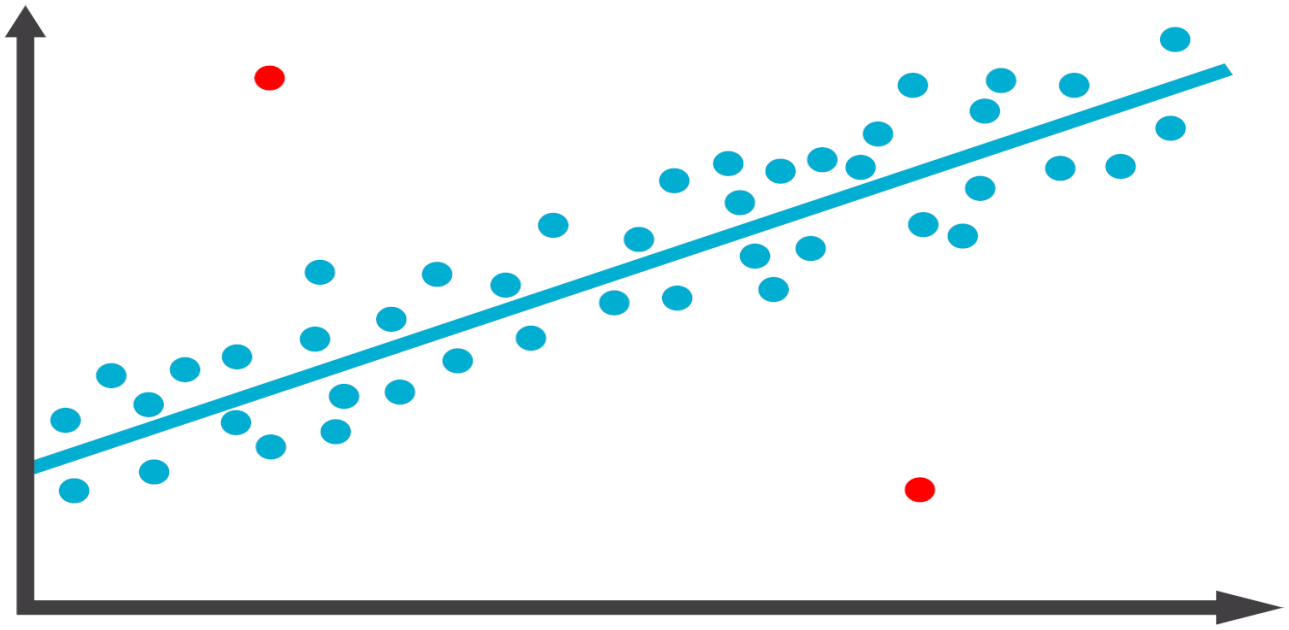
le cnam
école d'ingénieur·e·s

Rapport outlier detection

Arthur BARIBEAUD, Etienne MONTHIEUX, Mathilde VIRONDEAU, Paul REVERSAC



Sommaire

Introduction

Le but du projet était ici de déterminer si oui ou non, les fichiers à notre disposition entraient dans une catégorie de fichiers que nous connaissons.
Nos données sont sous la forme de fichiers json et ce sont des fichiers de :

- taxes d'habitations
- compromis de vente
- contrats de bail locatif
- justificatifs de domicile
- relevés de compte épargne
- bulletin de paie
- relevés de compte
- avis impôt
- avis de situation déclarative
- avis taxes foncières

Préparation des données

Le jeu de données utilisé pour entraîner nos modèles est ainsi composé de données textuelles. Il ne nous a donc pas été nécessaire d'utiliser des techniques d'OCR puisque ces dernières ont déjà été effectuées afin d'extraire les informations textuelles de données PDF avant d'être renseignées dans des formats JSON. L'un des problèmes existant avec les techniques d'OCR, est qu'en fonction de la qualité des images en entrée, les données textuelles en sortie ne seront pas forcément de bonne qualité. Notre jeu de données s'est ainsi retrouvé constitué de deux types de documents. D'une part des documents de bonne qualité et pour lesquelles les jetons sont correctement retranscrits. Et d'autre part, des documents de moins bonne qualité pour lesquels très peu de jetons sont correctement retranscrits.

```
avis situation déclaratif impôt revenir suite avis information complementaire revenir fiscal référ
ence dom rrrrrrrr information indiquer mémoire rcm déjà soumettre prélèvement social csg déductibl
e plafond epargne retraite plafond disponible déduction cotisation verser déclaration revenir sous
```

Document 1 : Bonne qualité des jetons extraits

```
bons je a et lo e nés oxouwe lee vos à le d 26 ef ga amer 2 ie jase nains de j uu mitai ds 2136s h
aut 27 us be dit 7 gicaute 4 ivres à ma pe fe houe ordi re arret ep fou hate pour fa cou bu de son
pobitahya pou di parle duvet dat pu les jabba cœur chair
```

Document 2 : Mauvaise qualité des jetons extraits

Il a ainsi été nécessaire d'effectuer une étape de nettoyage sur notre jeu de données pour tenter d'obtenir un jeu de données de la meilleure qualité qu'il soit.

