HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts Western Switzerland

Cours TAL – Labo 7 : classification de textes (pour la désambiguïsation lexicale)

Distribué le mardi 14 mai 2019

Objectif et plan

L'objectif de ce laboratoire est d'utiliser des méthodes d'apprentissage supervisé pour classifier des occurrences du mot *interest* selon leur sens : c'est la même tâche, avec les mêmes données, que le labo 6. Pour classifier les occurrences, on considère leur contexte (mots voisins), et on applique l'approche bayésienne vue en cours : on entraîne un classifieur à six classes (les six sens de *interest* annotés de 1 à 6) sur une partie des données et on le teste sur la partie restante.

Dans ce laboratoire, on explore deux façons différentes de coder les traits (*features*) pour cette tâche. Dans les deux cas, on entraînera un *NaiveBayesClassifier* fourni par NLTK.[1] Les deux façons sont :

1. Constituer un vocabulaire des mots qui apparaissent dans le voisinage de *interest* et définir ces mots comme traits. Pour chaque occurrence de *interest*, on extrait la valeur de ces traits sous la forme {('rate' : True), ('in' : False), … } et on ajoute la classe (de 1 à 6).

2. Si $word_{-1}$ est le mot précédant l'occurrence de *interest*, on définit comme traits $word_{-n}$, …, $word_{-2}$, $word_{-1}$, $word_{+1}$, $word_{+2}$, …, $word_{+n}$ (une fenêtre de taille 2*n* autour de *interest*). Les valeurs possibles de ces traits sont cette fois-ci les mots observés, ou 'NONE' si la fenêtre dépasse les limites de la phrase. Pour chaque occurrence de *interest*, on extrait la valeur de ces traits sous la forme {('word-1' : 'his'), ('word+1' : 'in'), … } et on ajoute la classe (de 1 à 6).

Dans les deux cas, il faut diviser les 2368 occurrences de *interest* en un jeu d'entraînement et un jeu de test, en respectant la proportion initiale de chaque sens. Puis on entraîne un classifieur bayésien naïf en respectant le format de données indiqué par NLTK[2], et on teste la performance du classifieur entraîné. L'objectif est de trouver les paramètres qui conduisent aux meilleurs scores de WSD.

Merci d'envoyer votre notebook Jupyter par email au professeur avant le **lundi 27 mai à 23h59**.

Étapes proposées

[1] Une gamme beaucoup plus variée de classifieurs, et des outils d'extraction de traits et de préparation des données, sont fournies par la bibliothèque *Scikit-Learn*.
[2] Chapitre 6, section 1.1-1.3 (https://www.nltk.org/book/ch06.html).

A. Traits lexicaux : présence ou absence de mots dans le voisinage de *interest*

1. Le fichier de données se trouve à http://www.d.umn.edu/~tpederse/data.html – chercher « *interest* » vers la fin de la page, et prendre le fichier marqué comme « original format without POS tags » (le même qu'au labo 6). Lire le fichier et générer une liste de listes de mots (une liste par phrase) appelée *tokenized_sentences*.

2. Définir une variable *window_size*, par exemple égale à 3 (on la fera varier plus tard), et une liste vide de mots *word_list*. Parcourir les *tokenized_sentences* et pour chaque phrase ajouter les mots voisins de *interest* (i.e. situés à une distance inférieure ou égale à *window_size*) dans la liste de mots *word_list*. Combien de mots contient celle-ci à la fin ? *Tokens* ou *types* ?

3. À l'aide d'un objet NLTK de type FreqDist, sélectionner parmi les mots de *word_list* les *N* plus fréquents, dans une nouvelle liste appelée *vocabulary* (p.ex. *N* = 500, mais on le fera varier). Affichez les 50 mots les plus fréquents. Est-ce une bonne idée d'enlever les *stopwords* ?

4. Parcourir à nouveau les *tokenized_sentences* et pour chaque phrase créer un couple (*dictionnaire*, *sens*), où le *dictionnaire* regroupe les traits et leurs valeurs, et le *sens* est un nombre de 1 à 6 indiquant le sens de *interest*. Les couples pour toutes les phrases seront rassemblés dans une liste appelée *feature_sets*.

