**Machine Learning**

# Choix Techniques

## Prétraitement des données

Pour utiliser de manière optimale un modèle de prédiction, il est nécessaire de faire un prétraitement des données pour les rendre plus simple à utiliser et à analyser pour la suite de notre travail. Ici, les données à traiter sont des paragraphes entiers de description d’un métier et d’une carrière, il s’agit donc de textes complexes. Nous allons utiliser la description de la première ligne, présente ci-dessous, pour montrer les différentes étapes de ce prétraitement.

She is also a Ronald D. Asmus Policy Entrepreneur Fellow with the German Marshall Fund and is a Visiting Fellow at the Centre for International Studies (CIS) at the University of Oxford. This commentary first appeared at Sada, an online journal published by the Carnegie Endowment for International Peace.

### Etape 1 : Normalisation - Ponctuation

Cette première étape consiste à supprimer toute la ponctuation du texte fournit, ainsi que de passer toutes les lettres en minuscule, pour ne pas avoir de perte de données dans les étapes suivantes.

she is also a ronald d asmus policy entrepreneur fellow with the german marshall fund and is a visiting fellow at the centre for international studies cis at the university of oxford this commentary first appeared at sada an online journal published by the carnegie endowment for international peace

### Etape 2 : Stop words

Nous allons ensuite simplifier nos textes en supprimant les mots qui ne seront pas utiles à l’étude de la description, il peut s’agir de déterminants, de pronoms, etc.. Pour cela, on va tout d’abord utiliser la fonction *word\_tokenize* pour découper notre texte en mots. A partir de là, on utilise la fonction *stop\_words* , qui contient un dictionnaire mots qui n’ont pas d’impacts pour le sens de la phrase, pour comparer avec les mots de notre texte et les supprimer de l’étude.

also ronald asmus policy entrepreneur fellow german marshall fund visiting fellow centre international studies cis university oxford commentary first appeared sada online journal published carnegie endowment international peace

### Etape 3 : Lemmatization vs stemming

Nous avons désormais supprimé tous les mots superflus de notre texte, il faut maintenant supprimer toutes les conjugaisons et autres pluriels de nos mots, afin d’avoir une base solide de mots similaire pour comparer et construire notre modèle. Pour cela, 2 options : la lemmatization ou le stemming.

D’un côté, le stemming rogne le début ou la fin du mot à partir d’un dictionnaire de préfixes et de suffixes.

La lemmatization lui permet de retrouver la racine des mots à partir d’une liste en retirant des lettres au mot de départ.

Par exemple, voilà notre texte après un stemming :

also ronald asmus polici entrepreneur fellow german marshal fund visit fellow centr intern studi cis univers oxford commentari first appear sada onlin journal publish carnegi endow intern peac

Dans ce cas, on a par exemple le verbe published qui est devenu publish, sa racine. Cependant, on remarque que beaucoup de mots sont faussés, tels que studies devenu studi ou encore policy devenu polici.

En face, la lemmatisation permet de remplacer les mots par une forme proche mais qu’on pourrait qualifier de plus simple. L’algorithme de la fonction utilise la morphologie des mots et se base sur un dictionnaire qui sert à faire le lien entre morphologie et le mot le plus proche :

also ronald asmus policy entrepreneur fellow german marshall fund visit fellow centre international study cis university oxford commentary first appear sada online journal publish carnegie endowment international peace

La lemmatisation semble donc plus efficace puisqu’elle transforme les mots en mot similaire. Par conséquent, le nombre d'occurrences des mots d’une même famille sera plus élevé et permettra de conserver le sens. Par exemple, le mot studies a bien été transformé en study, contrairement au stemming.

### Etape 4 : Vectoriseur TF-IDF

TF-IDF est une méthode de pondération qui permet de transformer nos données d'entrées, ici nos textes, en vecteurs en fonction de l'occurrence des mots qui le compose vis-à-vis de toutes les descriptions du fichier data.json. Dans notre cas, nous avons utilisé la classe TfidfVectorizer de sklearn pour vectoriser les données issues des premières étapes de prétraitement.

[‘also’, ‘ronald’, ‘asmus’, ‘policy’, ‘entrepreneur’, ‘fellow’, ‘german’, ‘marshall’, ‘fund’, ‘visit’, ‘centre’, ‘international’, ‘study’, ‘cis’, ‘university’, ‘oxford’, ‘commentary’, ‘first’, ‘appear’, ‘sada’, ‘online’, ‘journal’, ‘publish’, ‘carnegie’, ‘endowment’, ‘peace’]

(Etape 5 : Réduction des dimensions du jeu de données)

Les dimensions en sortie de la vectorisation TF-IDF dépend du nombre de données utilisées : plus il y a de texte, plus il y aura de mots, et donc plus il y aura de vecteurs en sortie. Sur l’ensemble de nos données, on a plus de 190 000 vecteurs. Cela peut poser problèmes pour manipuler d’aussi grandes dimensions de données et ralentir l’apprentissage.

