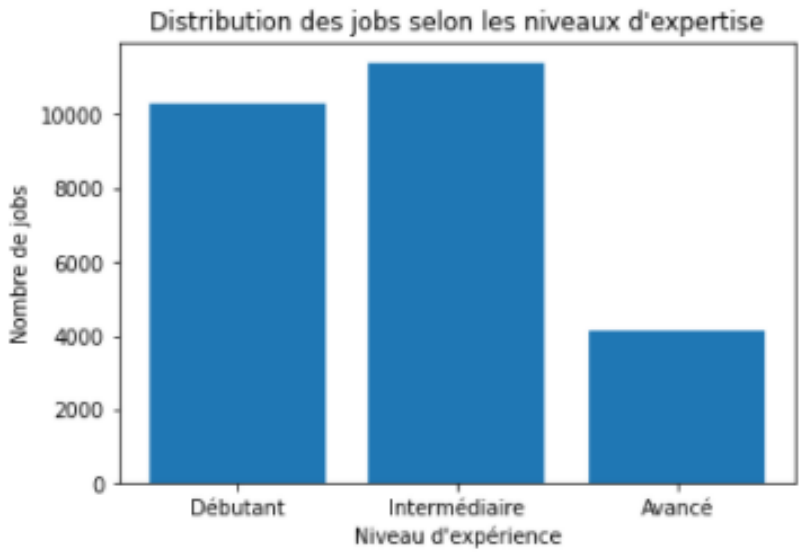
Les performances restent faibles malgré le lissage, mais comme notre objectif dans cette seconde partie est de modéliser les différents profils d'évolutions de carrière possible et non la classification des jobs, nous pouvons nous permettre de réapprendre notre modèle sur ses prédictions **lissées**, ceci aura pour effet de provoquer du surapprentissage et d'augmenter les scores, nous pouvons ainsi réitérer cette opération jusqu'à atteindre des scores jugés suffisant et enfin modéliser les profils, rappelons ici que l'objectif primaire de cette méthode n'est pas d'augmenter les scores du modèle mais plutôt d'obtenir la convergence de ce dernier.

Nous obtenons les résultats suivant avec la méthode de réapprentissage :

	Itération 1	Itération 2	Itération 3	Itération 4
Accuracy Train	76%	79%	85%	89%
Accuracy Test	62%	71%	83%	82%

On constate qu'on a bien les résultats attendus, un surapprentissage et amélioration des performances, on a choisi de s'arrêter à 4 itérations car nous considérons les scores suffisants.

Affichons maintenant la nouvelle distribution des jobs selon les niveaux (on considère maintenant les labels comme étant ceux prédit par le modèle ci-dessus à la 4ème itération après lissage)



4) Modélisation des profils :

Maintenant que nous avons notre modèle ainsi que nos étiquettes, nous pouvons entamer la tâche de modélisation des profils.

Pour cela nous allons nous intéresser dans un premier temps à la détection de toutes les évolutions possibles dans nos données, on obtient ainsi six profils :

- Profil débutant (tous les jobs sont de niveau débutant)
- Profil intermédiaire (tous les jobs sont de niveau intermédiaire)
- Profil avancé (tous les jobs sont de niveau avancé)
- Profil débutant/intermédiaire (l'individu commence en tant que débutant et évolue vers intermédiaire)
- Profil intermédiaire/avancé (l'individu commence en tant qu'intermédiaire et évolue vers avancé)
- Profil débutant/intermédiaire/avancé (l'individu commence avec un job débutant et atteint un job avancé en passant par le niveau intermédiaire)

On remarque que certains profils progressent au cours de leurs carrière tandis que d'autre non, on peut donc les classer en deux groupes :

