

# Projet Knowledge Extraction from Unstructured Text

Partie A : Preprocessing et Représentation Textuelle

Rapport Technique

Jacques Gastebois

Master 2 VMI - Université Paris Cité

IFLCE085 - Recherche et extraction sémantique à partir de texte

Prof. Salima Benbernou

30 Novembre 2025

## Résumé

Ce rapport présente la méthodologie complète de preprocessing appliquée au dataset SciREX dans le cadre du projet d'extraction de connaissances à partir de textes non structurés. Le pipeline implémenté comprend le nettoyage, la tokenization, le POS tagging, la lemmatization et la vectorisation TF-IDF. Les résultats sont exportés dans des formats exploitables pour la Partie B du projet.

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Contexte du Projet	3
1.2	Objectifs	3
1.3	Dataset	3
<b>2</b>	<b>Méthodologie</b>	<b>3</b>
2.1	Pipeline de Preprocessing	3
2.2	Justification des Choix Techniques	4
2.2.1	Nettoyage	4
2.2.2	Lemmatization vs Stemming	4
2.2.3	TF-IDF sur Textes Lemmatisés	4
<b>3</b>	<b>Implémentation</b>	<b>4</b>
3.1	Technologies Utilisées	4
3.2	Exemple de Code	4
3.2.1	Fonction de Nettoyage	4
3.2.2	Lemmatization avec spaCy	5
3.2.3	Vectorisation TF-IDF	5

<b>4</b>	<b>Résultats</b>	<b>5</b>
4.1	Statistiques du Preprocessing . . . . .	5
4.2	Exemple de Transformation . . . . .	5
4.3	Top 10 Features TF-IDF (Document 0) . . . . .	6
<b>5</b>	<b>Export des Données</b>	<b>6</b>
5.1	Fichiers Générés . . . . .	6
5.2	Structure du CSV . . . . .	6
<b>6</b>	<b>Utilisation pour la Partie B</b>	<b>7</b>
6.1	Chargement des Données . . . . .	7
6.1.1	Textes Prétraités . . . . .	7
6.1.2	Matrice TF-IDF . . . . .	7
6.1.3	Métadonnées . . . . .	7
6.2	Recommandations pour la Partie B . . . . .	8
<b>7</b>	<b>Accès aux Ressources</b>	<b>8</b>
7.1	Liens Importants . . . . .	8
7.2	Instructions d'Exécution . . . . .	8
7.2.1	Sur Google Colab . . . . .	8
7.2.2	En Local . . . . .	8
<b>8</b>	<b>Conclusion</b>	<b>9</b>
8.1	Résumé des Contributions . . . . .	9
8.2	Points Clés . . . . .	9
8.3	Perspectives pour la Partie B . . . . .	9

# 1 Introduction

## 1.1 Contexte du Projet

Ce projet s'inscrit dans le cadre du cours IFLCE085 et vise à développer un système complet d'extraction de connaissances à partir de textes scientifiques non structurés. La Partie A se concentre sur le preprocessing et la préparation des données.

## 1.2 Objectifs

- Nettoyer et normaliser le dataset SciREX (306 documents train)
- Appliquer un pipeline NLP complet (tokenization, POS tagging, lemmatization)
- Générer des représentations vectorielles TF-IDF
- Exporter les résultats dans des formats réutilisables pour la Partie B

## 1.3 Dataset

**SciREX** (Scientific Information Extraction) : Dataset de 438 articles scientifiques annotés provenant de AI2.

- **Train** : 306 documents
- **Dev** : 66 documents
- **Test** : 66 documents

# 2 Méthodologie

## 2.1 Pipeline de Preprocessing

Le pipeline complet suit une approche séquentielle en 5 étapes :

### Étape 1 : Nettoyage et Normalisation

- Conversion en lowercase
- Suppression des caractères spéciaux (garde lettres, chiffres, espaces)
- Normalisation des espaces multiples

### Étape 2 : Tokenization

- Outil : NLTK `word_tokenize`
- Découpage du texte en tokens individuels

### Étape 3 : POS Tagging

- Outil : NLTK `pos_tag`
- Annotation grammaticale (nom, verbe, adjectif, etc.)