On a donc normalisé dans un premier temps les documents en minuscule avant de les séparer en liste de jetons. Chaque jeton est ainsi traité à l'aide de pyspellchecker afin de corriger les éventuelles erreurs : exemples → "naisszнces" sera transformé en "naissances". Enfin, les stopwords ont également été retirés dans le but de ne garder que les mots porteurs de sens dans le jeu de données.

La gestion du bruit a par la suite été une étape importante. En effet, même si nos données ont été nettoyées, il reste encore des jetons pour lesquelles aucune signification n'existe et qui résulte souvent d'une mauvaise qualité du document d'entrée (à l'exemple des jetons du Document 2 présenté ci-dessus). Ainsi, à l'inverse du projet de détection des livrets de famille réalisé durant le premier semestre et pour lequel le "bruit" pouvait être synonyme d'écritures manuscrites, le "bruit" n'est ici aucunement intéressant pour détecter nos outliers. De ce fait, une règle très stricte a été mise en place et a permis de filtrer l'ensemble des "bruits" et ainsi de ne garder que les mots réellement porteurs de sens.

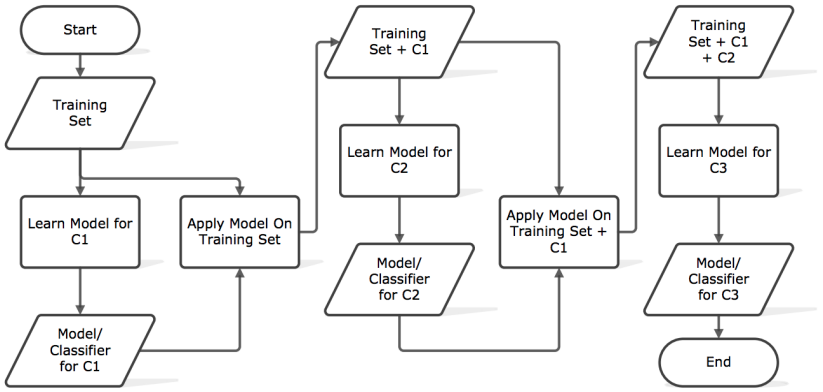
Cette étape, réalisée sur l'ensemble de notre jeu de données, composé de plus de 10 000 documents, est cependant une étape extrêmement chronophage. Elle a par conséquent été réalisée une seule fois. Les documents nettoyés ont ainsi été enregistrés dans un nouveau fichier .json prêt à l'emploi pour l'entraînement des algorithmes. (clean.csv et outlier.csv)

Approche expérimentale

Une fois les données préparées, elles ont pu être utilisées sur différentes approches destinées à détecter les documents inconnus parmi les classes de documents connus. Pour cela, différentes approches ont été testées afin d'identifier la meilleure façon de procéder. Ainsi, une première approche ensembliste a été de réduire notre problématique en une classification multilabel. Une deuxième approche a été l'utilisation d'algorithme dédié à ce genre de problématique d'outlier et novelty detection.

Multi-label

Le but du multi-label est de trouver quelle est la ou les classes dans lesquelles les données s'inscrivent. Pour y arriver nous avons utilisé un chain classifier. Le chain classifier entraîne un modèle par catégorie. Ensuite, il trie les modèles par score décroissant et il fait une prédiction pour chaque catégorie. Il est donc possible qu'une donnée d'entrée corresponde à plusieurs catégories, mais ce n'est pas le but de notre analyse. Cette méthode permet de connaitre les nouveaux documents, car ils ne seront pas prédits comme un document présent dans le modèle.



Fonctionnement du chain classifier

Pour réaliser le chain classifier nous avons re-traité les données avec 2 méthodes.
Une première approche avec un count vectorizer qui permet de représenter les mots par une matrice du nombre d'apparitions des mots par classe. La seconde approche ajoute un tfidf transformer qui centre et réduit la matrice de mots du count vectorizer. Ensuite, nous séparons le jeu de données des classes connues pour avoir 20% de données test qui n'influencent pas l'apprentissage et nous formatons les résultats attendus avec une colonne par classe pour ne pas donner une distance entre les classes.

```
avis_situation_declarative              0
avis_taxe_fonciere                      0
bulletin_de_paie                        1
compromis_de_vente                      0
contrat_bail_locatif                    0
epargne                                 0
impot                                   0
justificatif_domicile                   0
justificatif_domicile_taxe_habitation   0
releve_de_compte                        0
```