- Profil sans évolution
- Profil avec évolution

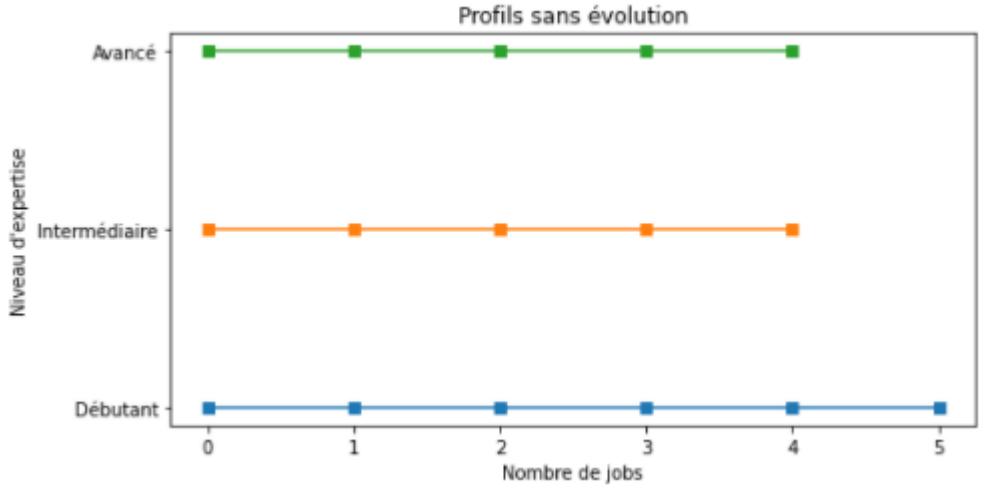
Pour les profils avec évolution nous pouvons introduire la notion de *« vitesse d'évolution »* qui définit la vitesse à laquelle évolue un individu durant sa carrière.

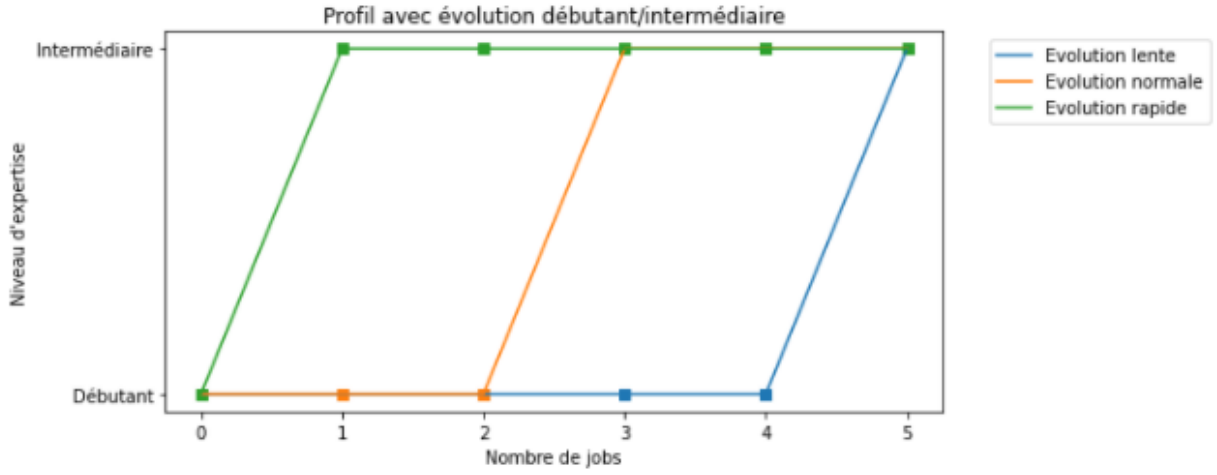
On supposera que nos profils avec évolution peuvent être à leurs tour classé en trois catégories :

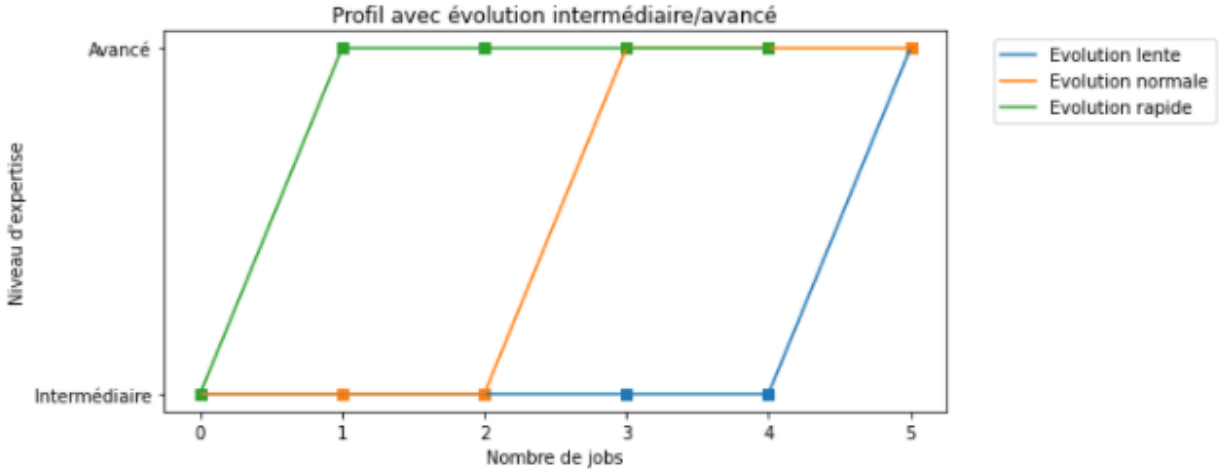
- Evolution lente
- Evolution normal
- Evolution rapide

