École Polytechnique de Montréal
Département Génie Informatique et Génie Logiciel
INF8460 – Traitement automatique de la langue naturelle

Objectifs d'apprentissage

- Effectuer du pré-traitement de corpus
- Coder un modèle simple BOW et l'utiliser pour de la classification
- Evaluer les performances du modèle

Logiciels

Ce TP utilise la librairie NLTK, ScikitLearn(nom abrégé en sklearn), numpy, matplotlib et Jupyter notebook. Ces librairies ainsi que toutes les librairies de base provenant de python sont permises.

Il est conseillé d'installer Anaconda avec Python 3.10.6

Corpus et code

Jeu de données : le jeu de données contient des textes du HuffPost dans l'attribut *description* ainsi que leur catégorie.
Un exemple extrait du jeu de données :

Le zip du TP contient :
- Un jeu de données : dataset.zip
- Le squelette du notebook qui doit être complété

Travail à faire

1 Pré-traitement et analyse statistique de corpus (16 points)

1.1 (*5 points***)** Implémentez la fonction pre-process () qui prend en entrée une phrase et retourne une liste de jetons prétraités. La fonction doit effectuer les traitements suivants :
- Supprimer les signes de ponctuations
- Transformer les lettres majuscules en lettres minuscules
- Transformer la chaîne de caractères en liste de jetons (segmentation)
- Supprimer les mots qui sont connus comme des « stop words ». Les stop words peuvent être obtenus avec from nltk.corpus import stopwords ;
- Effectuer une racinisation (stemming)

Appliquez le prétraitement pour tous les exemples dans l'ensemble d'entrainement et de test

1.2 (*4 points***)** Afficher les informations suivantes, avec graphique s'il a lieu, pour l'ensemble d'entraînement:

- Le nombre de jetons des descriptions en moyenne
- Le nombre d'exemples par catégorie (graphique à barre)

- Le nombre moyen de jetons des descriptions en moyenne par catégorie (graphique à barre)
- Le nombre maximal et le nombre minimal de jetons par catégorie

1.3 (*3 points***)** Implémentez la fonction build_voc qui extrait votre vocabulaire de l'ensemble d'entrainement et crée une liste des jetons qui ont une fréquence d'occurrence de 5 au moins.

1.4 (*1.5 points***)** Vous devez créer une fonction get_top_vocab(vocab, n) qui retourne les n jetons les plus fréquents et les affiche.

1.5 (*0.5 points***)** Affiches les 10 jetons les plus fréquents du vocabulaire de l'ensemble d'entraînement

1.6 (*1.5 points***)** Vous devez créer une fonction get_top_vocab_per_cls(dataset, n, cls) qui retourne les n jetons les plus fréquents de la classe cls et les affiche.

1.7 (*0.5 points***)** Affichez les 10 jetons les plus fréquents de la classe *Business* du vocabulaire de l'ensemble d'entraînement.

2 Représentation avec un modèle sac de mots (15 points)

Implémentez sans l'aide de librairies (à l'exception de numpy) les sacs de mots suivants (les formules sont indiquées dans le notebook). Vos modèles sac de mot doivent se baser sur le vocabulaire obtenu en 1).

2.1 (*7.5 points***)** Implémentez un modèle sac de mots pondéré avec des fréquences d'occurrence.

2.2 (*7.5 points***)** Implémentez un modèle sac de mots pondéré avec TF-IDF.

3 Classification automatique et évaluation (5 points)

3.1 (*3 points***)** Avec les 2 sac de mots en 2), entrainez un modèle Bayésien naïf multi classes avec ScikitLearn (MultinomialNB). Prédisez les classes de l'ensemble de test et reportez vos résultats dans une même table avec les métriques suivantes : Accuracy et pour chaque classe, la précision, le rappel et le F1 score.

3.2 (*2 points***)** Discutez de vos résultats en répondant aux pistes de réflexion abordées dans le notebook.

3.2.1 Quel modèle de sac de mots a la meilleure performance ? Qu'est-ce qui explique cette différence ?

3.2.2 Pourquoi le classificateur a-t-il une moins bonne performance sur la classe TECH? Soyez précis et détaillez votre réponse.

4. Optimisation des modèles (6 points)

Optimisez votre modèle Bayésien en trouvant les meilleurs paramètres à l'aide de la validation croisée de la manière suivante :

4.1 (*2 points***)** En utilisant votre sac de mot TF-IDF, utilisez GridSearchCV (sklearn) afin d'optimiser votre modèle Bayésien.

4.2 (*2 points***)** Utilisez maintenant la vectorisation TF-IDF de sklearn et combinez-la avec le modèle Bayésien afin d'optimiser les paramètres avec GridSearchCV.

4.3 (*2 points***)** Discutez de vos résultats en répondant aux pistes de réflexion abordées dans le notebook.

4.3.1 Quels paramètres avez-vous inclus dans le GridSearch et comment affectent-t'ils le modèle ? (Analysez chacun de vos paramètres)

4.3.2 Avez-vous obtenu un meilleur score en utilisant le sac de mot TF-IDF provenant de Scikit-Learn ? Pourquoi selon vous ?

4.3.3 Comment TF IDF est-il calculé par la librarie Scikit-Learn par rapport à la formule dans ce notebook? Pourquoi ce changement ? Indice: Regardez la documentation de TfidfTransformer.

LIVRABLES

Vous devez remettre sur Moodle un zip contenant les fichiers suivants :

1- *Le code* : Vous devez compléter le squelette inf8460_tp1.ipynb sous le nom equipe_i_inf8460_TP1.ipynb (i = votre numéro d'équipe). Indiquez vos noms et matricules au début du notebook. Ce notebook reprend les différentes questions, et doit contenir les fonctionnalités requises avec des commentaires appropriés. Le code doit être exécutable sans erreur et accompagné de commentaires appropriés de manière à expliquer les différentes fonctions. Les critères de qualité tels que la lisibilité du code et des commentaires sont importants. Tout votre code et vos résultats doivent être exécutables et reproductibles ;

2- Un fichier html représentant votre notebook complètement exécuté sous format html

3- Un document *contributions.txt :* Décrivez brièvement le pourcentage de contribution et la contribution de chaque membre de l'équipe. Tous les membres sont censés contribuer au développement. Bien que chaque membre puisse effectuer différentes tâches, vous devez vous efforcer d'obtenir une répartition égale du travail.

EVALUATION

Votre TP sera évalué selon les critères suivants :

1. Exécution correcte du code
2. Performance attendue du modèle
3. Organisation du notebook
4. Qualité du code (noms significatifs, structure, performance, gestion d'exception, etc.)
5. Commentaires clairs et informatifs
6. Réponses correctes aux questions de réflexion