Traitement automatique de Langue Naturelle

TD #1

5IABD2

AZERAD, Ariel

MURO, Daniel Alexander

Tout au début et sans aucune modification, l'accuracy du modèle était de 91.3%. Notre objectif était, alors, d'optimiser le résultat en ayant une valeur d'accuracy plus élevée.

Etant donné que le dataset n'a pas des valeurs manquantes, nous avons passé directement aux pré-traitements sur la colonne video_name :

- Suppression de stopwords : apporte environ 0.15% d'incrément sur l'accuracy finale.
- Suppression des majuscules sur tous les mots : si s'applique comme seul prétraitement sur le dataset, son effet c'est de diminuer l'accuracy finale, ce qui n'est pas du tout étonnant tenant en compte que ce que l'on essaie de prédire c'est la présence des noms propres, écrit toujours avec une majuscule. Cependant, si s'applique avec des autres prétraitements, permets d'améliorer l'accuracy d'un 0,06%.
- Suppression des chiffres : pareil que pour les majuscules.
- Lemmatisation : n'a pas apporté aucune amélioration sur le résultat.
- Stemming : a apporté un incrément de 0.02% sur l'accuracy.
- Suppression des signes de ponctuation : 0.01% d'améliorations sur l'accuracy.
- Mots peu significatifs : nous avons décidé de ne pas ignorer les mots qui apparaissent peu dans le dataset, car il sera probable que, en faisant cela, on ne prendra pas en compte les noms des comédiens pour l'entraînement du modèle.
- N-grams : nous avons testé (1, 2), (1, 3) et même (1, 4), mais nous avions à chaque fois un résultat moins bon que celui obtenu sans prendre en compte des groupes de mots.

Cependant, nous avons pu constater que la combinaison de plusieurs prétraitements provoque des effets plutôt intéressants sur le dataset :

Pré-traitement	**% accuracy**	**% amélioration constaté**
Aucun	91,9439	---
Stopwords	92,0943	0,1504
Lowercase + Stopwords	92,0340	0,0901
Stemming	91,9649	0,0210
Ponctuation	91,9539	0,0100
Chiffres	91,9239	-0,0200
Stopwords + Stemming	92,5469	0,6030
Stopwords + Lowercase + Stemming	92,4563	0,5124
Stopwords + Stemming + Ponctuation	92,5367	0,5928
Stopwords + Lowercase + Stemming + Ponctuation	92,4665	0,5226
Stopwords + Stemming + Chiffres	92,5063	0,5624
Stopwords + Lowercase + Stemming + Chiffres	92,5267	0,5828
Stopwords + Stemming + Ponctuation + Chiffres	92,5263	0,5824
Stopwords + Lowercase + Stemming + Ponctuation + Chiffres	92,5867	0,6428

Une fois définies les étapes de prétraitement à appliquer au texte, nous sommes passés au choix de l'algorithme utilisé pour la vectorisation. Nous avons incrémenté à 10 la quantité de lots utilisés dans la fonction de cross validation, et nous avons testé les quatre algorithmes de vectorisation de scikit-learn.

Afin d'assurer le résultat des tests, nous avons exécuté chaque algorithme un total de dix fois, et calculé l'accuracy moyenne de chacun d'entre eux :

Algorithme de vectorisation	**CountVectorizer()**	**HashingVectorizer()**	**CountVectorizer() + TfidfTransformer()**	**TfidfVectorizer()**
% accuracy	92,3939	82,6808	92,1929	92,5969
	92,2929	82,6838	92,0929	92,2959
	92,4939	82,4787	91,9929	91,7949
	92,5939	83,2828	92,2949	92,1959
	92,3929	82,0808	91,9929	92,8949
	92,2939	81,9818	92,3969	91,2929
	92,5939	82,0808	92,5959	92,2969
	92,6939	82,3828	92,3959	92,0939
	92,4939	82,4797	91,5929	91,6919
% accuracy moyenne	**92,4939**	**82,4818**	**92,0929**	**91,9919**

Ensuite, nous avons testé plusieurs modèles de classification :

Modèle	**% accuracy moyenne**
RandomForestClassifier	92,4939
AdaBoostClassifier	94,0939
DecisionTreeClassifier	93,7939
SGDClassifier	94,0989
MLPClassifier	93,0939
SVC	92,5939
GradientBoostingClassifier	94,1949

Finalement, ayant sélectionné les prétraitements, l'algorithme de vectorisation et le modèle de classification, nous avons ajusté certains paramètres du modèle, afin d'améliorer son résultat :

- n_estimators : En général, un grand nombre d'étapes de boosting entraîne des meilleures performances mais, après plusieurs tests, nous n'avons constaté aucune amélioration en utilisant plus de 150.
- criterion : squared_error nous a donné des meilleurs résultats que friedman_mse.

En résumé, nous avons constaté les meilleurs résultats en utilisant :

Pour le prétraitement des textes :

- Suppression des stopwords
- Suppression de majuscules
- Stemming
- Suppression des signes de Ponctuation
- Suppression des chiffres

Pour l'algorithme de vectorisation :

- CountVectorizer()

Pour le modèle de classification :

- GradientBoostingClassifier, avec 150 n_estimators et la fonction squared_error pour mesurer la qualité d'un fractionnement

Avec ces paramètres, nous avons réussi à obtenir une accuracy finale de 94,49%.