Pour déterminer cette vitesse d'évolution, nous allons tout d'abord regrouper les individus présentant le même profil et les ordonner selon leurs vitesses d'évolution (le nombre de jobs nécessaire pour passer aux niveaux supérieurs), il suffira après de diviser ces groupes en trois sous-groupes de plus ou moins même taille. On prendra ensuite la moyenne de chaque sous-groupe comme représentant de la vitesse d 'évolution correspondante.

Nous pouvons maintenant résumer ces différents profils avec les figures ci-dessous :







Distribution des profils :

0 -> Profil débutant

1 -> Profil débutant/intermédiaire

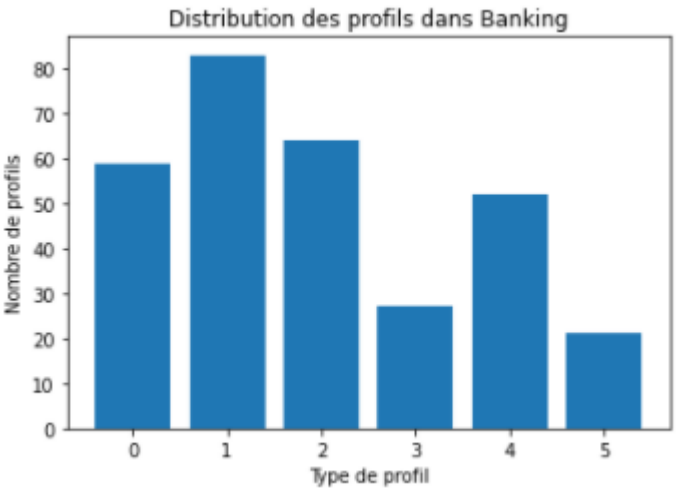
2 -> Profil intermédiaire

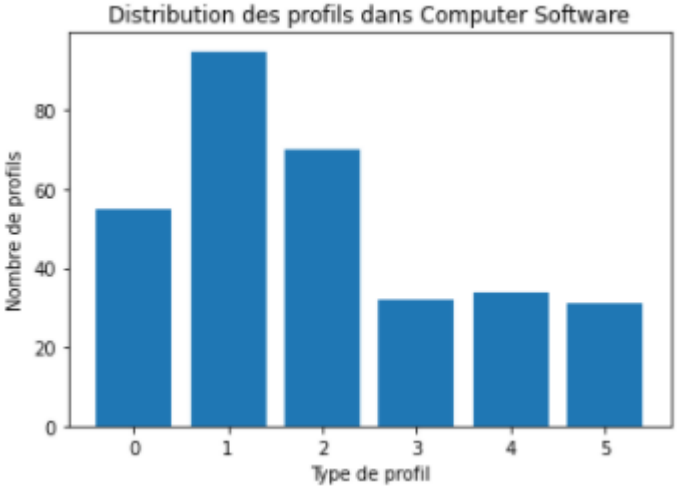
3 -> Profil débutant/intermédiaire/avancé

