seance_2

September 29, 2020

1 Séance 2 : Apprentissage non supervisé

Le corpus est le même que celui de la première séance. Cette fois, nous allons nous intéresser au contenu des articles par le biais de leur résumé.

L'objectif est d'étudier des algorithmes de clustering, de comparer quantitativement leur efficacité et les résultats obtenus.

1.1 Pré-traitement des données

Les algorithmes de clustering sont souvent gourmands en temps de calcul. Pour cette séance, il est conseillé de se contenter d'une toute petite partie du corpus, par exemple que les articles publiés en 2010 et qui sont dans le domaine de l'informatique (cf fonction sample_corpus).

Lorsqu'on travaille avec des données textuelles, la première étape est le pré-traitement des données : lemmatizer, stop words, transformation tf-idf, ... (cf cours RITAL de M1). Les traitements minimales vous sont donnés dans les blocs suivants.

[ ]:
```
import json
from datetime import datetime
import re, string
import sklearn
import numpy as np
from scipy.sparse import csr_matrix,lil_matrix
import gzip

from sklearn.feature_extraction.text import TfidfVectorizer,CountVectorizer
from nltk.stem import WordNetLemmatizer
from nltk import word_tokenize
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
```

[ ]:
```
FILENAME = "arxiv-2010-2017.json.gz"

def read_arxiv(f=FILENAME):
    """ lit le fichier arxiv.json """
```

```
    res = dict()
    with  gzip.open(f) as fp:
        for l in fp:
            js = json.loads(l)
            res[js["id"]] = js
    return res

def build_labels(dic,labels2id=None,id2labels=None,pref=None):
    """ construit la matrice des labels associée à dic
        renvoie la matrice, le dictionnaire de correspondance entre label et index, et le dictionnaire inverse
        ne conserve que les labels commencant par pref.
    """
    if pref is None: pref =""
    if labels2id is None:
        labels = set()
        for k in dic.values():
            labels.update([l for l in k["categories"].split(" ") if l.startswith(pref)])
        labels2id = dict(zip(sorted(labels),range(len(labels))))
        id2labels = dict(zip(range(len(labels)),sorted(labels)))
    labs = lil_matrix((len(dic),len(labels2id)))
    for i,k in enumerate(dic.values()):
        for l in k["categories"].split(" "):
            if l.startswith(pref):
                labs[i,labels2id[l]] = 1.
    return labs,labels2id, id2labels

def get_dates(dic):
    return [datetime.strptime(x["created"],"%a, %d %b %Y %H:%M:%S GMT") for k in dic.values() for x in k["versions"] if x['version']=="v1"]

def sample_corpus(dic,year=2011):
    """ filtre le corpus en enlevant les articles sans DOI et ne gardant que les articles de computer science et de date de publication < year"""
    dates = get_dates(dic)
    return dict([d for (d,y) in zip(dic.items(),dates) if d[1]['doi'] is not None and  sum([dd.startswith("cs.") for dd in d[1]['categories'].split(" ")])>=1 and y.year<year])
```

[ ]:
```
def get_titles(dic):
    return [k["title"] for k in dic.values()]
def get_abstracts(dic):
    return [k["abstract"] for k in dic.values()]


class LemmaTokenizer:
```

```
    def __init__(self):
        self.wnl = WordNetLemmatizer()
    def __call__(self,doc):
        return [self.wnl.lemmatize(t) for t in word_tokenize(re.sub('[\W_]+','␣
↪',doc.lower()))]
```

[ ]:
```
vectorizer =␣
↪CountVectorizer(tokenizer=LemmaTokenizer(),lowercase=True,max_features=100000,min_df=0.
↪01)
tfidf =␣
↪TfidfVectorizer(tokenizer=LemmaTokenizer(),lowercase=True,max_features=100000,min_df=0.
↪01)

dataraw = read_arxiv(FILENAME)
data = sample_corpus(dataraw)
labels,labels2id,id2labels = build_labels(data,pref="cs.")
x = tfidf.fit_transform(get_abstracts(data))
```

1.2 Travail à réaliser

Vous étudierez au moins les algorithmes de clustering suivant : * k-means * clustering agglomeratif * clustering spectral

La représentation brute tf-idf est très parcimonieuse et de grande dimension. Il est usuel d'utiliser une distance cosine dans ce cas plutôt que la distance euclidienne. Vous comparerez les résulats obtenus selon ces deux distances.

Par ailleurs, afin de réduire la dimensionalité et de rendre plus expressives les dimensions, vous pouvez utiliser des approches de réduction de dimension telles que l'ACP et la NMF.

Evaluation quantitative : vous disposez des catégories de chaque article, il vous est ainsi possible d'évaluer la composition de chaque cluster en fonction des catégories qui s'y retrouvent : homogénéité, complétude, ... Il est intéressant également de regarder la taille des clusters.

Evaluation qualitative : pour chaque technique qui s'y prête, analyser les résultats obtenus en fonction de l'interprétabilité du modèle (signification des axes pour l'ACP, des atomes pour la NMF, des barycentres pour k-means, ...).

Visualisation : vous pouvez utiliser des algorithmes de visualisation tels que MDS ou la t-SNE afin de projeter les représentations brutes des données ou après réduction de dimension en 2D (et en coloriant chaque article en fonction de sa classe). Vous pouvez les comparer aux représentations obtenus par l'ACP.