Machine learning for natural language processing : Sujet 1

Alexandre Queant

April 2024

1 Introduction

La base de données principale à notre disposition se compose en fait de deux tableaux. Le premier comporte des informations vérifiées et enregistrées à la main sur chaque individu à partir des registres manuscrits. La deuxième réalise supposément la même tâche mais à l'aide d'un logiciel de traitement d'image. Enfin, une dernière colonne précise le sexe vérifié de chaque individu. J'ai donc choisi de travailler exclusivement à partir du second tableau, car arriver à prédire le sexe d'un individu à partir du traitement imparfait de son recensement par un logiciel de traitement d'image permet de supprimer le travail humain dans le processus. Pour ce qui est du pre-processing, on enlève seulement la ponctuation, les accents et les majuscules pour chacune des variables.

On se retrouve donc avec une base de données très courte de 241 individus et donc les données sont très altérées. Au delà de la taille de la base de données, il y a deux difficultés principales : peu d'individus ont un attribut autre que le prénom enregistré comme non nul (28 lignes dont seulement le prénom et l'âge sont exploitables), et certains prénoms sont mal transcrits par le logiciel de reconnaissance d'image. Il y a donc des individus enregistrés avec des prénoms tels que : 'oarguerite', 'gadeleine', 'ctiennette' ou 'zean' (fig 1).

Les données d'âge sont également dures à exploiter, car nous n'avons pas la date de création du registre, et pour certains individus seulement la date de naissance est enregistrées. En effet, pour 75 individus, seulement la date de naissance est renseignée. Pour les autres, c'est l'âge qui est enregistré, et là aussi parfois imparfaitement. La base de données concerne par ailleurs 56% d'hommes et 44% de femmes.

Une base de données supplémentaire est à notre disposition, et elle contient les fréquences d'apparition de chaque prénom dans le registre en tant qu'homme ou femme. Le prénom est en effet la donnée cible principale pour ce qui est de la classification en sexe de l'individu, car à première vue, le nom de famille ne comporte aucune information sur le sexe, et notre base de données est très incomplète

	nom	prénom	date_naissance	lieux_naissance	employeur	relation	profession	état_civil	éducation	sex
18	vivier	angene	180n	aulgique	NaN	ch	NaN	d	NaN	0
24	debouese	herandre	42	chef	NaN	NaN	patron	NaN	NaN	0
25	marhi	oarguerite	30	d	NaN	ep	NaN	NaN	NaN	1
34	NaN	vigmie	1852	savit	NaN	ep	NaN	NaN	NaN	1
65	fleury	angloise	7	id	NaN	NaN	NaN	'	NaN	0

Figure 1: Exemples de prénoms mal reconnus.

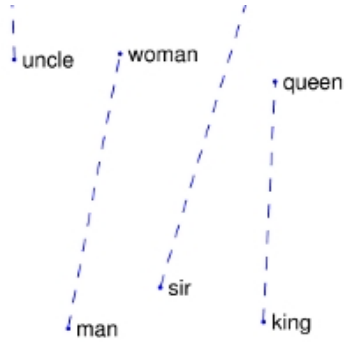
pour ce qui est des autres covariables. Après étude de la base de données et réflexion sur les variables d'intérêt pour la prédiction du sexe, nous utiliserons les colonnes prénom, employeur, relation et profession comme variables de prédiction.

2 Choix et analyse du modèle

Comme mentionné plus haut, une des faiblesses des données à notre disposition est leur faible taille. Pour notre tâche de classification, les 241 lignes de la base ne devraient pas suffire à entraîner n'importe quel modèle de prédiction de 0. Pour cela, j'ai décidé de me tourner plutôt vers un modèle pré-entrainé, que je pourrai réutiliser pour faire mon propre entraînement. Il y avait selon moi deux manières de considérer les données de prénoms : en prenant le prénom sous sa forme initiale, en entier, ou alors en utilisant un tokenizert à la manière des modèles comme BERT qui vont découper les mots en sous-entités pour l'entraînement. Le faible nombre de lignes ne justifiant pas selon moi le recours à un modèle préentrainé de type BERT, j'ai décidé de d'abord tenter de considérer les prénoms en entiers comme données, avec la possibilité de les découper en sous-entités si mes résultats n'étaient pas concluant.