Formatage du résultat

Enfin, nous utilisons plusieurs modèles pour le chain classifier. Un SVC, un Multinomial Naïve Bayes et une Linear Regression avec un cross-validation. Chaque modèle est fait avec count vectorizer et avec le tfidf en plus. Les résultats des modèles sont décevants. Le SVC avec seulement le count vectorizer a sur-appris et détecte 32% des classes avec les données inconnues. Avec le tfidf, le SVC ne reconnait pas les données d'entrainement et de test , mais il reconnait clairement les données inconnues. Pour le Multinomial Naïve Bayes, aucun modèle n'est convaincant. Pour finir, la Linear Regression n'a pas fonctionné car le count vectorizer ne permet pas d'avoir une modélisation linéaire des données.

	train	test	RMSE	model
train_svc	0.999	0.964	0.011	linear SVC
test_svc	NaN	0.327	0.288	linear SVC
tfidf_train_svc	0.012	0.011	0.314	linear SVC
tfidf_test_svc	NaN	0.996	0.017	linear SVC
train_bayes	0.599	0.598	0.210	MultinomialNB
test_bayes	NaN	0.041	0.375	MultinomialNB
tfidf_train_bayes	0.730	0.733	0.169	MultinomialNB
tfidf_test_bayes	NaN	0.235	0.309	MultinomialNB

Résultats des modèles

Pour aller plus loin dans cette approche du chain classifier, il est possible de faire une recherche d'hyperparamètres pour le SVC ou de tester d'autres modèles comme la NLP ou le du Deep Learning pour trouver des modèles plus pertinent et comprenant mieux le contexte du document.

Outlier et Novelty detection

Scikit-learn propose des algorithmes pour la détection de valeurs aberrantes et d'anomalies. L'un d'eux est celui du *Local Outlier Detection* (*LOF*). Il y a deux approches possibles : ou bien on dispose d'une base de données sans outliers, et l'on souhaite détecter si de nouvelles données entrantes sont différentes ou inhabituelles (on parle de *Novelty*) ; ou bien on souhaite directement détecter la présence d'*outliers* dans notre base.

LOF est une méthode non supervisée de détection d'anomalies qui calcule l'écart de densité local d'un point de données par rapport à ses voisins. Il considère comme valeurs aberrantes les échantillons qui ont une densité sensiblement inférieure à celle de leurs voisins. Pour l'appliquer dans notre cas, la méthode *bag of words* a été utilisée afin de vectoriser les documents.

La configuration avec un cas de *Novelty* a été la première testée. Il était possible de séparer notre base en sets de *train* et de *test*. Cela a permis de vérifier qu'il n'y avait pas de cas de surapprentissage. La méthode LOF a ainsi été entraînée sur des bases de données vides de tout *outlier*.

LOF ne présente aucune méthode déjà pré-développée dans le package de Scikit-learn pour évaluer son efficacité. Il faut donc calculer par nous-mêmes le nombre d'outliers détectés avec la méthode, et le comparer avec le nombre réel que l'on connaît (ici, 0). En résultat, LOF considère que 22% des données de notre base *train* sont des *outliers*, et que 23% des

données de notre base test sont des *outliers*. On semble éviter le surapprentissage. Il y a une marge d'erreur avec la méthode LOF, mais ce qui nous intéresse surtout, c'est de l'appliquer à de nouvelles données pour vérifier la présence d'*outliers*.

Les données que l'on teste sont toutes considérées comme des *outliers*. Il est donc souhaité que notre méthode renvoie un taux d'omission des *outliers* le plus faible possible. Et dans notre cas, la méthode LOF s'est avérée plutôt efficace : seulement 6% des *outliers* n'étaient pas détectés.

Pour vérifier si l'on pouvait encore plus améliorer nos résultats, nous avons essayé d'utiliser la méthode TF-IDF au moment de vectoriser nos documents. La méthode permettant d'évaluer l'importance d'un terme au sein d'un document, on pouvait supposer que la méthode LOF reconnaîtrait plus facilement les *outliers*. Malheureusement, cela n'a pas été concluant. Si l'entraînement de la méthode LOF a montré une nette diminution du taux d'outliers détectés sur notre base vide de tout *outliers* (on passait de 22% à 2% d'erreur lors de l'entraînement), la détection d'*outliers* avec les nouvelles données avait considérablement chuté : le taux d'omission des *outliers* s'élevait à 78%. La méthode TF-IDF pour vectoriser les documents a donc été mise de côté.

