Résumé des instructions pour le rendu du TP Fouille de textes

S. Aït-Mokhtar

Date limite: DCISS et WIC **: à préciser.** SSD **: à préciser.**

Rendu à envoyer à: sacours@outlook.com en précisant le nom des auteurs (1 ou 2, si en binôme)

1 Introduction

Le thème du projet est la fouille d'opinions dans les textes. L'objectif est l'implémentation d'un classifieur qui classifie les phrases d'avis sur des restaurants en 3 classes possibles : « positive » si la phrase exprime une opinion ou un sentiment positif, « negative » si la phrase exprime un sentiment négatif, ou « neutral » si elle ne contient pas d'opinion, ou si elle exprime des sentiments mixtes (positifs et négatifs). Le classifieur doit produire seule classe par phrase.

Exemples de phrases correctement classifiées :

positive	Un restau au bord de l'eau c'est sympa.
negative	Un tartare de saumon avec trop d'oignons.
neutral	C'est presque par hasard que nous avons dîné dans ce restaurant.
neutral	Les produits sont bons mais la cuisine reste simple par rapport aux tarifs.

Le rendu sera évalué en prenant en compte les éléments suivants donnés par ordre d'importance :

- Sa précision (pourcentage de phrase classifiées correctement / nombre de phrase)
- Son efficacité computationnelle (vitesse d'entraînement et de prédiction, mémoire requise).

2 A télécharger

Dans le répertoire partagé du cours (https://1drv.ms/u/s!Aksy8Pc5f6z8gocE0yYSaIvjRaaX3Q?e=XdGTXB), le sous-répertoire **TP** contient des éléments à télécharger :

- Ce document (**TP_Instructions.docx**) qui résume ce que je vous ai présenté lors des deux dernières séances en salle informatique.
- Un sous-répertoire « **data** » qui contient les données d'entraînement et de développement. Les fichiers de données fournis (frdataset1_train.csv et frdataset1_dev.csv) ont le format des exemples ci-dessus (section Introduction) : chaque ligne contient une phrase avec sa classe correcte, les deux champs étant séparés par un caractère de tabulation.
- Un sous-répertoire **src** qui contient des fichiers de code sur lesquels votre contribution sera basée.

- Un sous-répertoire **resources** qui contient un fichier de plongements de mots (word embeddings de type word2vec) pour le français que vous pouvez éventuellement exploiter (frWac_non_lem_no_postag_no_phrase_200_skip_cut100.bin). Si votre classifieur utilise d'autres ressources (par exemple liste de mots grammaticaux (stop words), ou liste de mots de polarité), c'est dans ce sous-répertoire **resources** qu'il faudrait les mettre.

