Rapport de Projet PJE : Analyse de Comportement avec Twitter

Binôme :
Mathis Lefevre
Sid Ahmed Ferroudj

» Accéder au Dépôt Git «

15 décembre 2025

Table des matières

1 Description générale du projet **2**
1.1 Description de la problématique . . . 2
1.2 Description générale de l'architecture . . . 2
1.3 Organisation au sein du binôme . . . 2

2 Détails des différents travaux réalisés **2**
2.1 Préparation et nettoyage des données . . . 2
2.2 Algorithmes de classification / clustering . . . 3
2.2.1 Approche par Mots-clés (Règles) . . . 3
2.2.2 k-Nearest Neighbors (k-NN) . . . 3
2.2.3 Classification Bayésienne (Naive Bayes) . . . 4
2.2.4 Clustering Hiérarchique . . . 4
2.3 Interface Graphique . . . 5

3 Résultats de la classification et analyse **5**
3.1 Méthodologie d'évaluation . . . 5
3.2 Résultats obtenus (Moyennes sur 5 plis) . . . 5
3.3 Analyse et Observations . . . 5

4 Conclusion **5**

1 Description générale du projet

1.1 Description de la problématique

L'objectif de ce projet est de concevoir une application capable d'analyser le "sentiment" (opinion) exprimé dans des tweets. La tâche consiste à classifier des textes courts, souvent bruités (argot, abréviations, émojis), en trois catégories : **Positif**, **Négatif** ou **Neutre**.

Le défi réside dans la nature non structurée du langage naturel et la brièveté des tweets, qui rendent difficile l'extraction de contexte. Nous avons exploré plusieurs approches : symbolique (Mots-clés), géométrique (k-NN), probabiliste (Naive Bayes) et exploratoire (Clustering).

1.2 Description générale de l'architecture

L'application a été développée en **Python** en utilisant une architecture modulaire séparant la logique métier de l'interface utilisateur.

- **Interface (Vue) :** Réalisée avec **Streamlit**. L'organisation se fait par onglets, permettant de suivre le flux de travail logique : *Chargement/Nettoyage* → *Annotation* → *Algorithmes* → *Comparaison*.
- **Logique Métier (Modèle/Contrôleur) :**
  - **Manager (`AlgorithmManager`) :** Un contrôleur central qui gère les instances des différents algorithmes. Il permet de changer d'algorithme dynamiquement sans redémarrer l'application.
  - **Wrappers :** Chaque algorithme (KNN, Bayes, etc.) est encapsulé dans une classe "Wrapper" qui standardise les méthodes `fit()`, `predict_one()` et `predict_batch()`. Cela permet une interchangeabilité totale des modèles.
  - **Modules spécialisés :** Le code est organisé en packages : `cleaning` (traitement de texte), `algorithms` (implémentation des modèles), `gui` (interface).

1.3 Organisation au sein du binôme

- **Mathis :** S'est concentré sur la classification probabiliste (Naive Bayes et ses variantes N-grammes), l'analyse non supervisée (Clustering Hiérarchique) ainsi que la mise en place du module d'analyse expérimentale (Validation Croisée).
- **Sid Ahmed :** A pris en charge l'implémentation de l'algorithme k-NN (k-Nearest Neighbors) et de l'approche par Mots-clés (dictionnaires), ainsi que leur intégration.
- **Commun :** Le développement de l'interface graphique (Streamlit), l'architecture globale de l'application et le débogage final ont été réalisés conjointement.

2 Détails des différents travaux réalisés

2.1 Préparation et nettoyage des données

La qualité des données étant critique, nous avons développé une classe `TweetCleaner` utilisant un pipeline de règles (`CleaningRule`).

