UNIVERSITÉ DE MONTRÉAL

IFT6285
TRAITEMENT AUTOMATIQUE DES LANGUES NATURELLES

Projet 1: Classification de blogs

par:

Ilyas Amaniss
(20148123)
Henri-Cedric Mputu Boleilanga
(20174905)
Eugénie Yockell
(20071932)

Date de remise: 10 novembre 2020

1 Introduction

Pour ce projet, un ensemble de texte composé de 18 819 auteurs qui ont écrit environs 657 000 publications provenant de blogs nous a été présenté. Venant d'auteurs multiples d'internet, ces publications peuvent contenir beaucoup d'erreurs. L'objectif du projet sera d'utiliser ces blogs afin de prédire le genre, l'âge et le signe astrologique de l'auteur ayant écrit une certain publication. Pour se faire, le corpus donné a tout d'abord été prétraité, transformé en *features* et différents modèles ont été entraînés. Ainsi, l'impact des *features* utilisés et des modèles sur chacune de ces trois tâches seront étudié.

2 Métrique et Architecture

Pour toutes les évaluations que nous avons eu à faire pour ce projet, nous avons utilisé la fonction *accuracy_score* de scikit-learn qui mesure la précision d'un modèle en comparant ses prédictions aux classes réelles et détermine une moyenne de réussite. La précision est donc le nombre de prédictions correctes divisé par le nombre total de prédictions faites.

Tout au long du projet, nous avons travaillé en utilisant la plateforme de collaboration en ligne: Google Colab. Un des avantages de cette plateforme est de facilement pouvoir partager le code de chaque coéquipier rapidement et facilement. Tout notre code a donc été écrit et exécuté sur le *notebook* qui lui roule sur les serveurs de Google. Le processeur de Google Colab est Intel(R) Xeon(R) CPU @ 2.30GHz et 13 GB de RAM. Aucune demande à un ordinateur en particulier n'a été faite.

3 Prétraitement

Cette section décrira les étapes de prétraitement appliquées aux textes de chacun des auteurs avant d'être transformé en *features*. Tout d'abord, les *stop words* de la langue anglaise ont été retiré dans chacune des publications. Ensuite, chacun des mots sont passé par une étape de *stemming*. Finalement, en utilisant ce nouveau corpus où tous *stop words* ont été retiré et les mots ont été *stemmed*, deux type des *features* ont été crées: un vecteur *Word2Vec* et un vecteur *Term Frequency-inverse Document Frequenecy (TFIDF)*.

Les *stop words* sont des mots considérés comme n'enrichissant pas le contenu d'une phrase. Quelques exemples de *stop words* de la langue anglaise sont *it, a, I, to*. Ces mots sont souvent retirés des corpus lors de tâches d'entraînement en traitement automatique de langues naturelles car cette étape a souvent montré une amélioration dans les performances des modèles comme en tâche de classification de sentiments (1).

Le *stemming* permet de ramener un mot vers sa racine. Par exemple, *argue*, *argued*, *arguing* sont ramené vers la racine *argu*. Tous ces mots ont la même signification sémantique. Il est alors logique de les représenter par un seul mot. Cette opération permet de donner rapidement de l'importance à des mots d'un texte en augmentant sa fréquence. Cela devient particulièrement utile dans le plongement de mots avec TFIDF où on s'intéresse à la fréquence d'un mot. Pour performer cette opération, le `SnowballStemmer` de la librairie `nltk` a été utilisé.

4 Plongement de mots

Après le prétraitement des données, un autre facteur important dans la prédiction à partir de texte est le plongement de mots. Nous nous sommes intéressé pour ce projet à deux plongements de mots populaires: le *Word2Vec* présent dans la librairie `gensim` et le `TFIDFVectorizer` de *scikit-learn*. Afin d'améliorer la qualité du vocabulaire appris dans chacun des deux cas, les textes de chacun des auteurs ont été combiné et donné aux plongements de mots.