4 -> Profil intermédiaire/avancé

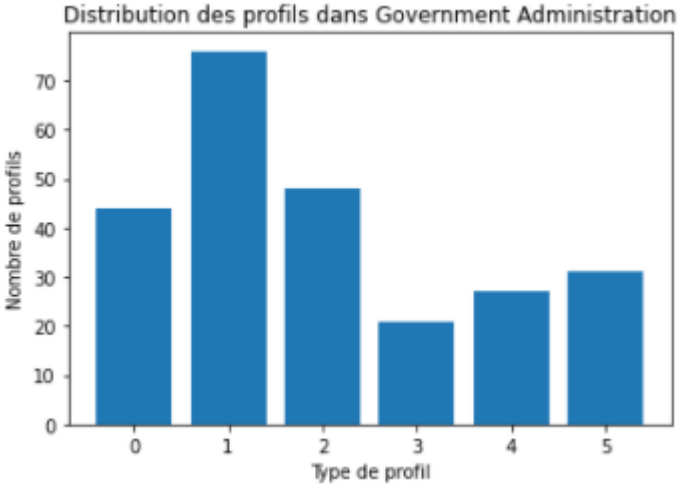
5-> Profil avancé

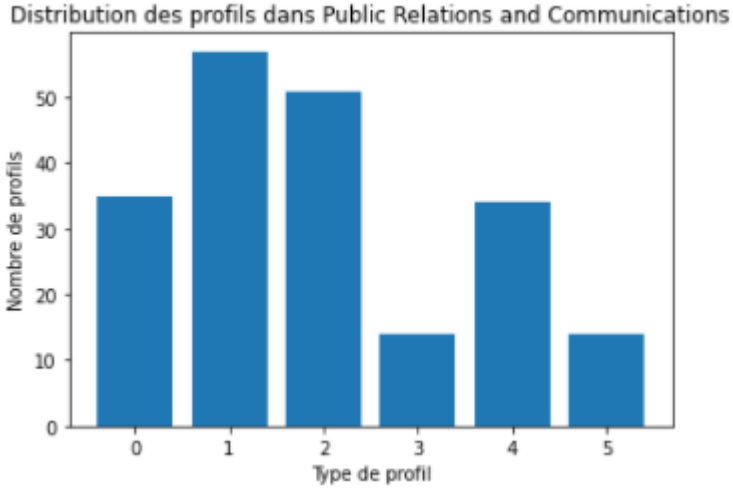
Un des aspects intéressants serait de voir si des profils sont plus présents dans certains domaines industriels que dans d'autres et pour vérifier cela, affichons la distribution des profils selon le domaine industriel :