L’Analyse en Composante Principale (ACP) permet de réduire la dimension des vecteurs en analysant ces vecteurs et en identifiant les dimensions omniprésentes. Avec l’ACP, nous pouvons donc réduire nos dimensions afin de ne retenir que celles qui auront été jugées comme les « plus importantes ». Cependant, il y a des risques de perte de précision sur notre modèle avec la réduction de dimensions. C’est pour cette raison que n’allons pas utiliser cette méthode.

## Modèle de prédiction

Après notre prétraitement de données, il faut choisir un modèle qui nous permettra d’avoir la meilleure prédiction possible. Pour cela, nous avons choisi de comparer le modèle K plus proches voisins et SVM. Afin de tester l’efficacité des modèles, nous avons pris un échantillon de nos données de base. Pour ne pas introduire de biais durant l’entraînement du modèle, nous avons utilisé un set de 2800 données composé de 100 textes de chaque type de métier (28 labels). Cette faible quantité de données de base nous permet d’orienter notre choix vers un modèle.

#### K plus proches voisins

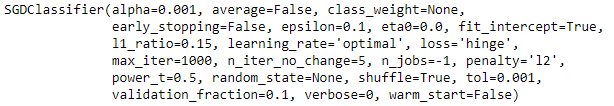
La première méthode que nous avons utilisée est celle des k plus proches voisins. C’est un des algorithmes les plus simples à utiliser. Cependant, les faibles résultats que nous avons eus ne sont pas très efficaces et encourageants si nous venions à le choisir.

#### Support Vector Machine (SVM)

La seconde méthode que nous avons testée est celle de la descente de gradient stochastique. C’est l’une des méthodes les plus favorables à l’analyse de texte car elle est capable d’être efficaces et correctes sur de grandes dimensions. Le modèle que nous avons utilisé est SGDClassifier() de la librairie Sklearn. Selon les nombreux paramètres que l’on peut modifier, notamment la fonction de perte (loss), nous pouvons avoir des modèles différents. Le paramètre *loss* par défaut du modèle est *hinge,* qui donne un SVM linéaire. Par rapport à un SVM classique, un SVM linéaire peut être entraîner de façon partielle pour éviter les problèmes de ressources matériels.

#### Bilan

Finalement, la descente de gradient est le modèle prédictif le plus rapide et le plus efficace des 2, et nous allons donc nous orienter vers ce choix.



## Evaluation du modèle choisit

***Score F1-Macro et Accuracy***

Dans un modèle de prédiction, on a l’accuracy qui est une manière d’évaluer le modèle. Elle est assez simple et standard, il suffit de diviser le nombre de prédictions vraies par le nombre totale de prédictions faites :

Une autre façon d’évaluer un modèle est le score F1-Macro. Il faut tout d’abord découper ce mot en 2 parties, avec le F1 Score d’un côté et Macro de l’autre.

Le F1 Score est un calcul pour la classification binaire (deux classes uniquement). Il se calcule avec les vrais positifs et négatifs, et faux positifs et négatifs :

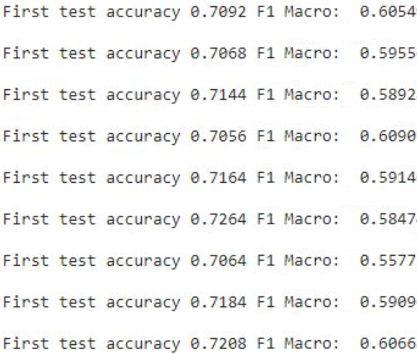
où est le taux de vrais positifs par rapport au nombre total de positifs

et est le taux de vrais positifs par rapport au nombre total de positif censé être obtenu

Le score F1-Macro est en fait un calcul pour la classification multi-classes, ce qui est notre cas. Pour cela, il suffit de faire la moyenne du score F1 pour chacune des classes. Il n’y a donc pas, à l’inverse de l’accuracy, de biais par rapport à la taille des classes (donc du nombre de personnes associées à un métier).

*Résultats*

Nous avons sélectionné le modèle que nous avons entraînés et avons regardé quels étaient les résultats de ces différentes évaluations. Pour cela, nous avons calculé les résultats sur 9 jeux de données :



Finalement, notre modèle a un résultat d’accuracy d’environ 71%, alors que le score F1-Macro est environ 10 points en dessous (60%). Ce ne sont pas forcément des résultats très élevés, mais nous pouvons ainsi comprendre que notre modèle sera plus efficace lors de la prédiction de métiers « populaires ».

# Résumé

* Pré-traitement
  + Normalisation de la description (ponctuation/minuscule)
  + Identification et extraction des ‘stop-words’
  + Lemmatisation
  + Vectorisation (TF-IDF)
* Modèle de type descente de gradient