4.1 Word2vec

Basé sur des réseaux de neurones, le *Word2Vec* a été le premier plongement de mots testé lors de ce projet, car il semble obtenir d'intéressants résultats lors de tâches de classification similaires (2). Le *Word2Vec* permet de représenter un mot en un vecteur d'une taille désirée. Nous avons utilisé *Word2Vec* avec l'apprentissage bag-of-words qui prédit le mot courant en fonction de la représentation distribuée du contexte. Donc, afin de représenter le texte d'un auteur, chaque mot est d'abord transformé en vecteur et ensuite une moyenne de tous les vecteurs de ce texte est calculée afin de n'obtenir qu'un seul et même vecteur représentant ce texte. L'utilisation de cette technique peut s'avérer avantageuse, car elle permet de préserver la plupart de l'information pertinente d'un texte tout en réduisant sa dimensionnalité. Donc, bien que notre corpus soit très grand, cette représentation de mots devrait nous permettre de réduire considérablement le temps d'entraînement des modèles.

4.2 Term Frequency–inverse Document Frequency (TFIDF)

Le second plongement de mot testé lors ce projet fut le TFIDF. L'idée principale de ce plongement de mots est de contrecarrer le fait que certains mots présent à très grande fréquence dans le corpus puissent faire de l'ombre à certain mots moins fréquents mais donnant beaucoup de valeur à une phrase. Ainsi, en utilisant ce plongement de mots qui a été montré comme étant très

efficace dans des tâches de classifications de texte (3), chacun des textes des auteurs ont été transformé en vecteur TFIDF et donné en entré aux modèles testé dans ce projet.

4.3 Comparaison

Nous avons étudier les performances de *Word2Vec* et *TFIDF* sur les trois tâches (prédictions du genre, âge et signe astrologique). Les résultats des deux plongements de mots ont été mesuré pour chaque tâche et avec les quatre meilleurs modèles de ces tâches. Les résultats pour la prédiction du genre sont présenté au tableau 2.

Les résultats pour la prédiction de l'âge sont présenté au tableau 5. Les résultats montrent tout d'abord très clairement que les modèles utilisant le TFIDF obtiennent des résultats supérieurs à ceux utilisant le *Word2Vec*. En effet, le classificateur Naive Bayes qui est le modèle ayant obtenu les meilleurs résultats lors de ces tests préliminaires a vu sa précision augmenter de presque 10% lorsque le TFIDF a été utilisé. Lorsque le *Word2Vec* fut utilisé, les pires résultats ont été obtenus avec la forêts d'arbres décisionnels et le Naive Nayes obtenant respectivement 50.21% et 48.59% de précision.

Les résultats pour la prédiction du signe astrologiques sont présenté au tableau 8. Les mesures des deux plongements de mots à été mesuré avec les quatre meilleurs modèles pour chaque cas. Chaque modèle est utilisé avec ces paramètres par défaut. Les conséquences des deux plongements de mots est moins évidente sur une tâche aussi difficile que le signe astrologique. La différence la plus flagrante est sur le modèle MLP. Cette différence peut s'expliquer par la taille des *features*. Le modèle MLP excelle avec une plus grand quantité de *features*, car il a plus de facilité à les exploiter au maximum.

En se basant sur les résultats obtenus durant nos expériences, il est claire que le type de *features* utilisés pour un modèle aura un impact important sur la précision dans cette tâche de classification. En effet, comme on peut le voir le TFIDF a surpassé son compétiteur *Word2Vec* dans toutes les tâches de prédiction. Plusieurs facteurs peuvent expliquer ce comportement.

Tout d'abord, notre modèle *Word2Vec* à une taille de 100 *features* pour chaque mot et ne prend qu'une moyenne des vecteurs de mots où on accorde la même importance à chaque mot. On perd beaucoup d'information, surtout dans un grand texte. Cette perte d'information n'est qu'empirer par le fait que la taille soit aussi petite. Les modèles ont ainsi une grande difficulté à extraire de l'information. Tandis que TFIDF peut créer de grand vecteurs d'information et donne une plus grande importance au mots pertinents d'un certain type de document et une moins grande importance au mots communs. On conserve alors mieux l'information qu'avec *Word2Vec*.

5 Modèles

Puisque chaque tâche a des besoins très différents, chaque tâche a été étudié de façon individuelle. En effet, l'autre solution aurait été de produire un seul modèle faisant la prédiction du genre de l'âge et du signe astrologique, cependant cette idée semblait manquer de flexibilité. Donc, dans cette section, nous allons étudier les différents modèles produits pour prédire le genre, l'âge et le signe astrologique.