Mon idée principale a été de partir d'embeddings déjà entrainés représentant bien la position du prénom par rapport au sexe de l'individu. Pour cela, j'ai utilisé des embeddings GloVe de 50 dimensions pré-entrainés sur un corpus de 6 milliards de textes provenant de Wikipédia. Les embeddings GloVe sont ici utiles car ils sont très performants sur les taches d'analogie telles que :

$$king - man + woman = queen$$



L'intuition est donc que le groupe des prénoms masculins aient des embeddings éloignés du groupe des prénoms féminins, et donc qu'un classifieur aura peu de mal à discerner le sexe d'un individu à partir de l'embedding de son prénom.

Cependant, pour que cette technique fonctionne il faudrait que tous les prénoms possèdent déjà un embedding dans ceux pré-entrainés, et cela n'est naturellement pas le cas pour les prénoms mal reconnus comme 'zean' ou 'oarguerite' par exemple. De plus, des prénoms comme 'marie' peuvent par exemple avoir un embedding très proche des autres prénoms féminins, car c'est un prénom majoritairement porté par les femmes, mais peuvent également être portés par des hommes. Il faut donc des informations supplémentaires aux embeddings des prénoms.

La seconde idée est donc d'utiliser également les embeddings des covariables, et de former un embedding par individu par un moyenne pondérée entre embedding du prénom et embedding des covariables. Par exemple, si un homme s'appelle 'marie' mais que sa relation est 'chef', alors il est quansiment certain que ce soit un homme. Alors pour chaque individu, si *embed*(prénom) est

l'embedding du prénom et que *embed*(variable) est l'embedding de la variable (dans les variables d'intérêt : employeur, relation et profession) on attribue l'embedding :

$$embed(\text{prénom}) + \alpha \sum_{\text{variable} \in \text{variables d'intérêt}} embed(\text{variable})$$

avec un α à déterminer.

Cela est réalisable car tous les individus possèdent au moins une variable ayant un embedding non nul dans le modèle pré-entrainé. De plus, afin de régulariser l'embedding des prénoms et des variables au sein de leurs voisins, supposément appartenant au même sexe, nous prenons l'embedding de chaque variable comme barycentre de ses n plus proches voisins (avec $n = 30$) par défaut. Nous justifierons ce choix dans la prochaine partie.

$$embed(\text{variable}) = \frac{1}{n} \sum_{\text{voisin} \in n \text{ plus proches voisins}} embed(\text{voisin})$$

Enfin, l'objectif est d'utiliser ces embeddings comme inputs pour un simple multilayer perceptron à 3 ou 4 couches. L'intérêt de ce modèle est donc sa simplicité, et un temps d'entraînement très faible comparativement à si on ré-entrainait un transformers. Toutefois, les choix faits ici laissent supposer que l'efficacité du modèle est très dépendante de la qualité des données retranscrites par le logiciel de traitement d'image.

2.1 Expérimentation

Pour l'expérimentation, nous commençons par un split aléatoire du dataset, avec 70% de train set et 30% de test set. De plus nous fixons un batch size de 32, un learning rate de 0.001, des entraînements de 100 epochs et un weight decay de 1e-5.

2.1.1 Nombre de couches et choix de α

Dans cette partie, nous cherhcons à déterminer une forme simple et efficace d'un perceptron et de notre α pour répondre à notre problématique de classification. Les perceptrons testés ont une forme simple : une dimension d'entrée de 50 (dimension des embeddings), puis une alternance de couches linéaires de tailles 64 et 128, avec à chaque sortie de couche une fonction d'activation ReLU et un dropout de 0.5. La sortie est quant à elle passée à travers une fonction sigmoïd, pour la classification binéaire.

Pour avoir une première idée des hyperparamètres optimaux, nous réalisons une validation K-Fold sur notre dataset, en faisant varier α et le nombre de couches du perceptron (fig 2).

Nous remarquons tout d'abord une très bonne précision dans la classification, et ce peu importe le choix des hyperparamètres. Toutefois, après analyse du K-Fold, un $\alpha = 0.4$ semble être un choix cohérent, et il restait à choisir le nombre de couches du perceptron. De plus, nous justifions ici également l'intérêt de lier l'embedding d'un mot avec celui de ses plus proches voisins. Dans ces tests, nous regardons l'efficacité moyenne cummulée d'un perceptron en fonction de son nombre de couches, avec un $\alpha = 0.4$ et un nombre de voisins considérés de 30 par mots, voir fig 3. De plus nous fixons également le nombre de couches à 6, le $\alpha = 0.4$ et nous observons dans la fig 4 l'effet sur l'efficacité moyenne cummulée du nombre de voisins considérés.

