

TP1 : Embeddings Word2Vec

Master 1 IA & Big Data - Université Paris 8

1. Objectif

Créer et analyser des embeddings Word2Vec sur deux datasets textuels pour comprendre comment les mots similaires sont capturés par leurs représentations vectorielles.

Dataset	Taille	Variable cible	Colonne texte
Films	45 466 films	vote_average (note)	overview (description)
Reviews Amazon	194 439 reviews	overall (1-5 étoiles)	reviewText (avis)

2. Preprocessing du texte

Étapes appliquées :

1. Lowercase : mise en minuscules
2. Suppression : chiffres et ponctuation (regex [^a-zA-Z])
3. Tokenisation : découpage en mots (NLTK word_tokenize)
4. Filtrage : stop words anglais + mots ≤ 2 caractères

Avant	Led by Woody, Andy's toys live happily in his room...
Après	[led, woody, andys, toys, live, happily, room, ...]

3. Réduction du vocabulaire

Paramètre **min_count=5** : Supprime les mots apparaissant moins de 5 fois.

Résultat : Vocabulaire réduit de ~75% (88 399 → 22 876 mots pour les films).

Avantage : Embeddings de meilleure qualité (plus de contexte pour chaque mot).

4. Word2Vec : Principe

Définition : Transformer chaque mot en vecteur de nombres (100 dimensions).

Méthode : Skip-gram (prédir le contexte à partir d'un mot central).

Principe : "Un mot est défini par les mots qui l'entourent".

Résultat : Les mots ayant des contextes similaires auront des vecteurs proches.

Paramètre	Valeur	Description
vector_size	100	Dimension des vecteurs
window	5	Contexte : 5 mots avant et après
min_count	5	Ignorer mots rares
sg	1	Skip-gram (vs CBOW)
epochs	10	Nombre d'itérations

5. Analyse des embeddings : Mots similaires

Les vecteurs capturent le sens sémantique. Les mots similaires ont des vecteurs proches.

Mot	Top 3 mots similaires
love	affection (0.74), romance (0.74), madly (0.73)
action	installment (0.81), paced (0.79), motion (0.78)
hero	protagonist, character, warrior

6. Relations vectorielles

Les embeddings capturent des relations sémantiques complexes :
king - man + woman ≈ princess, ruler, empress

Cela montre que les vecteurs encodent des concepts comme le genre, le statut social, etc.

7. Exemples de vecteurs (10 premières dimensions)

Chaque mot est représenté par un vecteur de 100 nombres réels :

'**love**' → [0.084, 0.115, -0.090, -0.551, -0.089, -0.299, 0.384, ...]
'**hero**' → [0.462, -0.068, 0.424, -0.315, 0.457, -0.075, 0.341, ...]

Ces vecteurs sont utilisables pour des tâches de NLP (classification, clustering, etc.).

8. Conclusion

Word2Vec transforme avec succès le texte en représentations vectorielles qui capturent :

- La **similarité sémantique** (mots proches = vecteurs proches)
- Les **relations complexes** (analogies, genre, etc.)
- Le **contexte d'utilisation** des mots

Ces embeddings sont la base pour des modèles NLP plus avancés.

Stack technique

- Python 3.13
- NLTK : tokenisation, stop words
- Gensim : Word2Vec
- Pandas / NumPy : traitement de données