Plusieurs algorithmes de la librairie scikit-learn ont été utilisé, dont les algorithmes Support Vector Machine (SVM), Linear Regression (LR), Gradient Boosting, Random Forest et même un réseau neuronal, le Multi Layer Perceptron (MLP). Les modèles de cette librairie ont été utilisé afin d'uniformiser les expériences et de faciliter le développement et la recherche grâce à l'utilisation d'une même pipeline de développement. Les avantages et inconvénients de chaque algorithme seront exploré dans les sections suivantes.

Pour chaque tâche, afin de déterminer à première vue quel type de features et quels modèles devraient être explorés plus en profondeur, plusieurs combinaisons de systèmes ont été battit. Ensuite, les modèles ayant le plus grand potentiel ont été pris afin de performer une recherche d'hyperparamètres sur ceux-ci afin d'optimiser leur performance. Enfin, ces résultats sont analysé afin de comprendre le comportement de chaque système et de la tâche en particulier.

Des *baselines* ont été défini à l'aide de DummyClassifier de scikit-learn. Les résultats sont présentés à la tableau1. Nous allons chercher à battre ces résultats.

Tâche	Précision
Genre	50.60%
Âge	38.63%
Signe	6.47%

Table 1: *Baselines* définie avec DummyClassifier pour chaque tâche.

5.1 Prédiction du genre

La prédiction du genre est, dans notre cas, une classification binaire (homme-femme), nous avons donc débuté notre exploration par une approche supervisée avec deux algorithmes de classification binaire. Le SVM, qui est l'un des meilleurs algorithmes conçus à cette fin et la régression linéaire, LR. Nous avons ensuite créé un modèle basé sur une approche non-supervisé en utilisant K-cluster afin de regrouper les hommes et les femmes chacun de leur côté. Notre dernier choix a été porté sur une

architecture de réseau neuronal: le MLP. Pour chacun de ces algorithmes, nous avons entraîné et tester nos modèles en utilisant nos deux plongements: *Word2Vec* et TF-IDF. Tous nos modèles ont été entraînée avec les paramètres par défaut des algorithmes: pour K-cluster, le nombre d'ensembles a été initialisé à deux. Une recherche des meilleurs hyperparamètres sera faite pour le meilleur modèle possible.

	Feature	SVM	LR	K-cluster	MLP
Précision	Word2Vec	55.92%	49.40%	49.38%	50.35%
	TFIDF	65.32%	63.70%	49.53%	61.73%

Table 2: Comparaison préliminaire de performance de modèles sur la tâche de Genre

Pour notre analyse des résultats, nous nous sommes concentré sur TFIDF qui nous a donné les meilleurs résultats en général comme le présente la tableau2. Nous allons donc essayer de comprendre pourquoi il y avait une différence de plus de 15% entre le SVM et K-cluster.

K-cluster commence par placer aléatoirement deux centroïdes (initialisé à deux, car nous savons que nous avons deux groupes hommes/femmes) et représente chacune des entrées, chaque texte par un point sur l'espace vectoriel. Itérativement, la distance entre chaque centroïde et les points est recalculée puis les centroïdes se déplace vers le centre des points qui sont plus proches d'eux. L'algorithme se termine lorsque la position où les groupes ne changent plus ou lorsque la distance à laquelle les centres de gravité changent ne dépasse pas un seuil prédéfini (ici la valeur par défaut est utilisée).

Le SVM, lui, représente également chaque texte par un point sur l'espace vectoriel mais au lieu d'utiliser des centroïdes, le SVM trace 3 lignes: 1 ligne seuil à égale distance des deux groupes classes et deux lignes de support de chaque côté de la ligne seuil.

On voit donc que dans les deux cas, il y a une séparation qui doit se faire entre les textes écrits par les hommes et ceux écrit par les femmes (entre deux ensembles pour K-cluster; de chaque côté de la ligne seuil pour SVM). Or, cette séparation a été plutôt difficile à faire et se traduit par des résultats autour de 50% pour le K-cluster. Il était alors nécessaire de comprendre pourquoi ces points étaient si difficiles à séparer. Pour cela, nous avons extrait les mots du corpus les plus fréquences chez les hommes et chez les femmes.

	1	2	3	4	5	6	7	8	9	10
Femme	like	go	get	one	know	time	think	realli	day	want
Homme	like	go	one	get	urllink	time	know	think	day	peopl

Table 3: Comparaison des 10 premiers mots les plus fréquents pour chaque genre.

