Université de Montréal

IFT6285 – Traitement automatique des langues naturelles

Devoir 6

---

par:

Eugénie Yockell
(20071932)
Bassirou Ndao
(0803389)

Date de remise: 6 novembre 2020

Pour ce devoir, différents classificateurs ont été entraîné pour différentes tâches sur des ensembles de publications de blogs où on cherche à prédire l'identification de l'auteur. Chaque cas suit la convention suivante: blog-A-B-C où A est le nombre de différents auteurs, B le nombre de publications par auteur dans l'ensemble d'entraînement et C est le nombre de publications par auteur dans l'ensemble de test. Nous avons deux ensembles de test. Le test fermé correspond à un ensemble qui contient uniquement des auteurs vu à l'entraînement, tandis que le test ouvert contient des auteurs qui n'ont pas été vu à l'entraînement.

Le prétraitement appliqué au test est important. Nous retirons les ponctuations excepté des apostrophes et tirets, plaçons tout en minuscule, regroupons les mots étant seulement des chiffres sous le label 'NOMBRE', appliquons un *stemming*. Ce prétraitement génère une grande amélioration dans le précision. Par contre, le retrait des *stop-words* faisait diminuer la précision. Ça peut être due au fait que comme chaque publication peut être plutôt petit, les *stop-words* sont importants à la contextualisation des mots.

Différents modèles de plongements de mots on été testés. En effet, le plongement de mots influence grandement les prédictions. Tout d'abord, le modèle `Word2Vec` entraîner sur le corpus complet d'entraînement des blogs. Pour créer le *embedding* d'un texte, on utilise la moyenne des plongements de chaque mots du texte. La précision sur l'identification des auteurs n'étaient pas bonne. Un autre modèle de plongement de mots testé est `TfidfVectorizer` qui convertie un texte à un vecteur représentant les valeurs de tf-idf qui pondère les mots selon leurs fréquences. Pour un cas, on entraîne un `TfidfVectorizer` en digérant chaque exemple, puis on fait le *embedding* de l'ensemble test.

Chaque tâche à des besoins différents. Pour cette raison nous avons identifié trois modèles à étudier dans chaque cas. Ces modèles ont été choisis en étudiant manuellement différents modèles comme XGBoost, régression logistique, arbre de décision, support vector machin (SVM), Naives Bayes (ComplementNB, GaussianNB, MultinomialNB), Multilayer Perceptron (MLP). De ces modèles, ComplementNB de `scikit learn`, MLP de `scikit learn` et XGBoost se démarquent. Il est alors nécessaire de déterminer les paramètres optimaux pour chacun de ces modèles pour chaque tâche. Ce rôle est rempli par la méthode `GridSearchCV` de `scikit learn`. Pour chaque modèle on ajoute au *grid search* le paramètre $max\_features$ qui représente la taille des features. C'est un paramètre important, car un trop grand nombre de features peut être difficile à traiter. Ci-dessous se trouve chaque modèle et paramètres étudier dans le *grid search*.

TfidfVectorizer ($max\_features = [1000, 2000, 3000, 4000]$)

XGBoost ($learning\_rate = [0.01, 0.3]$
$min\_child\_weight = [1, 5]$,
$subsample = [0.6, 1.0]$,
$colsample\_bytree = [0.3, 1.0]$
$max\_depth = [3, 6, 10]$)

NaiveBayes ($alpha = [0.1, 0.5, 0.8, 1]$)

MLP ($max\_features = [1000, 4000]$
$hidden\_layer\_sizes : [(50, 50)]$,
$solver = ['adam']$,
$learning\_rate = ['constant']$,
$learning\_rate\_init = [0.001, 0.01, 0.1]$,
$max\_iter = [2000]$,

Nous obtenons alors trois modèles optimisés pour chaque tâche, où on peut sélectionner le modèle le plus performant pour chaque tâche. Un tableau répertoriant la performance finale des meilleures modèles est présenté ci-dessous à la table 1.

Tâche	Modèle	Précision test fermé	Précision test ouvert
blog-10-10-10	**ComplementNB**	**74.0**	**74.0**
	MLPClassifier	62.0	62.0
	XGBClassifier	46.0	46.0
blog-10-50-10	**ComplementNB**	**69.0**	**69.0**
	MLPClassifier	63.0	63.0
	XGBClassifier	50.0	50.0
blog-10-100-10	**ComplementNB**	**78.0**	78.0
	MLPClassifier	76.0	76.0
	XGBClassifier	70.0	70.0
blog-50-10-10	**ComplementNB**	**55.4**	**51.3**
	MLPClassifier	37.4	34.6
	XGBClassifier	37.2	34.4
blog-50-50-10	**ComplementNB**	**59.2**	**54.8**
	MLPClassifier	49.8	46.1
	XGBClassifier	48.2	44.6
blog-50-100-10	**ComplementNB**	**60.0**	**55.6**
	MLPClassifier	52.4	48.5
	XGBClassifier	52.8	48.9
blog-100-10-10	**ComplementNB**	**44.2**	**40.6**
	MLPClassifier	28.3	26.0
	XGBClassifier	16.9	15.5
blog-100-50-10	**ComplementNB**	**45.9**	**42.1**
	MLPClassifier	31.2	28.6
	XGBClassifier	36.8	33.8
blog-100-100-10	**ComplementNB**	**45.6**	**41.8**
	MLPClassifier	35.5	32.6
	XGBClassifier	41.6	38.2

Table 1: Précision pour chaque tâche et chaque modèle ayant subi un *grid search*.

