Projet d'Industrialisation

Charlotte Dugrain – Lucie Miskic

Université Paris Diderot

Année universitaire 2019-2020

I. Le projet

A. Rappel du sujet

Le projet repose sur une tâche de découpage automatique d'offres d'emploi.
Ces annonces sont visiblement tirées d'internet puisqu'il reste des traces de html à l'intérieur.
L'annonce doit être découpée en quatre catégories distinctes :

- **id :** identifiant de l'offre
- **offerTitle :** l'intitulé de l'offre d'emploi
- **companyDesc** : description de l'entreprise
- **jobDesc :** description du poste
- **profileDesc** : description du profil

Il est important de noter que les offres sont censées être ordonnées toujours de cette façon, chaque offre doit être découpée dans l'ordre précis donné ci-dessus.

B. Données à disposition

Offers.csv : un ensemble qui sert de corpus d'entrainement d'offres correctement découpées, identifiées avec un index et séparées en différentes parties par des virgules. Au total ce fichier contient 50 000 offres.
Input.csv : un ensemble qui servira de test d'offres avec juste l'index et l'offre (10 au total)
Output.csv : la sortie attendue après le passage de notre modèle pour l'ensemble de test, (le découpage et la mise en forme requise).

II. Traitement du sujet

A. Analyse des données fournies

Avant de se lancer dans le projet, il nous a semblé crucial d'observer les données au préalable afin de voir quelle approche conviendrait le mieux.

Nous avons donc fait plusieurs scripts qui analysaient les données et nous sortaient quelques statistiques dessus. Nous avons fait de l'exploration pour pouvoir savoir par quel angle attaquer le problème.

Nous avons donc remarqué un certain nombre de points :

Les données de offers.csv sont très hétérogènes dans leur mise en forme.

Récupérées massivement, ces offres, bien qu'elles semblent avoir le même ordre des différentes parties que nous devons découper, ont une mise en forme très variable.

Il reste notamment énormément de traces de html :

- Les accents et autres caractères spéciaux
- Les balises de styles (liste pucées, paragraphes, centrage …)

De par leur nature et leur mise en forme en html, la ponctuation n'est pas toujours présente, ce qui nous indique que la tâche de découpage en phrases sera plus délicate que si nous avions travaillé sur des offres au format texte basique.

Certaines offres sont publiées par les mêmes entreprises, il n'est donc pas rare d'avoir des doublons au niveau de la description de l'entreprise.

1. Analyse des balises
 en tant que potentielle feature

La présence des
 (balises de retour à la ligne) semble très importante. Nous avons décidé d'explorer un peu plus en détails afin de savoir si elles apparaissaient de façon équilibrée et si elles pouvaient avoir une utilité pour différencier les différentes parties.

Après avoir analysé leur distribution parmi les différentes parties de toutes les offres, il se trouve finalement que bien que ces balises soient extrêmement nombreuses, elles ne soient pas réparties uniformément parmi les offres pour pouvoir nous servir en tant qu'indice d'appartenance à une partie plus qu'une autre.

	Index offre	**OfferTitle**	**CompanyDesc**	**JobDesc**	**ProfileDesc**
# par partie de l'offre	0	0	35 536	285 673	137 105
**# offre sans **	46 025	46 025	33 004	7 790	11 282
% sans selon partie de l'offre			71,70 %	16,92 %	24,51 %

La première ligne du tableau nombre le nombre de balises
 recensées au total pour toutes les offres dans la partie correspondantes.

A partir de ces premiers chiffres, nous avions bon espoir d'avoir peut être quelque chose de significatif, en effet il semblait y avoir de très nettes différences de nombre d'occurrences selon les parties qui nous intéressaient (CompanyDesc, JobDesc et ProfileDesc) avec JobDesc qui comprenait en tête le plus d'occurrences de cette balise, ProfileDesc en deuxième position. Ces résultats faisaient logiquement sens pour nous dans la mesure où nous pouvions facilement nous imaginer qu'une description d'emploi et de profil serait plus à même d'avoir une mise en forme de liste (avec bon beaucoup de retours à la ligne).