Nous avons observé les 50 mots les plus fréquents et nous pouvons voir au tableau 3(seul les 10 premiers sont affichés) que les mots sont presque les mêmes, et ce, peu importe le genre. Il n'y a que très peu de différences. Ceci explique pourquoi K-cluster a eu de la difficulté à séparer les deux genres.

Nous avons voulu savoir pourquoi, malgré cette forte similitude dans les mots, le SVM a tout de même obtenu les meilleurs résultats. La différence principale est que le SVM est un algorithme supervisé où nous connaissons le genre de chaque auteur. De l'autre côté, K-cluster n'a connaissance que des textes qu'il représente sous forme de points sans savoir si le point est au bon endroit, tandis que l'information sur le genre permet au SVM de faire une meilleure séparation, car il fera en sorte de placer les points au bon endroit en fonction de leur genre et de trouver la meilleure ligne de seuil possible. Lorsqu'il est difficile de trouver une ligne seuil, le SVM a l'avantage de pouvoir augmenter de dimension (de 2D à 3D par exemple) et ainsi d'avoir un plan seuil (toujours accompagné de 2 autres plans de part et d'autres) au lieu d'une droite seuil. Ceci permet donc au SVM de séparer deux classes plus aisément, ce qui a probablement été le cas pour notre corpus.

kernel	gamma	C	max iter	Précision	Temps d'entraînement
rbf	scale	100	-1	67.41%	45.23 min

Table 4: Les hyperparamètres et la précision du SVM.

Les hyperparamètres de SVM que nous avons exploré sont les suivants (ils ont été choisi aléatoirement): Kernel: Détermine le type de noyau à utiliser dans l'algorithme ('rbf', 'poly', 'linear' et 'sigmoïde'); Gamma : Coefficient de noyau pour 'rbf', 'poly' et 'sigmoïde'; Max_iter : Limite le nombre d'itérations dans le solveur ; C: Paramètre de régularisation. La force de la régularisation est inversement proportionnelle à C. Les résultats sont présentés au tableau 4 et représente le meilleur score pour la classification du genre.

Notre baseline pour la classification du genre est de 50,60% comme on peut le voir à la tableau1. Notre modèle le moins performant, qui est K-cluster avec un plongement de mots utilisant *Word2Vec*, a donné un résultat légèrement plus bas que notre *baseline*. Notre meilleur modèle, qui est SVM avec les hyperparamètres que nous avons trouvés grâce au random search, a eu un score meilleur que le *baseline* avec une augmentation de 16.81%.

5.2 Prédiction de l'âge

La tâche de prédiction de l'âge consiste en une classification des trois groupes suivants: les auteurs âgés de 0 à 19 ans, de 20 à 29 ans et des plus de 30 ans. Les modèles qui ont été exploré pour cette tâche sont le Random Forest, le gradient boosting, le MLP et le classificateur Naive bayes. Hypothétiquement, le réseau neuronal étant le modèle le plus grand et sophistiqué semble avoir le plus grand potentiel pour cette tâche. Cependant, le tableau 5 montrent la supériorité de deux modèles, par contre, très différents: le classificateur Naive Bayes et le réseau neuronal MLP. Ces deux meilleurs modèles ont donc été sélectionné et une recherche d'hyperparamètres fut lancé sur chacun d'eux.

	Feature	Random Forest	Gradient Boosting	MLP	Naive Bayes
Précision	Word2Vec	50.21%	53.76%	53.77%	48.59%
	TFIDF	54.50%	55.00%	56.50%	59.90%

Table 5: Comparaison préliminaire de performance de modèles sur la tâche de l'âge

Pour le Naive Bayes, les deux hyperparamètres qui ont été exploré sont les valeurs d'alpha et la normalisation des poids. La valeur d'alpha, entre 1 et 0 (aucun lissage), permet de contrôler le lissage Laplace présent dans ce classificateur afin d'éviter une probabilité de zéro pour l'une des classes. La normalisation des poids, elle, permet de performer une seconde normalisation afin d'éviter que les textes beaucoup plus longs soient trop dominants (4). Plus spécifiquement, le Complement Naive Bayes a été utilisé. Ce modèle utilise plusieurs heuristiques afin de résoudre les problèmes souvent recontré avec le Naive Bayes classique (4).