- **Traitements appliqués :** Conversion en minuscules, suppression des URLs, Mentions (@user), Hashtags (conservation du mot optionnelle), marqueurs 'RT', nettoyage de la ponctuation et normalisation des espaces blancs.
- **Filtres avancés :** Suppression des tweets contenant à la fois des émojis positifs et négatifs (ambiguïté) et détection de la langue.
- **Problèmes rencontrés :**

— *Encodage :* Certains fichiers CSV utilisaient `latin-1`. Nous avons utilisé la librairie `charset_normalizer` pour corriger l'encodage à la volée.
— *Formatage :* Certains labels étaient lus comme des flottants ("4.0") au lieu d'entiers. Nous avons ajouté une conversion robuste `int(float(value))` lors du chargement.

2.2 Algorithmes de classification / clustering

2.2.1 Approche par Mots-clés (Règles)

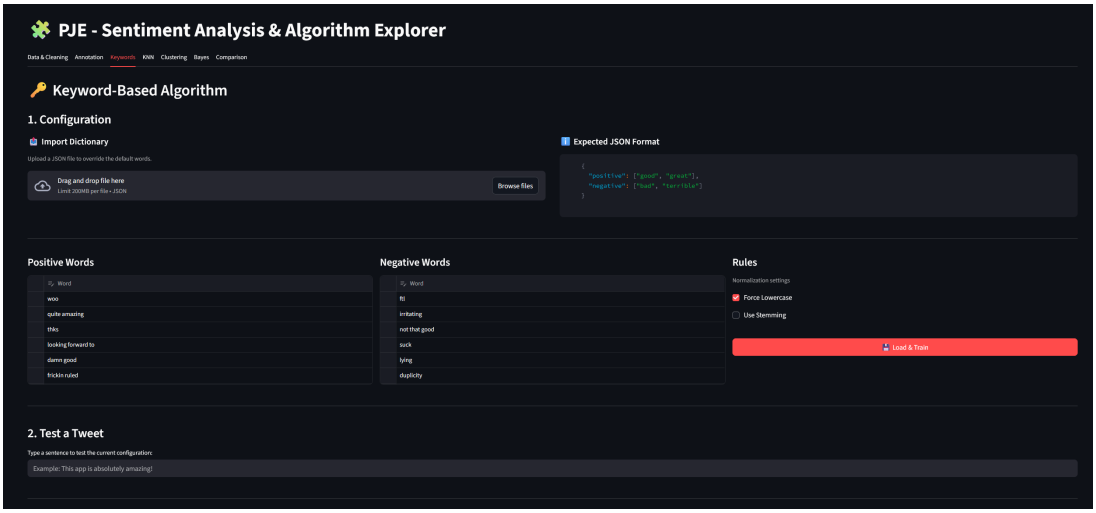

FIGURE 1 – Interface de configuration des mots-clés

Cette méthode déterministe repose sur deux lexiques (positif/négatif).
— **Fonctionnement :** Le score est la différence entre le nombre de mots positifs et négatifs.
— **Paramètres et Impact :**
  — *Richesse du dictionnaire :* Facteur critique. Un lexique pauvre augmente le taux de "Faux Neutres" (rappel faible).
  — *Normalisation :* La conversion en minuscules est essentielle pour la reconnaissance des mots.

2.2.2 k-Nearest Neighbors (k-NN)