Cependant l'important était de savoir si ce critère était réellement un indice pour définir des propriétés générales selon la partie.

Pour cela nous avons simplement décidé de compter pour chaque partie, le nombre de partie dans lequel la balise n'apparaissait pas du tout. Il s'agit des résultats dans la deuxième ligne du tableau.

Après un rapide calcul, il se trouve qu'un pourcentage non négligeable des offres ne contiennent aucune balise de retour à la ligne, ce qui nous indique qu'il ne s'agit pas d'un facteur aussi différenciant que nous le pensions, il est donc inutilisable en tant que feature.

2. Analyse du vocabulaire des offres

En revanche, après lecture des données, il nous a semblé évident que le vocabulaire utilisé selon les différentes parties était suffisamment différent pour pouvoir le prendre en considération.

En toute logique, le vocabulaire selon les parties semble être bien différencié :

- La partie offerTitle contient souvent par exemple le token (h/f)

Voici par exemple un wordcloud de la partie offerTitle :

- La description de l'entreprise contient un certain vocabulaire spécifique (mission, entreprise, tâche, clients, notre entreprise, rôle, importance)
- Un vocabulaire propre aux compétences attendues de l'individu peut apparaître dans la description du profil (capacité, compétence, aptitude, gestion, gérer, qualités, relations…)

Ceci nous a donc orienté vers une représentation des données en Bag Of Words.

Etant donné l'importance des termes dans les parties et la différence de vocabulaire notable entre elles, nous avons également choisi d'implémenter notre Bag of Words avec une mesure en TFxIDF plutôt qu'en nombre d'occurrences classique, puisque le

TFxIDF met en valeur l'importance d'un terme par rapport à l'ensemble des documents.

3. Analyse des longueurs des parties d'offres

Il semblait logique de regarder la longueur moyenne en terme de phrase sur chaque partie d'offre. C'est donc ce que nous avons fait en utilisant un tokenizer de nltk.

	companyDesc	**jobDesc**	**profileDesc**
Moyenne	1.682851	3.267292	2.892466
Mediane	1	3	3
Min	1	1	1
Max	14	55	23

Voici le tableau avec les longueurs moyennes, médianes les minimums et les maximums pour chaque partie.

La longueur moyenne et médiane semble effectivement différencier au moins la partie companyDesc des deux autres !

Il serait donc possible d'utiliser ces données dans un modèle où l'on pourrait les intégrer.

B. Les questions qui se sont posées

Voici les principaux questionnements que nous avons eus, et les problèmes majeurs auxquels nous avons dû faire face pendant le projet.

1. Découpage des offres

Pour pouvoir utiliser le TFxIDF, nous avons donc choisi de découper les offres en différentes parties, et de simplement considérer chaque partie comme un document à part entière.

2. Choix du modèle

Ce choix nous a orienté simplement vers le modèle que nous allions implémenter.

Nous hésitions initialement entre un modèle simple de type KNN et un modèle de classification linéaire. En choisissant de découper les documents en partie et de les considérer comme des documents à part entière, et en choisissant la représentation en Bag Of Words, il semblait compliqué de réaliser un KNN.

En effet, la nature des objets à classer aurait été différente des objets de référence : les objets de référence étaient des paragraphes (les parties déjà séparées d'après le document de référence), mais nous cherchions à délimiter ces paragraphes, donc ce que nous aurions fourni à prédire au KNN était sous la forme de phrase et non de paragraphes. Le KNN nous semblait donc compromis.

Il apparaissait donc plus naturel de traiter le problème comme un simple problème d'appartenance à une classe avec un classifieur linéaire multiclasse. Chaque partie de l'offre est une classe, nous fournissons une phrase au classifieur et il se charge de nous estimer l'appartenance de la phrase à la classe en question. Les classes étant donc : offerTitle, companyDesc, jobDesc, profileDesc Le problème du KNN est évité alors grâce au vecteur de paramètre qui comble lors de l'entrainement le gap entre la représentation de phrase et la représentation en paragraphe.