Le MLP, avait lui, plusieurs hyperparamètres à explorer: le paramètre de régularisation (alpha), la taille du réseau (le nombre de couches et de neuronnes), l'initialisation du rythme d'apprentissage, le nombre d'epochs d'entraînement, l'algorithme d'optimisation de poids et la fonction d'activation. Le nombre de couches et de neuronnes d'un réseau neuronal permettent de contrôler la taille et le nombre de transformations faites dans ce modèle. L'initialisation du rythme d'apprentissage est la valeur initiale de la vitesse à laquelle l'algorithme d'optimization des poids (comme l'algorithme d'Adam ou la simple descente graduelle) met à jour les poids du réseau. Finalement, le type d'activation, par exemple 'Relu' ou 'Tanh', est la fonction de transformation faites à chaque noeud du réseau. Donc, les meilleurs hyperparamètres et leur temps d'entraînement pour chacun de ces deux modèles sont montré dans les tableaux 6 et 7.

Taille	Activation	Solver	Apprentissage	Alpha	Epochs	Précision	Temps d'entraînement
(100, 1)	Tanh	Adam	0.0001	0.0001	200	59.44%	25.81 min

Table 6: Les hyperparamètre et la précision du MLP.

Alpha	Normalisation	Precision	Temps d'entraînement
1	Non	66.73%	48.21 sec

Table 7: Les hyperparamètres et la précision du Naive Bayes.

Étonnement, le classificateur Naive Bayes a été le modèle ayant obtenu les meilleurs résultats avec une précision de presque 67% et ce, en une fraction de temps d'entraînement du MLP avec un temps d'entraînement de seulement 48 secondes, comparativement aux 25.81 minutes du réseau neuronal. En effet, il aurait été facile de croire, comme on le suggérait, que le MLP serait le meilleur modèle. Ces résultats peuvent être expliqué par le fait que le Naive bayes a souvent été démontré dans la littérature comme ayant de très bons résultats pour les tâches de classification de textes (4). L'une des raisons de ses bonnes performances est les heuristiques implémenter dans la version Complement Naive Bayes tel qu'apprendre à classifier une classe sans jamais utiliser de textes venant de cette classe particulière. Cela permet de réduire les tendances du modèle a préférer une classe envers une autre (4).

Il est aussi important de comprendre que les résultats sont toujours spécifiques au problème et au corpus utilisé et qu'aucune règle générale ne peut être appliqué quant à la supériorité du Naive Bayes face au MLP, par contre encore une fois dans cette tâche de classification de texte, cette tendance se maintient. En effet, il est aussi utile de mentionner que le réseau neuronal qu'est le MLP n'était peut-être pas le meilleur réseau à utiliser lors de traitement de textes. En effet, un type de réseau récurrent tel que le Long Short-term Memory (LSTM) aurait été préféré due à la nature séquentielle des mots dans un texte. Cela pourrait donc expliquer les différences de performance du simple Naive Bayes comparativement à un apprentissage profond avec ce réseau neuronal pour la classification de l'âge.

5.3 Prédiction du signe astrologique

Théoriquement, les prédictions des signes astrologiques ne devraient pas dépasser des prédictions aléatoire. En effet, il n'existe pas de preuves scientifiques corrélant le signe astrologique et la personnalité d'un individu. Nos résultats devraient alors concordés avec cette hypothèse sauf si les données sont artificielles.

Au tableau 8 nous avons les performances des quatre meilleurs modèles étudiés. Ces modèles ont été choisis parmi différents algorithmes comme classifieur bayésien naïf (Naive Bayes), *support vector machine* (SVM), régression logistique, régression linéaire, forêts d'arbres décisionnels et le gradient boosting. Les modèles proviennent de la librairie scikit-learn et XGBoost. Le VotingClassifier fait voter les modèles LogisticRegression, RandomForestClassifier, ComplementNB. On remarque que les meilleurs modèles sont le MLP et VotingClassifier. Ce sont ces deux modèles que nous allons explorer plus en détail. Comme mentionné dans la section 4, on utilisera aussi un *embedding* avec *TFIDF*, car sa performance est la meilleure. Encore une fois, on s'attend à ce que le réseau neuronal performe le mieux, car c'est un modèle complexe et sophistiqué.

	Feature	MLP	SVM	VotingClassifier	Naive Bayes
Précision	Word2Vec	6.48%	6.43 %	8.90 %	8.24%
	TFIDF	9.01%	8.10%	8.97%	8.29 %

Table 8: Comparaison préliminaire de performance de modèles sur la tâche de prédiction du signe astrologique.