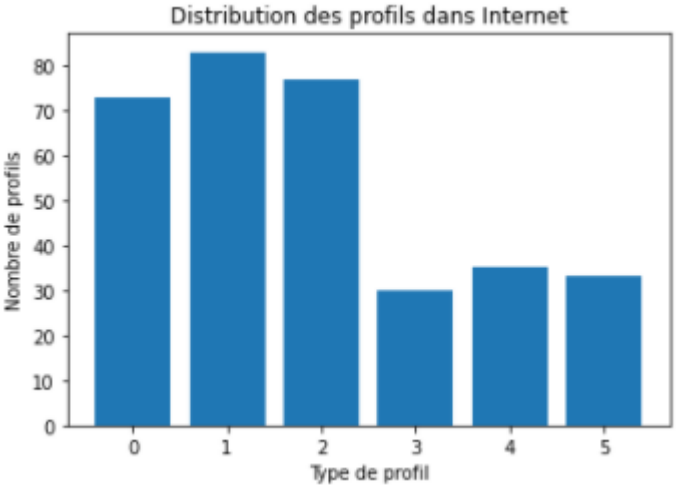

Distribution des profils dans Banking
Nombre de profils
Type de profil

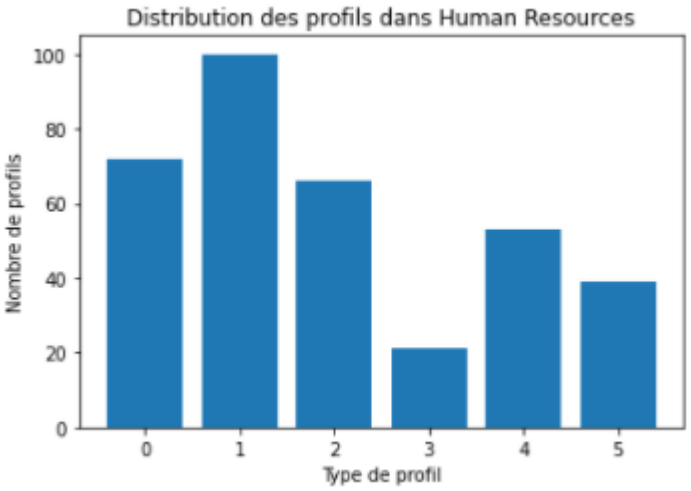

Distribution des profils dans Computer Software
Nombre de profils
Type de profil


Distribution des profils dans Food Production
Nombre de profils
Type de profil

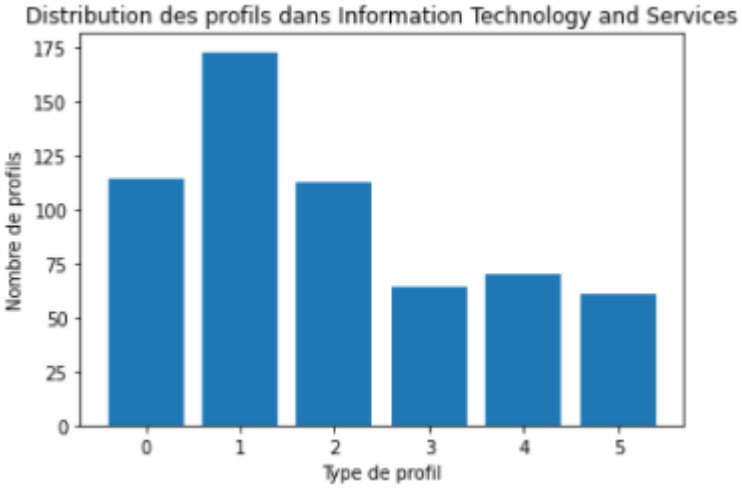

Distribution des profils dans Government Administration
Nombre de profils
Type de profil

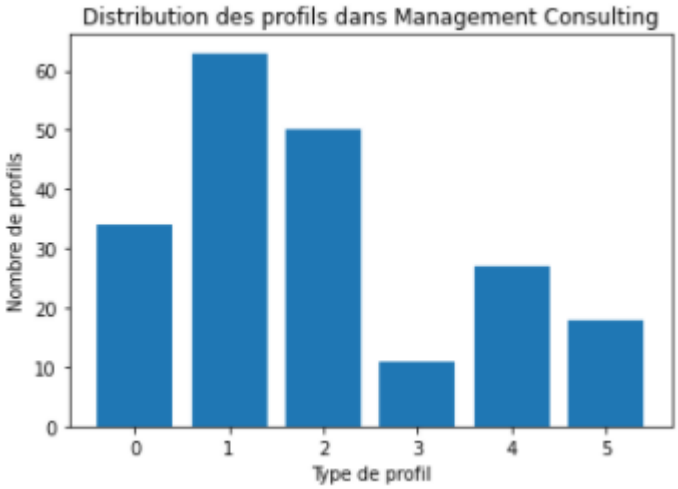

Distribution des profils dans Public Relations and Communications
Nombre de profils
Type de profil

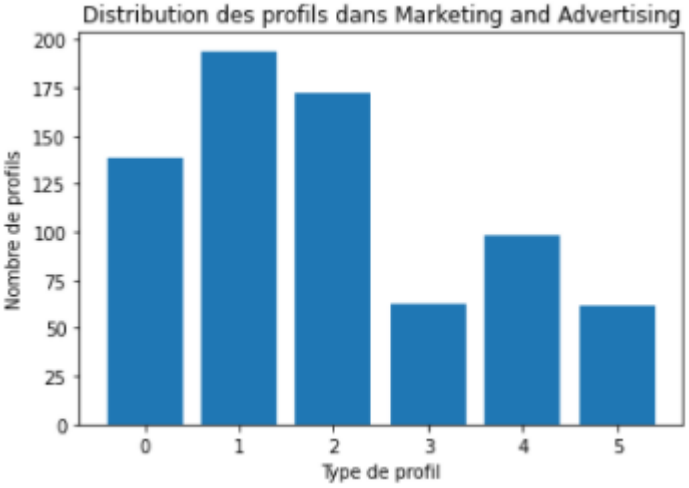

Distribution des profils dans Internet
Nombre de profils
Type de profil

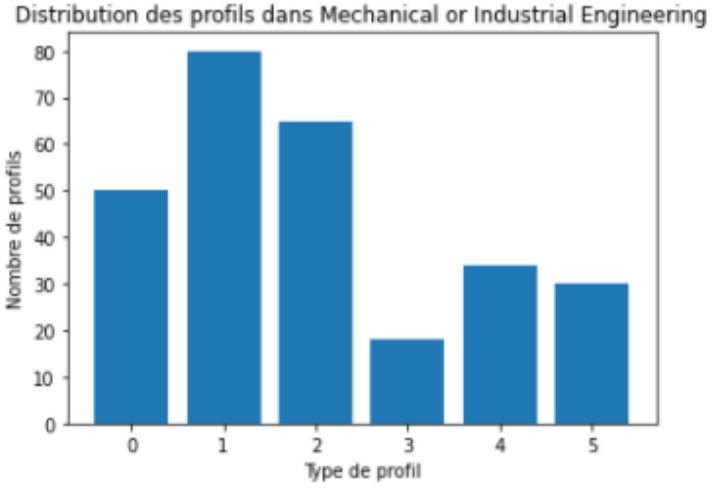

Distribution des profils dans Human Resources
Nombre de profils
Type de profil
0
20
40
60
80
100
0
1
2
3
4
5

