Fouille de données textuelles

Rapport approfondi sur Word2Vec De la théorie à la pratique

Harrouche Sarah

Master Informatique et Big Data

2024/2025

1 Introduction

Ce projet s'inscrit dans le cadre de l'analyse de données textuelles appliquée à une base de données de films. L'objectif est d'explorer l'utilisation du deep learning, particulièrement Word2Vec, pour exploiter l'information contenue dans les descriptions textuelles, et de mettre en pratique ces concepts à travers une implémentation concrète en utilisant des outils comme Gensim. Ce rapport couvre à la fois les aspects théoriques et pratiques de Word2Vec.

2 Contexte et objectifs

Ce rapport documente un projet visant à :

- Explorer et prétraiter un corpus de descriptions de films afin d'extraire des informations pertinentes et exploitables.
- Entraîner un modèle Word2Vec sur ce corpus pour apprendre des représentations vectorielles des mots et analyser la qualité des embeddings générés.
- Visualiser et interpréter les relations sémantiques entre les mots afin de mieux comprendre les tendances et associations linguistiques au sein du corpus.

3 Théorie de Word2Vec

3.1 Concepts fondamentaux

Word2Vec est une technique d'apprentissage non supervisé qui vise à représenter les mots d'un corpus sous forme de vecteurs continus dans un espace de dimension réduite. Ces représentations vectorielles capturent les relations sémantiques entre les mots, permettant ainsi de mesurer leur similarité ou d'effectuer des opérations analogiques.

3.2 Modèles Word2Vec

Word2Vec propose deux architectures principales pour apprendre ces représentations vectorielles :

- **CBOW (Continuous Bag of Words)** : Ce modèle prédit un mot cible à partir des mots contextuels environnants. Par exemple, pour la phrase "Le chat est sur le tapis", si le mot cible est "est", les mots contextuels seraient "Le", "chat", "sur", "le", "tapis".
- **Skip-gram** : Ce modèle fait l'inverse de CBOW. Il prédit les mots contextuels à partir d'un mot cible. Par exemple, pour le mot cible "est", le modèle prédirait les mots "Le", "chat", "sur", "le", "tapis".

3.3 Formulation mathématique

CBOW L'objectif de CBOW est de maximiser la probabilité conditionnelle suivante :

$$P(w_t|w_{t-m}, ..., w_{t+m}) \tag{1}$$

La fonction de perte associée est :

$$J = -\sum_{t=1}^{T} \log P(w_t|w_{t-m}, ..., w_{t+m}) \tag{2}$$

Skip-gram Le modèle Skip-gram maximise la probabilité suivante :

$$P(w_{t-m}, ..., w_{t+m}|w_t) \tag{3}$$

Sa fonction de perte est :

$$J = -\sum_{t=1}^{T} \sum_{-m \leq j \leq m, j \neq 0} \log P(w_{t+j}|w_t) \tag{4}$$

3.4 Optimisation

Pour accélérer l'entraînement, Word2Vec utilise :

- **Negative Sampling** : Réduit la complexité de calcul en mettant à jour un petit nombre de mots négatifs.
- **Hierarchical Softmax** : Utilise une arborescence binaire pour réduire le coût de calcul de la fonction softmax.

4 Mise en pratique avec Gensim

L'implémentation est réalisée en Python avec la bibliothèque Gensim.

4.1 Prétraitement des données

Le texte des descriptions de films a été nettoyé et transformé en corpus de phrases exploitables par Word2Vec.

```
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

stop_words = set(stopwords.words('english'))

def preprocess_text(text):
    text = text.lower()
    text = re.sub(r'[^\w\s]', '', text)
    tokens = word_tokenize(text)
    tokens = [word for word in tokens if word not in stop_words]
    return tokens
```

4.2 Entraînement du modèle Word2Vec

Après le prétraitement, le modèle Word2Vec est entraîné avec les paramètres suivants :

```
from gensim.models import Word2Vec

model = Word2Vec(sentences=processed_texts, vector_size=150,
window=7, min_count=10, workers=4, sg=1 , negative=15 )
```

5 Analyse des résultats

5.1 Exploration des similarités entre mots

Après entraînement, nous pouvons examiner les relations entre les mots :

```
print("\nAnalyse des similarit s entre mots...")
test_words = ['american', 'love', 'woman', 'september']
for word in test_words:
    try:
        similar_words = model.wv.most_similar(word)
        print(f"\nMots similaires   '{word}':")
        for w, score in similar_words[:5]:
            print(f"{w}: {score:.4f}")
    except KeyError:
        print(f"Le mot '{word}' n'est pas dans le vocabulaire")
```

5.2 Visualisation avec t-SNE

La réduction de dimension permet d'afficher les relations entre les vecteurs de mots :

```
tsne = TSNE(n_components=2, perplexity=perplexity, random_state=42)
reduced = tsne.fit_transform(word_vectors)

# Visualisation
plt.figure(figsize=(10, 6))
sns.scatterplot(x=reduced[:, 0], y=reduced[:, 1])
```

6 Conclusion

Word2Vec est une technique puissante pour la représentation des mots en NLP. En capturant les relations sémantiques et syntaxiques entre les mots, il a révolutionné de nombreuses tâches de traitement du langage naturel. Bien qu'il ait des limites, il reste un outil essentiel dans la boîte à outils du data scientist. Avec des bibliothèques comme Gensim, l'implémentation de Word2Vec est accessible et facile à intégrer dans des pipelines de NLP.

7 Références

- Mikolov, T., et al. (2013). Efficient Estimation of Word Representations in Vector Space.
- Gensim Documentation : `https://radimrehurek.com/gensim/`
- Stanford NLP Course : `https://web.stanford.edu/class/cs224n/`