Janvier 2022

Topic modeling

Projet Simplon #15

Présenté par

Adil et Véronique

Sommaire

Introduction

Le Topic modeling peut être utilisé pour modéliser des documents de texte afin de réinterpréter les textes sous forme mathématique, ce qui nous permet non seulement de calculer la distance des documents, mais aussi de les classifier ou de les regrouper selon nos besoins.

Pour répondre à ce besoin, notre équipe, composée d'un data analyst, un data scientist et d'un data engineer, a developpé un outil d'intelligence artificielle capable de faire ressortir la thématique générale d'un ou des plusieurs articles.

Cas d'usage

La recommandation peut se baser sur le principe de similarité: si vous consultez la page d'un portable sur le site, on va probablement vous proposer des portables similaires à ceux que vous venez de regarder, la similarité se calcule souvent par rapport à des critères comme le prix, le modèle, la couleur ou les fonctionnalités.

Puis une fois que vous avez choisi le portable à acheter, on vous propose ensuite les produits "complémentaires", par exemple pour le portable que vous avez choisi, on va vous proposer un câble de connexion, un chargeur et une housse.

Pour la recommandation des articles, on s'appuie généralement sur l'approche de similarité, c'est-à-dire trouver les articles du même sujet que celui consulté.

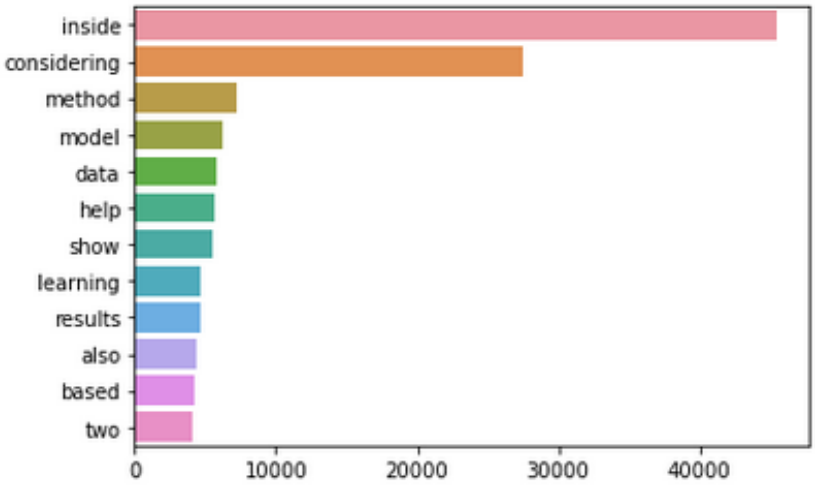
Notre projet

Notre dataset contient des articles de recherche scientifique.

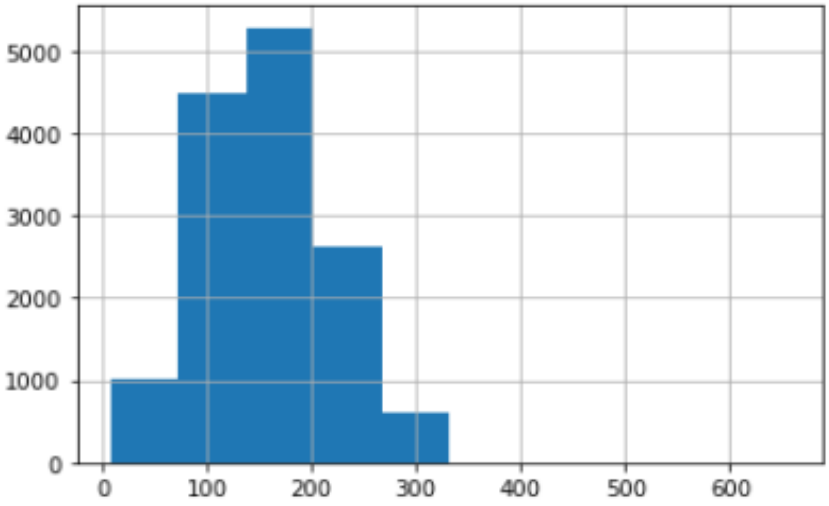
Nous devons utilisez l'ensemble de données pour prédire le sujet d'un article à l'aide de son résumé (colonne ABSTRACT).

	id	ABSTRACT	Computer Science	Mathematics	Physics	Statistics	Analysis of PDEs	Applications	Artificial Intelligence
0	1824	a ever-growing datasets inside observational a...	0	0	1	0	0	0	0
1	3094	we propose the framework considering optimal $...	1	0	0	0	0	0	0
2	8463	nanostructures with open shell transition meta...	0	0	1	0	0	0	0

EDA - Analyse



Nombre de mots les plus fréquents hors stopwords



Nombre de mots par résumé (colonne ABSTRACT)

Preprocessing

Tokenisation, lemmatisation et suppression des stopwords

```
# tokenize / remove stopwords / lemmatize

def preprocess_news(df):
    corpus=[]
    stem=PorterStemmer()
    lem=WordNetLemmatizer()
    for news in df_train['ABSTRACT']:
        words=[w for w in word_tokenize(news) if (w not in stop)]

        words=[lem.lemmatize(w) for w in words if len(w)>2]

        corpus.append(words)
    return corpus

corpus=preprocess_news(df_train)
```

Conversion du corpus en dictionnaire puis bag of words

```
# bag of words" avec gensim

dic=gensim.corpora.Dictionary(corpus)
bow_corpus = [dic.doc2bow(doc) for doc in corpus]
```

Preprocessing avec les librairies NLTK et Gensim
Identification des stopwords, création du corpus, tokenisation, lemmatisation et suppression des stopwords, conversion du corpus en dictionnaire (mappage des mots avec leur ID) puis bag of words (nombre de fois où le mot apparaît).