### Étape 4 : Lemmatization

- Outil : spaCy `en_core_web_sm`
- Réduction des mots à leur forme canonique
- **Appliqué sur TOUS les 306 documents train**

### Étape 5 : Vectorisation TF-IDF

- Outil : scikit-learn `TfidfVectorizer`
- **Calculé sur les textes lemmatisés** (pipeline complet)
- Paramètres : `max_features=5000`, `ngram_range=(1,2)`

## 2.2 Justification des Choix Techniques

### 2.2.1 Nettoyage

Le nettoyage est essentiel pour réduire le bruit et standardiser le texte. La suppression des caractères spéciaux permet de se concentrer sur le contenu sémantique.

### 2.2.2 Lemmatization vs Stemming

La lemmatization a été préférée au stemming car elle produit des formes linguistiquement correctes (ex : "better" "good" au lieu de "bett").

### 2.2.3 TF-IDF sur Textes Lemmatisés

**Décision critique** : Le TF-IDF est calculé sur les textes lemmatisés (et non sur les textes nettoyés) pour garantir :

- Une réduction du vocabulaire (formes canoniques)
- Une meilleure capture des concepts sémantiques
- Une cohérence avec le pipeline NLP complet

## 3 Implémentation

### 3.1 Technologies Utilisées

Librairie	Version/Utilisation
Python	3.12
pandas	2.2.2 (manipulation de données)
numpy	2.0.2 (calculs numériques)
NLTK	3.9.1 (tokenization, POS tagging)
spaCy	3.8.11 (lemmatization)
scikit-learn	1.6.1 (TF-IDF)
scipy	1.16.3 (matrices sparse)

TABLE 1 – Stack technologique

### 3.2 Exemple de Code

#### 3.2.1 Fonction de Nettoyage

```
def clean_text(text):  
    """Nettoie le texte."""  
    if not isinstance(text, str):  
        return ""  
  
    # 1. Lowercase  
    text = text.lower()
```

```
# 2. Suppression caracteres speciaux
text = re.sub(r'^a-z0-9\s', ' ', text)

# 3. Normalisation espaces
text = re.sub(r'\s+', ' ', text).strip()

return text
```

### 3.2.2 Lemmatization avec spaCy

```
def lemmatize_text(text):
    """Lemmatiser le texte avec spaCy."""
    doc = nlp(text)
    lemmas = [token.lemma_ for token in doc]
    return ' '.join(lemmas)
```

### 3.2.3 Vectorisation TF-IDF

```
# Preparation des textes LEMMATISES
train_texts = [doc['lemmatized_text']
                for doc in train_data
                if 'lemmatized_text' in doc]

# Vectorisation
tfidf_vectorizer = TfidfVectorizer(
    max_features=5000,
    min_df=2,
    max_df=0.8,
    ngram_range=(1, 2)
)

tfidf_matrix = tfidf_vectorizer.fit_transform(train_texts)
```

## 4 Résultats

### 4.1 Statistiques du Preprocessing

### 4.2 Exemple de Transformation

Texte original :

*"Full-Resolution Residual Networks for Semantic Segmentation in Street Scenes"*

Texte nettoyé :

*"full resolution residual networks for semantic segmentation in street scenes"*

Texte lemmatisé :

*"full resolution residual network for semantic segmentation in street scene"*

Métrique	Valeur
Documents train traités	306
Documents dev traités	66
Documents test traités	66
Documents lemmatisés	306 (100%)
Taille matrice TF-IDF	306 × 5000
Densité matrice	23.7%
Vocabulaire TF-IDF	5000 features

TABLE 2 – Statistiques de preprocessing

### 4.3 Top 10 Features TF-IDF (Document 0)

Feature	Score TF-IDF
stream	0.2819
cityscapes	0.2445
residual	0.2226
resolution	0.1905
segmentation	0.1893
reference reference	0.1770
pooling	0.1688
pooling operations	0.1449
image	0.1281
semantic segmentation	0.1195

TABLE 3 – Top 10 features TF-IDF pour le premier document

## 5 Export des Données

### 5.1 Fichiers Générés

Tous les fichiers sont exportés dans le dossier `preprocessed_data/` :

Fichier	Format	Taille
<code>train_preprocessed.csv</code>	CSV	20.26 MB
<code>tfidf_matrix.npz</code>	NumPy sparse	1.96 MB
<code>tfidf_vectorizer.pkl</code>	Pickle	0.19 MB
<code>tfidf_feature_names.npy</code>	NumPy	0.06 MB
<code>correspondence_dict.json</code>	JSON	0.01 MB

TABLE 4 – Fichiers exportés

### 5.2 Structure du CSV

Le fichier `train_preprocessed.csv` contient 4 colonnes :

