Application à la classification : l'analyse d'opinions

Rendu attendu

Le compte-rendu du TP est à déposer sur le site pédagogique (rubrique Rendus TP text mining) sous la forme d'un notebook incluant une discussion sur les résultats obtenus (versions .pynb et **.pdf**) avant le vendredi 16 décembre minuit. Attention, n'oubliez pas de déposer le .pdf.

Objectif

L'objectif de ce TP est d'implémenter un algorithme de classification de critiques de films en fonction de la polarité des opinions exprimées (positif/négatif). On parle en anglais de "sentiment analysis". L'algorithme utilisé sera le classifieur bayésien naïf (*Naive Bayes*). Le langage à utiliser est Python.

Matériel et documentation

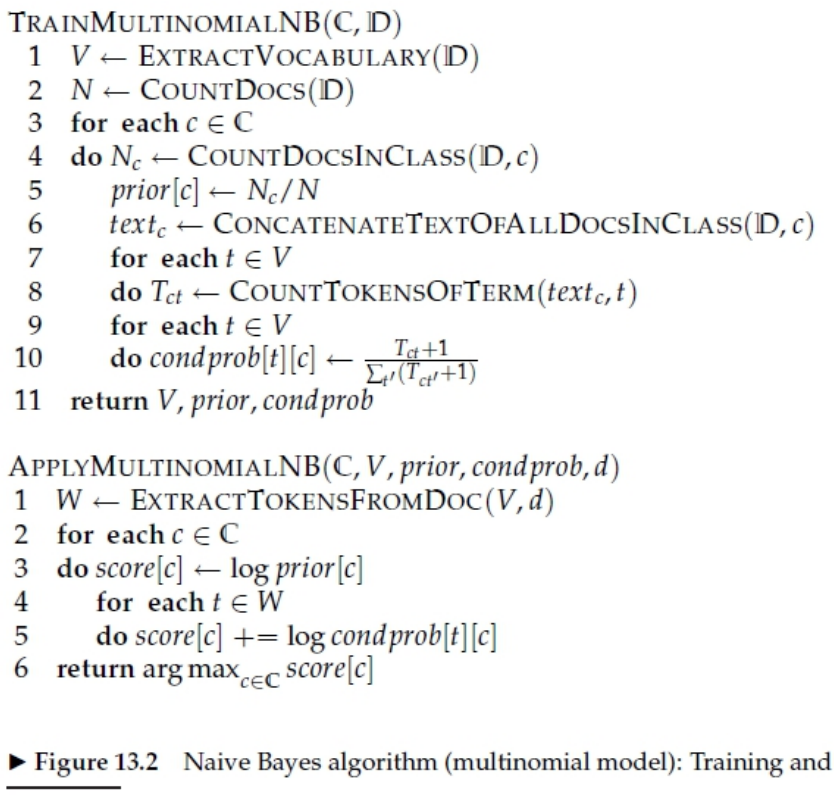
Nous vous fournissons pour ce TP :
- les critiques de film dans le répertoire `data/imdb1`,
- le squelette de programme en python `sentiment_analysis.py`
- la description de l'algorithme du classifieur bayesien naïf `NaiveBayes.pdf`
- le pseudo-code de l'algorithme, p 260 de [2] : http://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf. Cet algorithme est une simplification de l'article [1] et présenté en Fig. 1.

Implémentation du classifieur

Questions :

1. Compléter la fonction `count_words` qui va compter le nombre d'occurrences de chaque mot dans une liste de `string` et renvoyer le vocabulaire.
2. Expliquer comment les classes positives et négatives ont été assignées sur les critiques de films (voir fichier `poldata.README.2.0`)
3. Compléter la classe `NB` pour qu'elle implémente le classifieur *Naive Bayes* en vous appuyant sur le pseudo-code de la figure 1 et sa documentation ci-dessous :
   - le vocabulaire V correspond à l'ensemble des mots différents composant un ensemble de documents (`vocabulary` dans `count_words`)
   - $\mathbb{C}$ correspond à l'ensemble des classes et $\mathbb{D}$ l'ensemble des documents,
   - la fonction *countTokensOfTerm(text,t)* représente le nombre d'occurrences d'un mot t dans un ensemble de textes *text* (calcul fait dans `count_words`),
   - l'étape de lissage appelée lissage de Laplace (+1 ligne 10) permet l'attribution de probabilité non nulle à des mots qui n'interviendraient pas dans l'ensemble d'apprentissage,
   - la fonction *ExtractTokensFromDoc(V,d)* récupère le vocabulaire associé au document *d*.