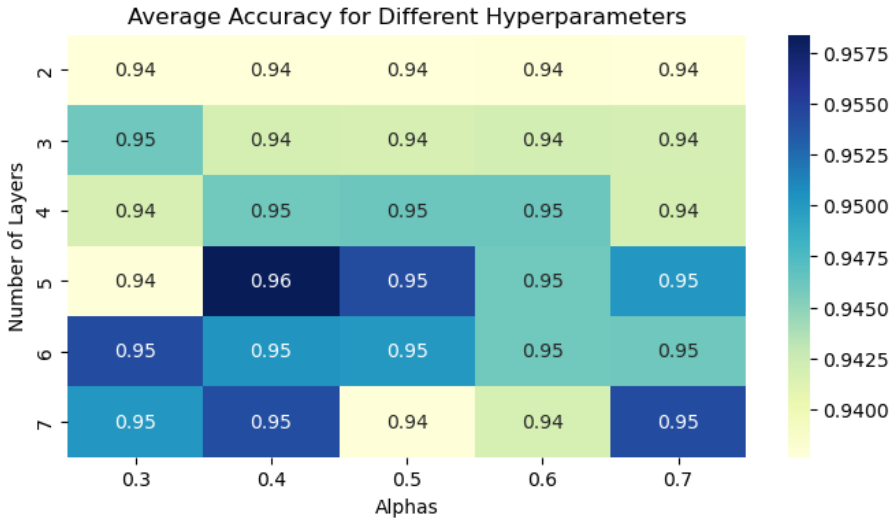


Figure 2: Validation croisée KFold sur le nombre de couches et le α.

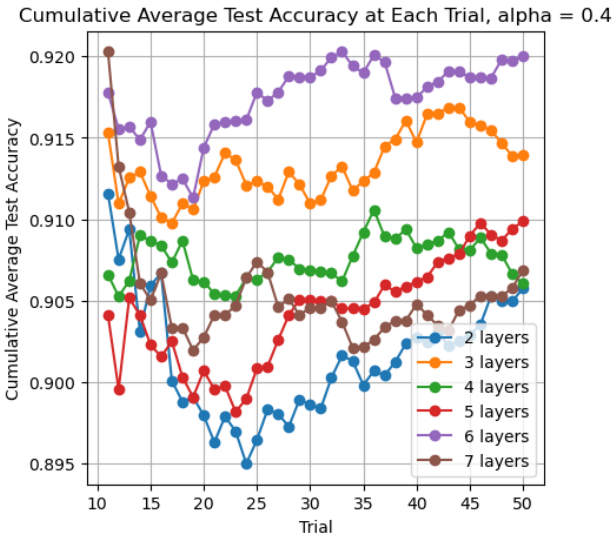


Figure 3: Efficacité par nombre de couches.

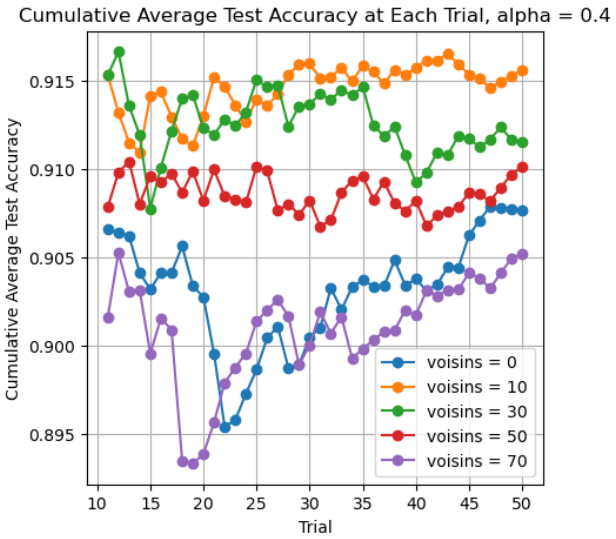


Figure 4: Efficacité par nombre de voisins.