3 Comment procéder

1. Vous pouvez travailler soit individuellement soit en binôme.
2. Installer python (3.7.x) via anaconda ou miniconda (donc scikit-learn et pandas inclus), keras (=2.2.4), spacy (= 2.1.8) et gensim (= 3.8.0).
3. Télécharger le contenu du sous-répertoire TP (voir section « A télécharger » ci-dessus)
4. Le sous-répertoire **src** contient 3 fichiers de code importants : tester.py, datatools.py, classifier_bow.py (le 4ème, classifier.py est là pour montrer les deux méthodes nécessaires à avoir dans votre classe Classifier : train() et test())
5. Vous ne devez pas modifier **datatools.py**, il contient du code pour lire et charger en mémoire les données.
6. Vous ne devez pas modifier non plus **tester.py**, il contient du code pour lancer l'exécution du projet : l'entraînement de votre classifieur sur les données d'entraînement, puis l'évaluation de sa précision sur les données de développement.
7. Le fichier **classifier_bow.py** contient un exemple basique de classifieur.
8. Pour exécuter le projet avec cet exemple basique de classifieur, lancez un terminal de commande, allez dans le sous-répertoire **src** et tapez : **python tester.py**
9. L'exécution doit aboutir à l'entraînement du modèle et son évaluation. A la fin s'affichent la précision moyenne du classifieur sur les données de développement (Mean Dev Acc.) ainsi que le temps d'exécution. « Mean Test Acc. » est la précision sur les données de test : vous pouvez ignorer cette valeur de votre côté (-1), car vous ne disposez pas des données de test.
10. **Ne modifiez pas « tester.py » ni datatools.py** svp ! Votre classifieur doit fonctionner sans erreur en tapant la commande « python tester.py » (voir le point 8 ci-dessus)
11. **Votre travail consiste à modifier le fichier classifier_bow.py** (ou même le réécrire totalement, si vous le souhaitez) qui contient le classifieur de base, afin d'implémenter un meilleur classifieur, et d'obtenir une meilleure précision. La précision sur les données dev du classifieur de base, avec un nombre max de traits (features) limité à 8000, est en moyenne de 63,75%. L'objectif est d'obtenir le meilleur score possible en modifiant ce classifieur de base ou en le réécrivant totalement.
12. Pas de contraintes sur les modifications que vous pouvez apporter au classifieur de base, si ce n'est que l'exécution du projet doit fonctionner (dans le sous-répertoire src avec la commande python tester.py, voir le point 8) sans avoir à installer d'autres modules que ceux listés au point 2.
13. Parmi les modifications possibles que vous pouvez réaliser :
    a. Modifier les valeurs des hyper-paramètres : nombre max de traits, nombre d'itération (epochs), la taille de « batch », etc.
    b. Modifier la représentation des phrases : uni/bi/tri-grammes, tf/idf, la tokénization, la normalisation (casse, lemmatisation), la suppression des mots grammaticaux (stop

words), le rajout d'une représentation de mots polarisés (mots négatifs/positifs), exploitation des relations syntaxiques produites par sPacy, etc.

c. Utilisation des plongements lexicaux (word embeddings)
d. Modification de l'architecture du réseau de neurones (ajout de couches intermédiaires), utilisation de réseaux récurrents comme les LSTMs, etc.

14. Si vous décidez d'utiliser d'autres ressources que les plongements de mots (déjà fournis dans le sous-répertoire « resources »), vous devez les mettre dans ce même sous-répertoire. Dans le code de votre classifieur, vous y accèderez avec des chemins relatifs pour que l'exécution du projet (voir point 8) puisse se faire quel que soit l'endroit où les sous-répertoires **src**, **resources** et **data** sont mis.
15. Si vous avez besoin de créer de nouveaux fichiers de code, mettez-les dans src (toujours en vous assurant que l'exécution du projet, voir point 8, se fait sans erreur).
16. Le **contenu exact du rendu** est décrit dans la section suivante.
17. Le rendu doit être sous le format d'un répertoire compressé au format zip (pas de gz, bz ou autre format svp).
18. Le nom du fichier compressé doit contenir le nom de famille de l'étudiant auteur du rendu, ou les deux noms de familles des deux étudiants auteurs du rendu.
19. Le fichier du rendu ne doit pas excéder 5 Mo (si vous utilisez le fichier des word embeddings que j'ai mis dans **resources**, vous devez l'enlevez avant de compresser le répertoire du rendu).

4 Contenu du rendu

Lorsque le fichier du rendu est décompressé, le répertoire racine doit contenir les éléments suivants :

Elément	**description**
README.txt	Un fichier texte pur qui doit contenir les information suivantes: 1. Le(s) nom(s) complet(s) des étudiant(s) auteur(s) du rendu (**max=2**) 2. Un ou deux paragraphes décrivant le classifieur (hyper-paramètres, type de représentation, type d'architecture, ressources etc.) 3. La précision moyenne que vous obtenez sur les données de dev.
resources	(optionel): sous-répertoire contenant les ressources additionnelles que vous utilisez (en dehors du fichier des word embeddings distribué)
src	Sous-répertoire contenant TOUS les fichiers de code python nécessaires à l'exécution du projet avec la commande « python tester.py »)