En premier lieu, on observe que la taille du modèle est un facteur important, influençant grandement l'exactitude des prédictions. Ce paramètre varie sous deux volets. Le nombre de classe ainsi que le nombre de représentant de cette classe. Dans notre cas l'auteur représente la classe. Le nombre d'échantillon pour chaque auteur est la représentation de la classe.

Pour les trois modèles, on remarque que plus la représentation des classes est grande mieux est la précision. Cette observation est intuitive et rejoins le fait que plus la quantité de donnée est élevée, meilleure est la généralisation du modèle. En effet on a plus de chance d'avoir vu les données test. Par contre, comme on le voit à table 1, plus il y a grand nombre de classes à l'entraînement, plus le modèle performe médiocrement. Intuitivement, cela peut être due à plusieurs facteurs. Il peut y avoir un effet de dilutions des probabilités de chaque classes lors des prédictions. Aussi, plus on voit des auteurs, plus la probabilité de trouver des similitudes entre eux est plus grande et la tâche de les différencier devient plus difficile.

La précision des prédictions faites sur les données de test ouvert est beaucoup plus faible que celles faites sur les données de test fermée. Cette précision baisse au fur et à mesure que la taille de l'entraînement augmente. Cette remarque est à mettre en relation avec le fait que l'on tente de faire des prédictions de données qui non jamais été en rencontré en entraînement.

De plus, le modèle de `ComplementNB` est systématiquement le modèle le plus performant. On étudie les paramètres optimaux déterminé avec `GridSearchCV` à table 2. On remarque que plus le nombre de classe augmente, plus le paramètre *alpha* augmente. Ce paramètre représente le lissage de Laplace utiliser dans l'algorithme Naïve Bayes. Il permet de donner une probabilité non-nulle aux mots inconnus d'une classe quelconque. Dans les cas où on augmente le nombre d'auteur et le nombre de publications, il est nécessaire d'avoir un plus grand *alpha* pour avoir des probabilités non-nulle pertinentes. Le nombre de $features$ augmente aussi avec le nombre de classes. Ce résultat semble logique puisqu'il se peut que le modèle nécessite plus de *features* pour chaque classe, car il y a un plus grand nombre de classes. On cherche alors à augmenter la complexité de chaque échantillon.

Tâche	**alpha** `ComplementNB`	**max_features** `TfidfVectorizer`
blog-10-10-10	0.1	2000
blog-10-50-10	0.1	3000
blog-10-100-10	0.1	4000
blog-50-10-10	0.1	4000
blog-50-50-10	0.5	4000
blog-50-100-10	0.5	4000
blog-100-10-10	0.5	4000
blog-100-50-10	1.0	4000
blog-100-100-10	1.0	4000

Table 2: Paramètres optimaux du modèle le plus performant, `ComplementNB`, pour chaque tâche.

Dans le cas d'un ensemble de test ouvert pour le *blog-quiz*, on veut assigner la valeur 'AUTRE' aux auteurs inconnus du modèle. Pour s'y faire, on étudie la probabilité de chaque classe pour un échantillon de l'ensemble test. Cette information peut être extraite de la fonction `predict_proba` de *scikit-learn*. Au-dessous d'un certain seuil de probabilité prédéfini on considère que le modèle à trop de difficulté à prédire la classe et alors le texte provient d'un auteur inconnu. Dans le cas de *blog-quiz*, on utilise le modèle `ComplementNB(alpha = 0.5)` avec un nombre de $features$ de taille 4000. Ça correspond au modèle étudié pour le cas *blog-50-50-10*, car c'est un cas qui ressemble au *blog-quiz* avec 20 auteurs et 50 publications chacun. Nous avons alors étudier un seuil intéressant pour le *blog-50-50-10*. On remarque que la probabilité maximale (probabilité de la classe prédite) tourne toujours autour de la probabilité aléatoire 1/50, mais on sait que la probabilité maximale ne sera jamais sous la probabilité aléatoire. Nous avons étudié un seuil tel que si la probabilité de la classe est sous $1/20+\epsilon$, alors on considère que c'est un auteur inconnu. Les meilleurs résultats étaient générés par $\epsilon = 0.0011$.

En conclusion, les classificateurs ont de la difficulté à prédire les auteurs particulièrement lorsqu'il y a plusieurs auteurs. Il est pertinent de s'attarder sur les méthodes prenant en compte des classes non rencontrées en entraînement. La librairie *scikit-learn* propose une méthodes dites de *novelty detection* que l'on a essayé sans avoir des résultats concluants. Elle permettrait de détecté si une donnée test s'éloigne assez de celles vu en entraînement. Dans un tel cas, elle serait marquée comme 'AUTRE'.