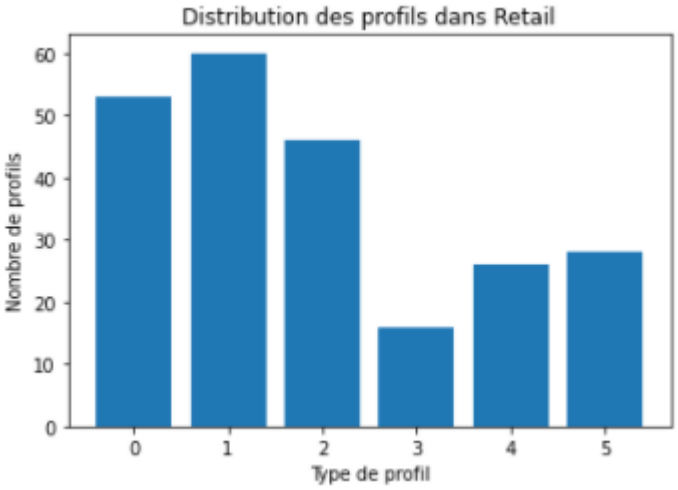

Distribution des profils dans International Trade and Development
Nombre de profils
Type de profil
0
10
20
30
40
50
60
70
0
1
2
3
4
5

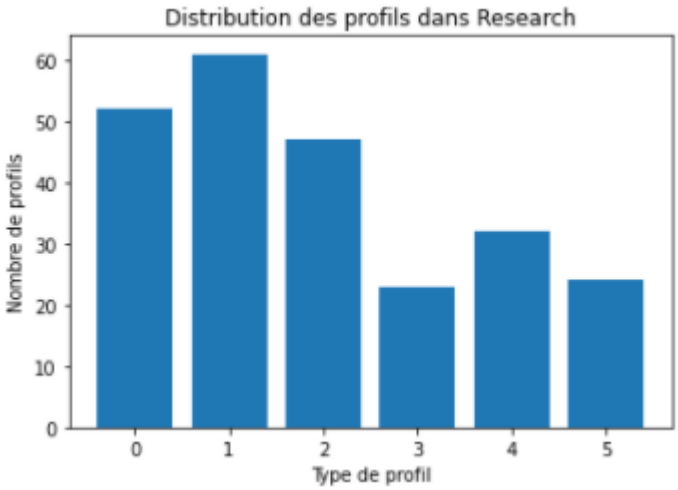

Distribution des profils dans Information Technology and Services
Nombre de profils
Type de profil
0
25
50
75
100
125
150
175
0
1
2
3
4
5


Distribution des profils dans Management Consulting
Nombre de profils
Type de profil
0
10
20
30
40
50
60
0
1
2
3
4
5


Distribution des profils dans Marketing and Advertising
Nombre de profils
Type de profil
0
25
50
75
100
125
150
175
200
0
1
2
3
4
5


Distribution des profils dans Mechanical or Industrial Engineering
Nombre de profils
Type de profil
0
10
20
30
40
50
60
70
80
0
1
2
3
4
5





Remarque :

On remarque que la distribution des profils selon le domaine industriel est assez proche de la distribution générale, on ne peut donc pas déduire de distinctions concernant cette partie.

IV) <u>Conclusion :</u>

Durant la première partie concernant la classification de CV selon leurs domaines industriels, nous avons pu remarquer que nos données présentaient un sérieux problème qui est le fait que des domaines industriels distinct possède le même jargon, ce qui induit en erreur notre modèle, néanmoins nous avons proposé un modèle capable de répondre à cette tâche avec un taux de bonne classification correct selon la métrique « top-n».

Dans un second temps nous avons réussi à générer des profils types qui modélisent les différentes manières d'évolution d'un individu durant sa carrière professionnel, nous pourront ainsi par la suite prédire la vitesse et la manière d'évolution de ce dernier.

La combinaison des deux parties précédentes nous permettra donc de différencier les CV entre eux d'une manière plus efficace et nous ouvre toute une panoplie de possibilités d'applications.