Travail pratique # 2
Classification de textes et modèle de langue avec des réseaux de neurones

Automne 2022

Proposé par Luc Lamontagne

OBJECTIF :

- Utiliser différents plongements de mots préentraînés (*pretrained word embeddings*)
- Comprendre comment programmer et entraîner des modèles *feedforward* (MLP) et des modèles récurrents avec PyTorch.
- Mener des expérimentations avec les réseaux de neurones pour classifier des textes et choisir la bonne version d'un proverbe. Évaluer les performances de ces modèles sur des jeux de données.

INSTRUCTIONS :

- Matériel disponible le 3 novembre 2022.
- Ce travail sera noté sur 100 et vaut 20% de la note du cours.
- Rapport et code : À remettre le 25 novembre sur MonPortail, le tout compressé en format Zip.
- Format de la remise : Soit des *notebooks* Jupyter bien documentés (html + notebook) ou un rapport PDF accompagné de code Python. L'un ou l'autre.
- Références : Chapitres 6, 7 et 9 de la 3e édition du livre de Jurafsky et Martin.
- Seul langage de programmation autorisé: Python 3.
- Ressources disponibles sur le site du cours:
    - Des fichiers de textes pour mener vos expérimentations.
    - Pour ce travail, aucun fichier de démarrage n'est fourni.
    - Vous pouvez réutiliser des parties des notebooks Jupyter sur le site du cours.
- Librairies autorisées :
    - Réseaux de neurones : *PyTorch*. Aucun autre *framework* d'apprentissage profond autorisé (par ex. *TensorFlow*). On suggère l'utilisation de Poutyne (optionnel).
    - Tokenisation : Choix libre. On recommande Spacy.
    - Normalisation de textes : aucune normalisation (sauf la tokenisation de mots).
    - Manipulation de fichiers de données : Peuvent être manipulés avec du code Python, Scikit-learn ou NumPy (si nécessaire).

TÂCHE 1 – CLASSIFICATION DE QUESTIONS – RÉSEAU *FEEDFORWARD* (MLP) ET PLONGEMENTS DE MOTS

À partir du fichier d'entraînement *data/questions-train.txt* et du fichier de test *data/questions-test.txt*, entraînez des modèles de réseau de neurones de type *feedforward* multicouche (MLP) pour déterminer la classe d'une question. Par exemple, la question

How tall is the Sears Building ?

correspond à la classe de quantité numérique (*QUANTITY*).

Pour cette tâche, on vous demande de mener les 2 analyses suivantes :

Tâche 1.a – FastText vs. Word2Vec

Comparez l'efficacité des modèles avec les plongements de mots préentraînés de FastText et ceux de Word2Vec (par l'entremise de Gensim).

Présentez les résultats clairement à l'aide de figures. Analysez les résultats. Expliquez les fonctions particulières qui ont été implémentées pour pouvoir utiliser les plongements de FastText et de Word2Vec.

Tâche 1.b – Impact de la taille de la couche cachée du réseau *feedforward*

Évaluez les performances des modèles en fonction de la taille de la couche cachée du réseau MLP. Expliquer clairement dans votre notebook/rapport comment vous avez procédé. Formulez des recommandations précises sur la configuration de réseau qui vous semble la plus performante.

Note : Vous n'êtes pas tenu d'évaluer la performance des modèles en fonction d'un nombre variable de couches cachées.

TÂCHE 2 – COMPLÉTEZ LE PROVERBE AVEC UN RÉSEAU LSTM ET DES PLONGEMENTS Spacy

On reprend la tâche du premier travail qui consiste à compléter des proverbes. Pour cette remise, vous allez construire un réseau de neurones LSTM qui vous aidera à compléter des proverbes incomplets en ajoutant un mot au bon endroit. Par exemple, étant donné la phrase incomplète:

*aide-toi, le *** t'aidera*

et la liste de mots candidats [*médecin*, *ciel*, *professeur*, *whisky*], on souhaite que votre modèle LSTM retourne la séquence la plus probable qui est

aide-toi, le ciel t'aidera

Entraînement du réseau LSTM : entraînez le réseau récurrent LSTM comme un modèle de langue neuronal à partir du fichier *data/proverbes.txt* (le même que celui du premier travail). Quelques précisions :

- Chaque proverbe correspond à 1 ligne du fichier. Aucune normalisation des mots.
- On utilise les plongements français de Spacy.
- Vous devez travailler avec un vocabulaire fermé. À vous de voir ce que cela implique.
- On mène l'entraînement avec une stratégie de type *teacher forcing*.
- Pour plus de détails sur les modèles de langue avec des réseaux récurrents, voir la section 9.3 de Jurafsky et Martin.

Compléter les proverbes:

- Les consignes sont similaires à celles du premier travail pratique. Compléter un proverbe consiste à remplacer les étoiles (un mot masqué) par l'un des mots de la liste de choix.
- Une des difficultés de ce problème est de déterminer comment estimer la probabilité d'un proverbe complété. À vous de voir comment cela peut être fait avec un LSTM.
- Utilisez le fichier de test *data/test_proverbes.txt* pour évaluer la performance du modèle LSTM. Comme ce fichier ne contient pas les bonnes réponses, on vous recommande de créer un autre fichier (avec les solutions) qui vous permettra de faire une évaluation automatique de la performance du modèle. Merci d'inclure ce nouveau fichier dans votre remise.
- Analyses à inclure dans votre *notebook*/rapport :

- Décrivez clairement comment vous avez procédé pour entraîner le modèle LSTM et expliquez précisément comment ce modèle est utilisé pour choisir le bon mot qui complète un proverbe.
- Présentez les résultats obtenus.
- Est-ce que le modèle capture bien le langage utilisé dans les proverbes ?
- Comparer les résultats avec ceux obtenus dans le travail #1 avec des modèles de langue N-grammes.

ÉVALUATION DU TRAVAIL

T1 – Classification de questions – MLP, FastText, Word2Vec et couche cachée	50%
T2 – Compléter le proverbe – modèle de langue neuronal LSTM et Spacy	40%
Qualité des *notebooks* ou du rapport	10%