Rapport de projet Outil de programmation

Réalisé par Kaoutar AKHSASS et Fatema ABOURA

DataSet

Les tweets ont été extraits de Twitter, et un marquage manuel a été effectué par la suite.
Les noms et prénoms d'utilisateur ont été codés pour éviter tout problème de confidentialité.

Colonnes :

1. **Location** → Localisation
2. **Tweet At** → Date du Tweet
3. **Original Tweet** → Tweet Original
4. **Label** → Étiquette

1. Prétraitement des données

- **Objectif** : Nettoyer les données textuelles et les préparer pour l'entraînement du modèle.
- **Étapes principales** :
    - Chargement des données.
    - Nettoyage du texte : suppression des caractères spéciaux, conversion en minuscules, suppression des stopwords, et lemmatisation.
    - Encodage des étiquettes (labels) avec `LabelEncoder` pour un traitement compatible avec les modèles.

2. Tokenisation et préparation des séquences

- **Objectif** : Convertir le texte en une forme numérique compréhensible par les modèles de deep learning.
- **Étapes principales** :
    - Utilisation de `Tokenizer` pour créer un vocabulaire basé sur les mots les plus fréquents.
    - Conversion des textes en séquences d'entiers (`texts_to_sequences`).
    - Remplissage (padding) des séquences pour garantir une longueur fixe (par exemple, 128 mots).

3. Division des données

- **Objectif** : Séparer les données en ensembles d'entraînement et de test pour évaluer les performances.
- **Étapes principales** :
    - Utilisation de `train_test_split` pour diviser les données en proportions 80/20 (ou autre ratio choisi).
    - Encodage des labels dans les deux ensembles.

4. Construction du modèle CNN

- **Objectif** : Construire un modèle de classification basé sur un réseau de neurones convolutif (CNN).
- **Architecture du modèle** :
    - **Embedding Layer** : Convertit les indices des mots en vecteurs denses (représentations continues).
    - **Convolutional Layer (Conv1D)** : Extrait les motifs locaux dans les textes.
    - **GlobalMaxPooling1D** : Réduit les dimensions tout en conservant les caractéristiques importantes.
    - **Dense Layers** : Apprend les relations complexes pour la classification.
    - **Dropout** : Ajouté pour éviter le surapprentissage.
    - **Output Layer** : Utilise une fonction d'activation `softmax` pour prédire les probabilités pour chaque classe.

5. Entraînement et validation

- **Objectif** : Ajuster les paramètres du modèle pour minimiser l'erreur de classification.
- **Étapes principales** :
    - Utilisation de `model.fit` pour entraîner le modèle avec les données d'entraînement.
    - Évaluation sur l'ensemble de validation.
    - Validation croisée (K-Fold) pour ajuster les hyperparamètres et vérifier la robustesse.

6. Évaluation des performances

- **Objectif** : Évaluer les prédictions du modèle sur les données de test.
- **Métriques calculées** :
    - Précision, rappel, F1-score : Fournissent une évaluation complète des performances.
- **Visualisation** :
    - Matrice de confusion : Affiche les erreurs de classification.
    - Courbe ROC (si applicable) : Montre le compromis entre le taux de faux positifs et de vrais positifs.

---

7. Optimisation et régularisation

- **Objectif** : Améliorer les performances et réduire le surapprentissage.
- **Techniques utilisées** :
    - Régularisation L2 dans les couches convolutives et denses.
    - Ajout de couches Dropout.