La deuxième configuration testée n'intégrait pas le paramètre du *Novelty*. Les *outliers* ont été ajoutés à la base de données auparavant vide de tout *outlier*, et la méthode LOF a été appliquée une unique fois sur cette base.

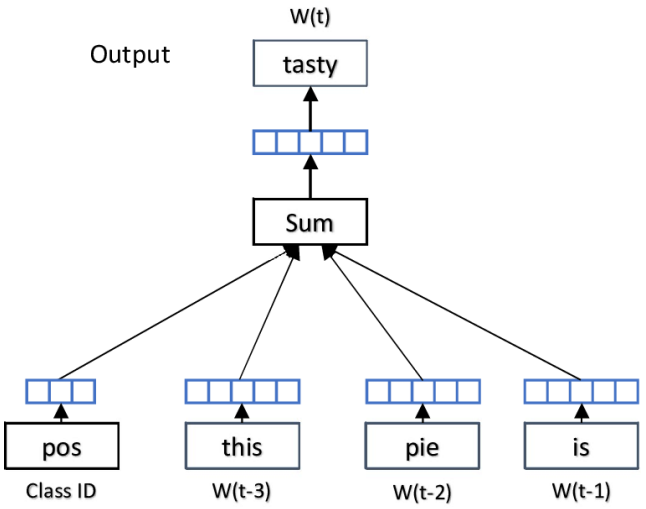
Là encore, il n'y a pas de méthode pré-établie pour estimer l'efficacité de la méthode. Il a fallu à nouveau calculer le nombre d'outliers détectés, et le comparer à celui que l'on connaissait. Les résultats ont été plutôt mitigés : alors qu'on souhaitait avoir un taux d'*outlier* proche de 12%, la méthode LOF renvoyait plus du double d'*outlier*, avec un taux à 26%. Ce n'est en soi pas si étonnant : il faut déjà prendre en compte les non-outliers pourtant évalués comme tel dans la base de départ par la méthode LOF, qui s'ajoutent donc aux vrais *outliers*. Ensuite, sans *Novelty*, on peut supposer que la méthode LOF considère les *outliers* ajoutés artificiellement à la base comme un nouveau cluster au moment de ses calculs (la densité des outliers étant proche entre eux). Les *outliers* détectés après seraient donc possiblement uniquement les valeurs extrêmes qu'il y avait déjà dans notre base vide d'outliers, auxquels s'ajoutent les valeurs aberrantes au sein même des *outliers*.

Dans notre cas, appliquer la méthode LOF sans le paramètre *Novelty* ne semble pas très efficace.

Ouverture : Dov2Vec

Une ouverture de recherche a, pour finir, été entreprise afin d'étudier la méthode de Doc2Vec. Par manque de temps et au vu des bons résultats obtenus avec les autres méthodes, cette approche n'a pas pu être développée, mais elle semble cependant intéressante à présenter.

Cette dernière consiste donc à encapsuler le contexte d'un document dans un vecteur. Il est ensuite possible de comparer plusieurs vecteurs de documents à l'aide de la similarité du cosinus. Cette méthode est initialement dérivée de word2vec utilisée comme technique d'embedding.



Cette méthode se rapproche finalement de celle "ensembliste" présentée précédemment. En effet, l'idée ici serait d'encapsuler le contexte de l'ensemble des différentes classes de documents, puis de calculer le cosinus de similarité d'un nouveau document avec les thèmes connus.

Il est ainsi probable que les résultats soient en conséquence du même ordre que ceux obtenus pour la méthode ensembliste multilabel. Par manque de temps, cette approche n'aura ainsi pas pu être mise en place. De plus, et en cas de mauvais résultats des techniques classiques d'outlier et novelty detection, elle aurait pu s'avérer intéressante à utiliser.

Conclusion

Après avoir récupéré nos documents types sous forme de fichiers json, nous avons retiré les bruits et corrigé les erreurs.
Nous avons ensuite testé plusieurs approches différentes : l'approche multi-label et l'approche Outlier et Novelty detection.
L'approche multi-label n'a pas donné de très bons résultats, pour l'améliorer nous aurions pu faire des recherches, cependant, nous avons cherché une autre approche : la méthode LOF.
La configuration avec un cas de Novelty puis de la méthode LOF, entraînée sur des bases de données vides de tout outlier, a donné de bons résultats : seulement 6% d'erreurs. Pour essayer de l'améliorer, nous avons ajouté la méthode TF-IDF, mais celle-ci n'a pas fonctionné. La configuration sans paramètre Novelty n'a pas été concluante non plus.

Si l’on avait eu plus de temps, nous aurions pu utiliser la méthode Doc2Vec qui aurait pu améliorer encore nos résultats. Bien que le taux de réussite de la méthode LOF de 94% soit très bon.