Les courbes nous indiquent qu'un perceptron à 6 couches peut être légèrement meilleur que les autres, même si l'écart est très faible. De plus, nous remarquons également que de prendre l'embedding d'un mot comme l'embedding du barycentre de ses plus proches voisins possède bien un intérêt. Nous fixons finalement un structure à 6 couches, avec un $\alpha = 0.4$ et un nombre de voisins de $n = 10$.

2.1.2 Performance, faiblesse du modèle et recommendations

On atteint très vite une haute précision, mais qui semble stagner rapidemment. Une modification des paramètres et hyperparamètres ne résoud pas ce problème. En observant les prénoms mal classifiés par le perceptron, nous pouvons émettre quelques hypothèses sur cette stagnation (fig 6). Un sexe de 1 veut dire féminin, et 0 masculin.

Figure 5: Training loss et test accuracy du modèle au cours de l'entraînement.

	nom	prénom	date_naissance	lieux_naissance	employeur	relation	profession	état_civil	éducation	sex	single embedding	preds
2	pyrin	marie	55	NaN	NaN	d	NaN	NaN	NaN	1	[tensor(0.1361), tensor(0.7602), tensor(0.1158...	0.0
47	giron	marie	74	id	NaN	id	NaN	NaN	NaN	1	[tensor(0.0665), tensor(0.4672), tensor(0.2740...	0.0
138	ducher	ctiennette	75	co	NaN	NaN	NaN	s	NaN	1	[tensor(0.), tensor(0.), tensor(0.), tensor(0....	0.0
146	pesin	marie	1891	coulonges	NaN	culliniere	lameau	NaN	NaN	0	[tensor(0.1798), tensor(0.4375), tensor(-0.282...	1.0
193	detouset	marthy	99nie	"h	NaN	NaN	NaN	NaN	NaN	1	[tensor(0.), tensor(0.), tensor(0.), tensor(0....	0.0
204	d	entonie	16	d	NaN	f	NaN	NaN	NaN	0	[tensor(-0.2640), tensor(0.7027), tensor(0.670...	1.0
205	foury	jean-haptiste	28	chef	NaN	chemin	NaN	NaN	NaN	0	[tensor(0.0997), tensor(-0.1132), tensor(-0.30...	1.0
217	bazelique	mareleine	38	niaore	NaN	NaN	NaN	NaN	NaN	1	[tensor(0.), tensor(0.), tensor(0.), tensor(0....	0.0
219	soziche	franco	2	id	NaN	NaN	id	NaN	NaN	1	[tensor(0.0075), tensor(-0.2782), tensor(0.209...	0.0

Figure 6: Exemple d'individus mal classifés par le modèle.

On remarque que les indivius mal classés peuvent être divisés en deux groupes. Soit le prénom est mal retranscrit par le logiciel de reconnaissance d'image, avec en plus des variables d'intérêt très mauvaises ou inexistantes. Dans ce cas, l'individu ne peut pas avoir d'embedding représentant son sexe dans le modèle GloVe, et il est donc impossible de bien le classifier (par exemple "entonie" dans la fig 6). Ou bien le prénom existe dans le modèle GloVe et possède un embedding placé proche d'un sexe, mais l'individu est en fait du sexe opposé et les informations complémentaires ne sont pas d'assez bonne qualité pour le déterminer (par exemple les hommes nommés Marie dans la fig 6).

C'est probablement sur ce type de cas qu'un modèle plus complexe, prenant en compte des entités plus petites que des mots (syllabes, lemmas,...), pourrait mieux performer que le notre. Toutefois un tel modèle risque de demander un entraînement bien plus long que nous réalisons ici (à peine quelques secondes), et les performances ne seraient pas significativement meilleures.

Nous avons donc montré ici qu'en réutilisant des embeddings pré-entrainés d'un modèle GloVe, et en couplant cela avec un simple MLP, nous parvenons à une classification peu couteuse et très précise du sexe d'un individu à partir des données offertes par le logiciel de reconnaissance d'image. La rapidité du modèle proposé permettrait de l'adapter facilement et sans coût à l'ensemble du corpus, mais nous avons toutefois vu qu'un faible nombre de cas particuliers ne pourraient pas être bien classifiés.