3. Découpage du test

Il avait été initialement abordé l'idée d'un curseur dans l'implémentation de la solution. C'est-à-dire qu'il aurait été intéressant de regrouper les phrases entre elles linéairement, et de créer artificiellement des parties d'offres et d'évaluer directement ces paragraphes pour en retirer les plus pertinents.

Bien que cette approche avait l'air intéressante, elle nous semblait complexe et avec un potentiel combinatoire éventuellement compliqué à gérer.

Nous avons donc simplement souhaité rester dans une logique simple avec le classifieur multiclasse, nous avons donc choisi de découper le fichier test en phrases.

Cependant cette tâche n'a pas été des plus évidentes étant donné que les offres des fichiers test ne contenait presque aucune ponctuation (à cause de la mise en page HTML qui à un objectif de mise en page graphique). - Nous avons donc décidé de sectionner sur les retours à la ligne, puisqu'en toute logique un retour à la ligne dans une représentation graphique représente effectivement une séparation logique entre des parties de texte.

4. Choix du type de classifieur

Le choix d'un classifieur multiclasse de type Naïve Bayes repose principalement sur le fait qu'il s'agit d'un modèle simple à implémenter, efficace lorsque les données ne sont pas nombreuses, mais également sur le fait qu'il s'agit d'un classifieur avec lequel nous avions de l'expérience d'implémentation. - Il est relativement simple et fournit de bons résultats sur ce type de tâche.

C. Notre implémentation

Le modèle utilisé est donc un classifieur bayesien naïf implémenté par la librairie sklearn.

1. Nettoyage des données

Nous nous sommes rendu comptes qu'une grande partie des offres n'étaient pas découpables correctement dans le fichier offers.csv.

Ce fichier devait comporter 50 000 offres, avec une offre par ligne. Or avec une analyse rapide, le fichier contenait 94067 lignes. Si nous prenons les offres qui se découpent correctement sur les critères suivant :

- Un split sur « , » est possible
- La longueur du résultat du split est égal à 5 (le numéro de l'offre, et les quatre parties)

Nous obtenons un total de 46 032 lignes (et donc offres) analysables.

Nous avons fait le choix de faire un script qui recrée un fichier qui nous servira de base pour l'ensemble d'entraînement ne contenant que les offres correctement analysables et qui gère également les symboles spéciaux de HTML qui seraient restés.

Finalement, après avoir lancé le modèle, il se trouve que l'importance de sortir les caractères spéciaux HTML ne présente pas un grand intérêt (et pose même problème lors de la reconstruction de l'offre à la fin du processus).

Concernant les fichiers input et output, nous avons décidé d'ignorer l'offre 4 qui n'était pas bien formée.

2. Entrainement

La représentation vectorielle est donc de type Bag of Words avec comme données le TFxIDF.

L'ensemble d'entraînement a donc été découpé par parties.

Le gold du train a été reconstitué manuellement à partir des listes de données de chaque classe.

L'ensemble de test lui est découpé par phrases (inférées à partir des balises
).

L'algorithme doit donc simplement apprendre un vecteur de paramètres qui va permettre de classifier la phrase donnée dans la bonne classe.

A la sortie de l'algorithme, il reste donc à reconstruire l'offre avec la bonne segmentation et ajouter de nouveau les
 sur lequels nous avions fait le découpage afin de comparer avec le fichier cible fourni.

3. Résultats

Faire une analyse de la performance de notre classifieur sur sa performance basique d'attribuer une classe correcte pour la phrase en entrée était relativement simple.

En revanche cela ne capturait pas la précision par rapport au bon découpage de l'offre. C'est pour cela qu'il est nécessaire de reconstruire l'offre et d'évaluer notre modèle sur l'offre complète une fois qu'elle a été reconstruite, pour pouvoir avoir une idée de la performance réelle sur les données.

Pour ce qui est de la performance phrase par phrase, sur le fichier input que nous découpons en 111 morceaux/phrases nous obtenons une précision de 91% si l'on entraîne et teste sur les fichiers en gardant les caractères spécifiques au html, et de 92% si on les retire à l'aide la méthode html.unescape(). Ces résultats semblent satisfaisants.