Nous allons étudier les différents hyperparamètres des modèles MLP et VotingClassifier de scikit-learn. Pour le *MLP*, les mêmes hyperparamètres présentés pour la tâche de classification d'âge ont été exploré. Un *grid search* a donc été effectués sur tous ces hyperparamètres. Le meilleur modèle est représenté au tableau 9.

Pour le modèle de VotingClassifier, un *grid search* a été appliqué sur chaque modèle unique pour étudier les meilleurs paramètres. Pour le modèle de LogisticRegression, on étudie le paramètre de tolérance pour l'arrêt (stopping criteria) et le nombre d'epochs d'entraînement. Pour le RandomForestClassifier, on étudie les paramètres de *min_samples_split*, soit le minimum d'exemples nécessaire pour diviser un noeud, et le paramètre qui définit le nombre d'arbres dans la forêt. Pour ComplementNB, les mêmes paramètres qu'à la prédiction de l'âge sont explorés, soit le lissage de Laplace (alpha) et la normalisation des poids. Le meilleur modèle est représenté au tableau 10.

Taille	Activation	solver	Apprentissage	Alpha	Epochs	Précision	Temps d'entraînement
(6,5)	*relu*	*lbfgs*	0.001	1×10^{-6}	200	10.17%	48.82 sec

Table 9: Les hyperparamètres et la précision du MLP.

Modèle	Paramètres	Précision	Temps d'entraînement
LogisticRegression	*stopping criteria*=0.001 et *epoch*=200		
RandomForestClassifier	*min_samples_split*=4 et *n_estimator*=100	9.71%	3.54 minutes
ComplementNB	*alpha*=1		
MLPClassifier	Paramètre du tableau 9 avec une taille de (6, 6, 6, 6)		

Table 10: Les hyperparamètres et la précision du VotingClassifier.

Avant d'analyser les résultats des modèles, il est important de s'intéresser à la répartition de l'ensemble d'entraînement et de test, particulièrement dans un cas où le modèle performe aussi mal. En effet, s'il y a une classe plus présente dans l'entraînement, un modèle aurait le risque de prédire uniquement cette classe. Heureusement, l'ensemble d'entraînement est bien réparti pour chaque signe astrologique. Par contre, l'ensemble test n'est pas parfaitement répartie. On nomme ici les classe seulement par un nombre, puisque le valeur du signe astrologique est inutile. Les classes sont représentées en moyenne 8.3% ce qui suit l'aléatoire ($1/12 = 0.083$). La classe 2 correspond à 17.9% de l'ensemble, la classe 10 correspond à 11.7% de l'ensemble et la classe 9 correspond à 4.55% de l'ensemble. Un classifieur intelligent pourra démontrer ces différences.

Nous avons exploré un MLP, avec 4 couches cachées de 6 neurones, ayant une précision de 12%. Le problème est que cette précision était due au déséquilibre des données de test. On prédit fortement seulement les classe les plus populaire (2 et 10), et les sept autres classe sont prédites aucune fois. Une meilleur précision ne signifie pas un meilleur modèle. Nous avons alors opté pour le modèle moins complexe présenté au tableau9 qui est plus général. Comme voulu, ce modèle est capable de prédire en plus grande quantité la classe 2 et 10 et rarement la classe 9, tout en ayant une bonne distribution dans les autres classes. On suppose que le MLP performe mieux, car sa complexité permet de bien caractérisé le grand nombre de classes.

Les modèles d'ensemble peuvent être puissants. Au lieu d'utiliser un seul modèle, on utilise plusieurs modèles faibles qui performe bien sur différentes parti de l'espace des classes. Chaque modèle individuel ne performe pas aussi bien que lorsqu'ils sont réunis. Un autre des avantages, est qu'il est plus facile d'éviter les pièges du surentraînement. En effet, on peut augmenter la précision en ajoutant le modèle MLP définit juste en haut qui avait une précision unique de 12%. Dans le VotingClassifier, le MLP ne fait pas tendre les prédictions vers une classe et tend seulement à augmenter la précision du modèle (augmentation de 1%). Il devient un bon modèle précis pour un espace particulier des prédictions. Il aurait été intéressant d'explorer encore plus de paramètre dans le VotingClassifier, mais c'était une tâche très difficile et longue étant donné la grande quantité de paramètres. Il aurait peut-être été possible de dépassé la précision du réseau de neurone.