- `doc_id` : Identifiant unique du document
- `raw_text` : Texte original
- `cleaned_text` : Texte nettoyé
- `lemmatized_text` : Texte lemmatisé

## 6 Utilisation pour la Partie B

### 6.1 Chargement des Données

#### 6.1.1 Textes Prétraités

```
import pandas as pd

# Charger le CSV
df = pd.read_csv('preprocessed_data/train_preprocessed.csv')

# Accéder aux différentes versions
raw_texts = df['raw_text']
cleaned_texts = df['cleaned_text']
lemmatized_texts = df['lemmatized_text']
```

#### 6.1.2 Matrice TF-IDF

```
from scipy.sparse import load_npz
import pickle
import numpy as np

# Charger la matrice TF-IDF
tfidf_matrix = load_npz('preprocessed_data/tfidf_matrix.npz')

# Charger le vectoriseur (pour transformer de nouveaux textes)
with open('preprocessed_data/tfidf_vectorizer.pkl', 'rb') as f:
    vectorizer = pickle.load(f)

# Charger les noms de features
feature_names = np.load('preprocessed_data/tfidf_feature_names.npy')

# Transformer un nouveau texte
new_text = ["your_new_text_here"]
new_vector = vectorizer.transform(new_text)
```

#### 6.1.3 Métadonnées

```
import json

# Charger le dictionnaire de correspondance
```

```
with open('preprocessed_data/correspondence_dict.json', 'r') as f:
    metadata = json.load(f)

# Accéder aux informations
n_docs = metadata['metadata']['n_documents']
preprocessing_steps = metadata['metadata']['preprocessing_steps']
```

## 6.2 Recommandations pour la Partie B

1. Utiliser les textes lemmatisés pour toute analyse sémantique
2. Réutiliser le vectoriseur TF-IDF pour transformer de nouveaux documents
3. Exploiter la matrice sparse pour économiser la mémoire
4. Consulter le dictionnaire de correspondance pour comprendre le pipeline

## 7 Accès aux Ressources

### 7.1 Liens Importants

- GitHub Repository :  
[https://github.com/635jack/Projet\\_Semantique\\_AB\\_BEM\\_FD\\_JG](https://github.com/635jack/Projet_Semantique_AB_BEM_FD_JG)
- Google Colab Notebook :  
[https://colab.research.google.com/github/635jack/Projet\\_Semantique\\_AB\\_BEM\\_FD\\_JG/blob/master/PartieA\\_Preprocessing.ipynb](https://colab.research.google.com/github/635jack/Projet_Semantique_AB_BEM_FD_JG/blob/master/PartieA_Preprocessing.ipynb)
- Dataset SciREX :  
<https://github.com/allenai/SciREX>

### 7.2 Instructions d'Exécution

#### 7.2.1 Sur Google Colab

1. Ouvrir le lien Colab ci-dessus
2. Exécuter toutes les cellules : Runtime Run all
3. Télécharger les fichiers exportés depuis preprocessed\_data/

#### 7.2.2 En Local

1. Cloner le repository :  
`git clone https://github.com/635jack/Projet_Semantique_AB_BEM_FD_JG`
2. Installer les dépendances :  
`pip install -r requirements.txt`
3. Exécuter le notebook :  
`jupyter notebook PartieA_Preprocessing.ipynb`



## 8 Conclusion

### 8.1 Résumé des Contributions

Ce travail a permis de :

- Implémenter un pipeline NLP complet et robuste
- Traiter l'intégralité du dataset SciREX (306 documents)
- Générer des représentations vectorielles de haute qualité
- Exporter les résultats dans des formats standardisés

### 8.2 Points Clés

1. **Pipeline complet** : Nettoyage Tokenization POS Lemmatization TF-IDF
2. **TF-IDF sur textes lemmatisés** : Garantit la cohérence du preprocessing
3. **Tous les documents traités** : 306/306 documents lemmatisés
4. **Formats réutilisables** : CSV, NPZ, Pickle, JSON

### 8.3 Perspectives pour la Partie B

Les données préparées sont prêtes pour :

- Extraction d'entités nommées (NER)
- Classification de documents
- Clustering sémantique
- Analyse de relations
- Modèles d'embeddings avancés

## Annexes

### A. Commandes Utiles

Vérifier la taille des fichiers :

```
ls -lh preprocessed_data/
```

Compter les lignes du CSV :

```
wc -l preprocessed_data/train_preprocessed.csv
```

### B. Dépendances Complètes

Voir `requirements.txt` dans le repository GitHub.