Pour ce qui est de la précision du découpage en tant que telle, elle descend fortement en effet, si par exemple même un seul des 25 morceaux de la phrase est mal prédit toute l'annonce est considérée comme mal découpée. Nous obtenons une précisions de 55% (44% si l'on ne corrigeait pas le fait que la dernière partie est forcément profileDesc).

III. Evaluation de notre modèle et conclusion

A. Avantages de notre approche / modèle :

Simple et efficace (pas de réseaux de neurones).
Fonctionne bien sur peu de données d'entrainement.

B. Limites de notre approche / modèle :

Un minimum de pré-traitement et/ou nettoyage est nécessaire, on ne peut pas balancer les données dans un modèle telles quelles nous sont présentées.

Notre modèle attribue une partie à une phrase, mais ne fait pas le découpage à proprement parler (en tout cas pas en une seule phase). Il s'agit d'un classifieur multiclasse et d'une reconstruction par la suite.

De ce fait, bien que notre prédiction phrase par phrase/morceau par morceau est relativement satisfaisante, elle l'est moins après reconstruction puisqu'une seule phrase mal placée rend tout le découpage de l'annonce faux.

En effet le fait de procéder de la sorte ne permet pas de potentiellement corriger le découpage par simple décalage d'une frontière entre 2 parties, on peut se retrouver avec des phrases au milieu d'une partie auxquelles elles n'appartiennent pas, ce qui n'est pas très satisfaisant.

La méthode nous permettant de récupérer les gold dans le cas du test, donc phrase par phrase serait peut être à revoir, en effet sur un plus grand nombre de données on risquerait d'obtenir plusieurs gold pour une seule phrase car la phrase pourrait apparaitre dans plusieurs parties du fichier, notamment pour les morceaux très courts (du fait du découpage sur les sauts de lignes notamment).

C. Pour aller plus loin

Nous avions regardé la longueur moyenne et médiane des différentes parties comptées en nombre de phrase. Ce paramètre aurait pu être intéressant à prendre en compte si nous avions développé un modèle avec curseur.

Nous aurions également pu analyser l'importance des stop-words. Effectivement la liste que nous avons est une liste pré-faite, parmi laquelle il y a des déterminants possessifs comme « notre », « votre », … qui pourraient avoir leur importance selon la partie de l'offre dans laquelle ils apparaissent. Il n'est pas idiot de s'imaginer que

l'emploi de « notre » sera plus fréquent dans la partie de l'offre où l'entreprise parle d'elle, et que l'emploi de « votre », « vos », « vous », apparaît dans les parties qui concernent le candidat.

Il aurait également pu être intéressant de faire une analyse des offres en bigrammes, cela aurait peut être permis de capturer certaines expressions relativement figées ou des colocations fréquentes. Peut être cela aurait-il permis d'obtenir des résultats légèrement meilleurs.

Nous aurions également pu prendre en compte le nombre d'entités nommées apparaissant selon les parties, cela aurait pu être intéressant à analyser (bien qu'il est possible que ce facteur ne soit pas réellement différenciant).

Nous aurions également pu prendre en compte certaines balises html comme des indices de parties (nous pouvons imaginer qu'une liste est plus utilisée dans la description du profil attendu que dans la description de l'entre prise par exemple). Mais du fait de l'inconsistance de mise en forme des offres, cela n'aurait probablement pas été très significatif.

Il nous a semblé que le fait d'avoir seulement dix offres pour tester notre modèle, il aurait pu être intéressant d'avoir un découpage en train et test un peu plus équilibré.

Nous aurions également pu ajouter d'autres règles de correction comme celle partant du principe qu'il n'est pas possible d'avoir autre chose que profileDesc en fin d'annonce, nous aurions pu avoir le même raisonnement pour telle partie ne peut pas être suivie par celle là, bien que cette correction un peu manuelle ne soit peut être pas la manière la plus appropriée d'améliorer nos prédictions.

Nous aurions également pu tester d'autres modèles comme la clusterisation, un modèle utilisant les curseurs, ou encore des réseaux de neurones.