Modèle

Le modèle LDAMulticore utilisent tous les cœurs de processeur pour accélérer la formation du modèle.

```
lda_model = gensim.models.LdaMulticore(bow_corpus,
                                       num_topics = 25,
                                       id2word = dic,
                                       passes = 10,
                                       workers = 2)
```

bow_corpus : notre corpus transformé en bag of words

num_topics = 25 : nombre de sujets que l'on souhaite trouver

Id2word = dic : déterminer la taille du vocabulaire, ainsi que pour le débogage

passes = 10 : nombre de passages dans le corpus pendant l'entraînement.

workers = 2 : propre au LDAMulticore pour optimiser le temps/performance

Performance

Afin de mesurer et analyser notre modèle, nous avons utilisé un métrique :

Coherence score / Topic coherence : Le score de cohérence thématique évaluent le degré de similitude sémantique entre les mots les mieux ‘notés’ dans le sujet.

Nous avon sobtenu un coherence score de 0.47 avec notre modèle LDA, ce qui est plutôt moyen.

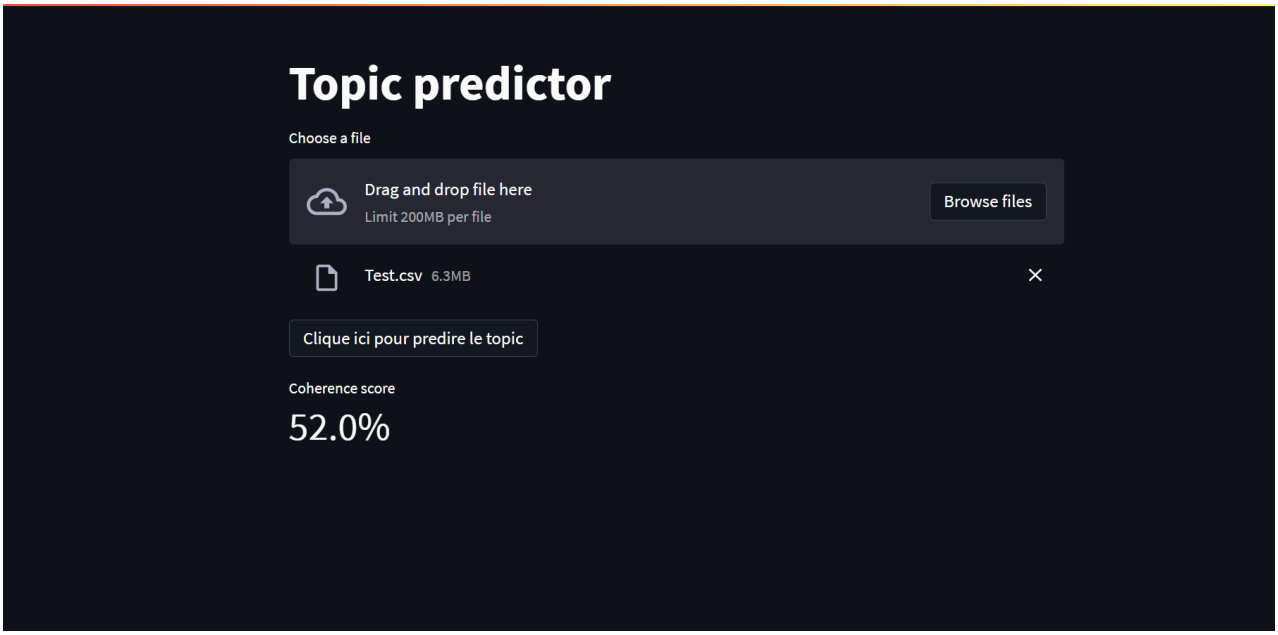
Amélioration

Une amélioration de notre modèle est souhaitable et vous vous proposons de tester d’autres techniques et/ou algorithmes comme LDA2vec en deep learning.

LDA2vec
Algorithme “hybride” de deep learning créé, en 2016, par Christopher Moody. Il associe ainsi les algorithmes de LDA et word2vec, dans le but d’optimiser le topic modeling.

Interface utilisateur

Disponible uniquement en local, notre application a été codé en python avec Streamlit, et vous propose de calculer le cohérence score de votre modèle.



Livrables

Les livrables suivants sont disponibles sur Github :

https://github.com/bonjourcerise/simplon_topic_modeling_nlp

1. **notebook_EDA** : Jupyter Notebook avec l'exploration et analyse du dataset
2. **notebook_model** : Jupyter Notebook avec le modèle que nous avons sélectionné
3. **model.pkl** : notre modèle format pickle
4. **planning.txt** : notre planning pour réaliser ce projet

Planning

- 12/01 PM: Veille sur le topic modeling
- 13/01 AM/PM: Veille, recherches, EDA, premiers modèles
- 14/01 AM/PM: Test de plusieurs modèles et techniques
- 19/01 PM: Test de plusieurs modèles et techniques
- 20/01 AM/PM: Choix du modèle, nettoyage des notebooks, app streamlit en local, rédaction de la présentation
- 21/01 AM: Test word embedding, rédaction de la présentation, rédaction du rapport