Bref, nos résultats concordent avec l'hypothèse posé qui mentionnait que les prédictions devraient faiblement dépasser l'aléatoire, car il n'y a pas de corrélation entre la personnalité, soit la façon d'écrire, d'un individu et son signe astrologique. Les modèles ne sont pas en mesure de capté le signe astrologique dans l'écriture d'une personne.

6 Conclusion

Les résultats finaux sont énoncés au tableau 11 où on présente la précision de chaque tâche sur l'ensemble de test. Il est évident que la classification du genre, de l'âge et du signe astrologique avaient tous trois différents niveaux de difficultés. On remarque que la tâche de prédiction de signe astrologique a été la plus difficile. Nous n'avons pas pu réfuter l'hypothèse scientifique qui affirme que le signe astrologique et la personnalité d'un individu n'est pas corrélé.

De plus, on note qu'en comparaison à une prédiction randomisé (1/2 pour le genre et 1/3 pour l'âge), le prédiction de l'âge performe mieux beaucoup. On pourrait alors conclure que l'âge est plus facile à distinguer dans la manière dont quelqu'un écrit que le genre. Ces résultats semblent logiques puisque un homme et une femme du même âge passe par des expériences similaires et discutent de sujets plus similaires. Contrairement aux différentes tranches d'âges qui discutent de sujets bien différents selon leurs âges.

Tâche	**Précision**
Genre	67.41 %
Âge	66.73 %
Signe	10.17 %

Table 11: Précision finale de chaque tâche sur l'ensemble de test.

6.1 Améliorations

Comme on voit dans le tableau 3 de la section 5.1, les mots les plus populaires des deux classes sont presque identiques et ont peu de signification. Ces résultats démontrent qu'ils pourraient être intéressant d'avoir un ensemble de *stop-words* encore plus sévère. On se débarrasserait ainsi de mots insignifiants qui n'apporte pas d'informations pour toutes les classes de chaque tâche. De cette façon, les autres tâches auraient sûrement plus de facilité à distinguer les classes. On aurait un ensemble de texte très significatif pour chaque type de personne. Il aurait aussi été possible de pousser le traitement plus profondément en retirant les mots qui ne sont pas de l'anglais, en retirant tous les URL qui n'étaient pas tous retirés ou en donnant une certaines valeurs aux emojis. On pourrait penser que les jeunes utilise plus souvent des emojis. Corriger une erreur commune de ce type de publications où un auteur répète certaine lettre du mot tel que 'lameeee' au lieu du mot 'lame' pourrait aussi avoir des conséquences positives.

Avec faibles résultats que nous avons eus, il serait intéressant d'explorer des algorithmes de classification connue comme pouvant être plus performants en général : des modèles neuronaux plus complexe Fine-tuning Bert.

Une autre approche serait aussi d'explorer nos modèles de plongement. On pourrait tenter de jouer avec *Word2Vec* et faire un plongement de mot de plus grande dimension ou d'ajouter au modèle des bi-grams ou des tri-grams. De plus, la fonction *TFIDF* contient plusieurs paramètres avec lesquels il est possible d'expérimenter. Par exemple, les paramètres *min_df* pour ignorer les mots en dessous de cette fréquence, *max_df* pour ignorer les mots trop fréquents, *max_features* pour définir la taille des *features* ou *ngram_range* pour extraire des n-grams particuliers. Bref, étudier les conséquences de ces paramètres sur la précision de chaque tâche aurait été pertinent.

References

[1] K. V. Ghag and K. Shah, "Comparative analysis of effect of stopwords removal on sentiment classification," in *2015 International Conference on Computer, Communication and Control (IC4)*, pp. 1–6, 2015.

[2] Y. Shao, S. Taylor, N. Marshall, C. Morioka, and Q. Zeng-Treitler, "Clinical text classification with word embedding features vs. bag-of-words features," in *2018 IEEE International Conference on Big Data (Big Data)*, pp. 2874–2878, IEEE, 2018.

[3] Y. Wang, Z. Zhou, S. Jin, D. Liu, and M. Lu, "Comparisons and selections of features and classifiers for short text classification," in *IOP Conference Series: Materials Science and Engineering*, vol. 261, p. 012018, IOP Publishing, 2017.

[4] J. D. Rennie, L. Shih, J. Teevan, and D. R. Karger, "Tackling the poor assumptions of naive bayes text classifiers," in *Proceedings of the 20th international conference on machine learning (ICML-03)*, pp. 616–623, 2003.