Analyse de texte et traitement du langage naturel

1. Introduction au text mining

- utilisation du ML pour l’analyse textuelle

- C’est l’analyse statistique de textes

- transformer un texte non structuré en données structurées pour ensuite procéder à l’analyse

- NLP (Natural Language Processing, “traitement naturel du langage”) : permet aux machines de comprendre et de traiter le langage humain automatiquement

- ex : récupérer et analyser les commentaires sur un produit donné

Différence avec le Data mining

Le text mining se focalise sur des données textuelles (non structurées) en utilisant des concepts et des outils statistiques mais également linguistiques (lexique, syntaxe…).

Nécessite une transformation profonde du texte pour pouvoir être exploité.

Le data mining obtient des données homogènes tandis que celles du text mining sont hétérogènes.

Fonctionnement

- Extraire des connaissances et des structures à partir de documents
- structures = patterns, schémas
- similarités entre des textes
- causalité = on cherche des relations de cause à effet
- catégoriser, classer

Applications

- **Classification de documents** : attribue une étiquette/label à chaque texte, ex : analyse de sentiments (commentaires), détection de spams

- ***Clustering*** (“segmentation”) : faire des regroupements les plus intelligents possibles, tri sur d’ énormes d’énormes bases documentaires

- ***Information retrieval*** (“fouille de documents”) : tri et recherche dans documents

- Text mining sur image
- Détection de la langue et traduction automatique
- Génération automatique de texte, détection de plagiat

Etapes

- *Segmentation* : fragmentation du texte (phrases)
- *Tokenization* : phrases fragmentées en mots
- *Stop words* : mots vides, inutiles, superflus
- *Stemming* : racine d'un mot
- *Lemmatization* : am/are/is => be
- *Speech tagging* : nom/verbe/article/préposition…
- *Named entity tagging*

Sur le stemming et lemmatization :
https://www.analyticssteps.com/blogs/what-stemming-and-lemmatization-nlp

Les algorithmes statistiques du *text mining* prennent en entrée des chiffres et non pas des textes

mots ⇒ chiffres ⇒ Bag of words : méthode d'analyse de texte

- tous les mots sont mélangés, dénombrement (nombre d'occurrence), binarisation : mot apparaît ou pas

⇒ TF-IDF ("*Term Frequency-Inverse Document Frequency*") : pondération en prenant compte de 2 facteurs : la fréquence du mot dans un document, fréquence de ce mot dans tous les documents

⇒ donne un poids élevé aux mots apparaissant souvent, diminue le poids de ceux qui apparaissent dans d'autres documents du corpus ("*inverse*").

- *Workcloud* : permet de visualiser les mots apparaissant le plus souvent et qui perdent leur caractère discriminatoire.

- Web scraping avec Python : Scrapy, Pyspider, Requests, BeautifulSoup, Selenuim

2. La matrice documents termes

Objectif : convertir corpus en tableau de données “attributs-valeurs” pouvant par la suite être traité par des algorithmes (tout en évitant le plus possible la perte d’information).

Bag of words

```
from sklearn.feature_extraction.text import CountVectorizer

# Création d'un sac de mots vide
vectorizer = CountVectorizer()

# Création d'un corpus (3 documents)
corpus = ["decouverte du text mining", "ceci n'est pas un test", "sac de mots"]

# Calcul du sac
vectorizer.fit(corpus)
bow = vectorizer.transform(corpus)
```

```
bow
```

```
<3x12 sparse matrix of type '<class 'numpy.int64'>'
        with 12 stored elements in Compressed Sparse Row format>
```

Extraction des termes :

3. La catégorisation de textes (document classification)

Consiste à assigner des étiquettes/labels aux données de texte non structurées (cadre de l'apprentissage supervisé)

- *Topic Analysis* (Analyse du sujet) : principaux thèmes ou sujets d'un texte
- Analyse de sentiment (à partir du texte)
- *Topic detection* : identification des sujets émergents

Word embedding (Word2Vec)

- Prolongement lexical. Représentation des termes à l'aide d'un vecteur numérique, contextualisée par le voisinage. Représentation des documents. Word2Vec with Gensim - Python

On appelle la technique de représentation d’un mot, ou un ensemble de mots en vecteurs de dimension inférieure, word embeddings, soit littéralement « plongement de mot ».

- transformation sophistiquée des textes en chiffres
- prise en compte du contexte = mot avant et mot après

mots ⇒ colonnes de chiffres

contexte proche ⇒ colonnes de chiffres proches

= trouver les mots qui sont proches

Ex : Rome = (Paris-France) + Italie

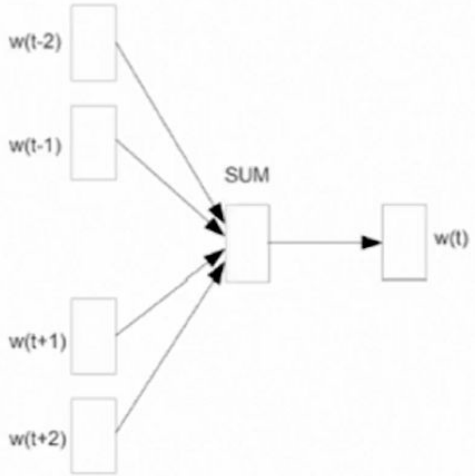
Paris ⇒ France

Rome ⇒ Italie (calcule la distance)

- travaille au niveau du sens commun des mots, ce sens est caché dans le contexte d'utilisation des mots, Word embedding permet d'extraire ces notions
- Gensim = API avancée de text mining
- Word2vec = algorithme permettant de construire une représentation contextuelle des mots qui appartiennent à un document
- 2 méthodes de calcul du Word embedding :
    - CBOW (Continuous Bag of Words): déduit le mot à partir des mots qui l'entourent
    - Skip-gram : comment prédire les mots positionnés juste avant et juste après chaque mot sur lequel porte l'analyse

CBOW

Entrée Projection Sortie



Skip-gram

Entrée Projection Sortie