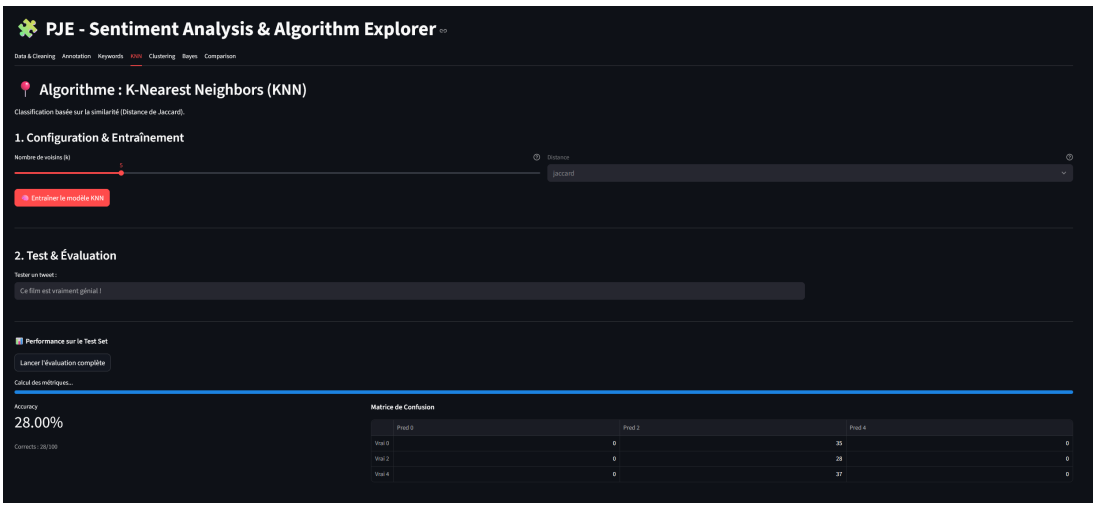

FIGURE 2 – Paramétrage et résultats du k-NN

— **Distance :** Nous utilisons la **Distance de Jaccard** $(1 - \frac{|A \cap B|}{|A \cup B|})$, idéale pour mesurer le chevauchement de mots sans biais de fréquence.
— **Paramètres et Impact :**
  — **Nombre de voisins (k) :**
    — Un k petit (1-3) rend le modèle sensible au bruit (overfitting).
    — Un k grand (15+) lisse la décision mais risque de manquer les spécificités locales (underfitting). Nous utilisons $k = 5$ par défaut.

2.2.3 Classification Bayésienne (Naive Bayes)

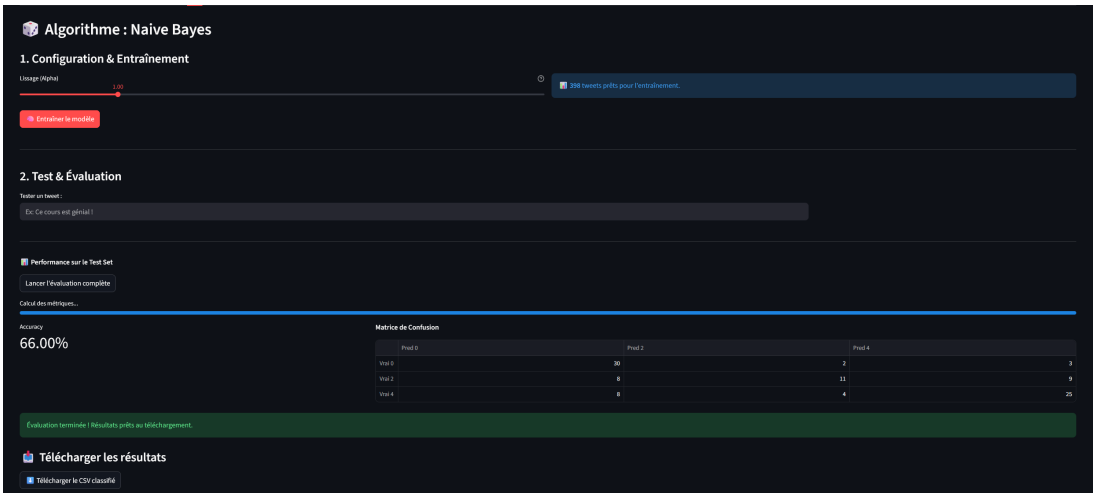

FIGURE 3 – Interface Naive Bayes et probabilités

— **Modèle :** Multinomial Naive Bayes adapté au texte.
— **Paramètres et Impact :**
  — **Lissage (α) :** Permet de gérer les mots inconnus. Une valeur $\alpha = 1.0$ généralise mieux qu'une valeur faible ($\alpha = 0.1$) qui sur-apprend le vocabulaire d'entraînement.
  — **N-grammes :** L'ajout des **Bigrammes** (paires de mots) améliore la performance ($\approx +2\%$) en capturant des contextes comme la négation ("not good"), invisibles pour les unigrammes seuls.