   Autrement dit,
4. Evaluer les performances de votre classifieur en cross-validation 5-folds.
5. Modifiez la fonction `count_words` pour qu'elle ignore les "stop words" dans le fichier `data/english.stop`. Les performances sont-elles améliorées ?



TRAINMULTINOMIALNB(C, 𝔻)
1 $V \leftarrow$ EXTRACTVOCABULARY(𝔻)
2 $N \leftarrow$ COUNTDOCS(𝔻)
3 **for each** $c \in \mathbb{C}$
4 **do** $N_c \leftarrow$ COUNTDOCSINCLASS(𝔻, c)
5 $prior[c] \leftarrow N_c/N$
6 $text_c \leftarrow$ CONCATENATETEXTOFALLDOCSINCLASS(𝔻, c)
7 **for each** $t \in V$
8 **do** $T_{ct} \leftarrow$ COUNTTOKENSOFTERM($text_c, t$)
9 **for each** $t \in V$
10 **do** $condprob[t][c] \leftarrow \frac{T_{ct}+1}{\sum_{t'}(T_{ct'}+1)}$
11 **return** $V, prior, condprob$

APPLYMULTINOMIALNB(C, V, $prior$, $condprob$, d)
1 $W \leftarrow$ EXTRACTTOKENSFROMDOC(V, d)
2 **for each** $c \in \mathbb{C}$
3 **do** $score[c] \leftarrow \log prior[c]$
4 **for each** $t \in W$
5 **do** $score[c] \mathrel{+}= \log condprob[t][c]$
6 **return** $\arg\max_{c \in \mathbb{C}} score[c]$

▶ **Figure 13.2** Naive Bayes algorithm (multinomial model): Training and testing.

FIGURE 1 – Pseudo-code de l'algorithme *Naive Bayes* : apprentissage et classification

Utilisation de scikitlearn

Vous avez implémenté votre propre classifieur. Vous allez maintenant utiliser `scikitlearn` et NLTK [1].

Question 1 : Comparer votre implémentation avec `scikitlearn`.
On utilisera la classe `CountVectorizer` et un `Pipeline` :

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.pipeline import Pipeline
```

Vous expérimenterez en autorisant les mots et bigrammes ou en travaillant sur les sous-chaines de caractères (option `analyzer='char'`).

Question 2 : Tester un autre algorithme de la librairie `scikitlearn` (ex : LinearSVC, LogisticRegression).

Question 3 : Utiliser la librairie NLTK afin de procéder à une racinisation (*stemming*). Vous utiliserez la classe `SnowballStemmer`.

```
from nltk import SnowballStemmer
```

1. NLTK (Natural Language ToolKit) est une librairie pour Python offrant de nombreuses fonctionnalités pour le traitement automatique du langage naturel. Elle permet en particulier d'étiqueter et de lemmatiser des corpus en langue anglaise. La documentation de cette librairie se trouve sur http://nltk.org/book/

Question 4 : Filtrer les mots par catégorie grammaticale (POS : Part Of Speech) et ne garder que les noms, les verbes, les adverbes et les adjectifs pour la classification.

```
from nltk import pos_tag
```

Références

[1] Pang, Bo and Lee, Lillian and Vaithyanathan, S, *Thumbs up ? : sentiment classification using machine learning techniques*. ACL-02 conference on Empirical methods in natural language processing-Volume 10, p79-86, 2002. 1

[2] Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze, *Introduction to information retrieval*. Vol. 1. Cambridge : Cambridge University Press, 2008.

1