    - Prendre modèle sur https://www.nltk.org/book/ch06.html (début du 1.2)
    - Pour le dictionnaire, il faut créer un trait pour chaque mot de *vocabulary*, et examiner si ce mot est présent dans une fenêtre de taille *window_size* autour de l'occurrence de *interest* : si oui, le trait est True, sinon il est False. Par exemple, on aboutit à : {'contains(the)': False, 'contains(,)': True, 'contains(rates)': True, …}.
    - Ajouter aussi le trait 'word0' qui note si l'occurrence est *interest* ou *interests* (pluriel).
    - Combien d'occurrences pour chaque sens de *interest* y a-t-il dans *feature_sets* ?

5. Diviser les données de *feature_sets* en deux sous-ensembles : l'un comportant 80% des données est le *train_set*, et l'autre (20%) est le *test_set*. Attention, il faut respecter deux conditions :

    - Chaque sens doit être présent dans les mêmes proportions dans *train_set* et dans *test_set* (donc il faut faire la division de manière séparée pour chaque sens).
    - Mélanger avec *shuffle()* les occurrences avant de prendre les 80% premières pour le *train_set* et les 20% restantes dans le *test_set.*

6. Entraîner un classifieur de type *NaiveBayesClassifier* de NLTK sur *train_set*, puis le tester sur les données de *test_set*. Quelle est la précision (*accuracy*) atteinte ?

7. Adapter le code précédent pour effectuer plusieurs divisions des données en *train* et *test* (par exemple 10), et calculer la moyenne des scores obtenus. Comment se compare cette moyenne avec votre premier résultat ?

8. Cherchez les meilleurs paramètres pour la taille de la fenêtre (p.ex. 1, 3, 5, 7, 11) et la taille du vocabulaire (50, 100, 200, 500, 1000 mots). Combien d'expériences faut-il exécuter ? Quelle est la meilleure combinaison *fenêtre* x *vocabulaire* et quel est le score moyen obtenu ?

B. Traits lexicaux positionnels : valeurs des mots précédant/suivant *interest*

Pour cette deuxième partie, on réutilisera beaucoup d'éléments de la première. Seule la nature des traits utilisés et leur extraction vont changer.

1. Partir de la liste de listes de mots (une liste par phrase) précédente, appelée *tokenized_sentences*.
2. Définir une variable *window_size2*, par exemple égale à 3 (on la fera varier plus tard).
3. Parcourir les *tokenized_sentences* et pour chaque phrase créer un couple (*dictionnaire*, *sens*), où le *dictionnaire* regroupe les traits et leurs valeurs, et le *sens* est un nombre de 1 à 6 indiquant le sens de *interest*. Les couples pour toutes les phrases seront rassemblés dans une nouvelle liste appelée *feature_sets2*.
    - Pour le dictionnaire de traits, il faut cette fois-ci créer un trait pour chaque position relative par rapport à *interest*, donc 'word-1', 'word+1', etc. (jusqu'à *window_size2*). La valeur du trait sera le mot trouvé à cette position, ou 'NONE' si on sort de la phrase. Par exemple {('word-1' : 'his'), ('word+1' : 'in'), … }.
    - Ajouter aussi le trait 'word0' qui note si l'occurrence est *interest* ou *interests* (pluriel).
4. Diviser les données de *feature_sets2* en deux sous-ensembles (80%/20%) appelés *train_set2* et *test_set2* avec la même procédure qu'à la partie A.
5. Entraîner un classifieur de type *NaiveBayesClassifier* de NLTK sur *train_set2*, puis le tester sur les données de *test_set2*. Quelle est la précision (*accuracy*) atteinte ?
6. Effectuer plusieurs divisions des données en *train* et *test* (par exemple 10), et calculer la moyenne des scores obtenus. Comment se compare cette moyenne avec votre premier résultat ?
7. Cherchez les meilleurs paramètres pour la taille de la fenêtre (p.ex. entre 1 et 15). Quelle est la meilleure valeur et quel est le score moyen obtenu ?
8. Quelle est le meilleur score obtenu entre (A) et (B) ?
9. *Thème de réflexion facultatif* : les différences des scores sont-elles statistiquement significatives ?