2.2.4 Clustering Hiérarchique

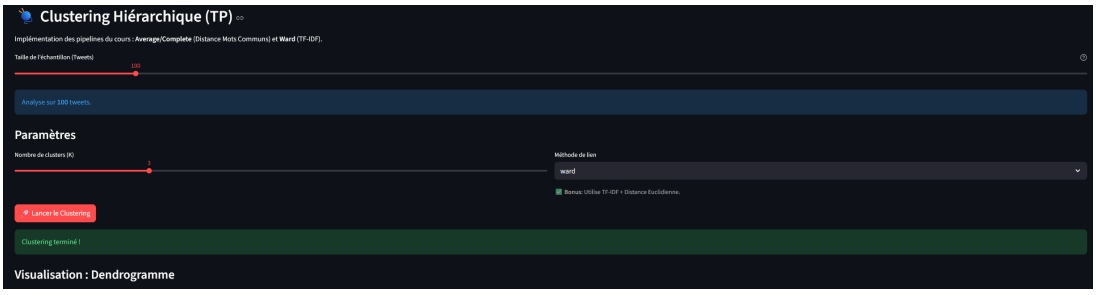

FIGURE 4 – Visualisation du Dendrogramme

— **Méthodes de Lien :**
  — *Average Linkage (Jaccard) :* Sensible aux chaînes de points, produit des clusters de tailles variables.

— *Ward Linkage (TF-IDF) :* Minimise la variance interne. Produit des clusters plus compacts et sphériques, souvent plus pertinents pour séparer les thématiques textuelles.

— **Nombre de clusters (k) :** Détermine le niveau de coupe de l'arbre.

2.3 Interface Graphique

L'interface Streamlit centralise toutes les opérations : upload dynamique, paramétrage via sliders, test interactif sur phrase manuelle, et téléchargement des résultats en CSV.

3 Résultats de la classification et analyse

3.1 Méthodologie d'évaluation

Nous avons implémenté une procédure de **Validation Croisée (K-Fold Cross-Validation)** avec $k = 5$ plis. Cette méthode garantit que le modèle est testé sur des données qu'il n'a jamais vues.

3.2 Résultats obtenus (Moyennes sur 5 plis)

Algorithme	Variante	Précision (Accuracy)
Naive Bayes	**Présence, Uni+Bi**	**$\approx$ 68.8%**
Naive Bayes	Présence, Unigramme	$\approx$ 66.8%
KNN	$k = 5$, Jaccard	$\approx$ 59.5%
Naive Bayes	Présence, Bigramme	$\approx$ 59.5%
Mots-clés	Dictionnaire	$\approx$ 55-60%

Table 1 – Comparaison des performances des algorithmes

3.3 Analyse et Observations

1. **Supériorité des N-grammes combinés :** La variante **Naive Bayes (Uni+Bi)** offre la meilleure performance. L'ajout des bigrammes permet de capter des contextes locaux (ex : "not good") que l'unigramme seul perd.
2. **Limites du Bigramme seul :** Utilisé seul, le modèle Bigramme performe mal car les paires de mots sont trop rares (sparse) dans des tweets courts.
3. **KNN vs Bayes :** Le KNN performe nettement moins bien que Bayes ($\approx$ 10 points d'écart). Cela s'explique par la "malédiction de la dimensionnalité" : deux tweets peuvent avoir le même sentiment sans partager exactement les mêmes mots.
4. **Présence vs Fréquence :** Sur des textes courts, un mot apparaît rarement deux fois. L'information de "Fréquence" n'apporte donc pas de gain significatif par rapport à la simple "Présence".

4 Conclusion

Ce projet nous a permis de mettre en œuvre une chaîne complète de traitement de données textuelles. Nous avons démontré que pour la classification de sentiments sur des tweets courts, l'approche probabiliste **Naive Bayes (avec Unigrammes + Bigrammes)** est la plus robuste. Les approches géométriques comme le k-NN se heurtent à la sparsité du vocabulaire. L'architecture modulaire mise en place permettrait d'intégrer facilement des modèles plus avancés à l'avenir.