Approche par Réseaux Neuronaux Siamois (SNN) pour la Résolution d'Entités

Ce document décrit l'architecture et la méthodologie employées pour la résolution d'entités à l'aide de Réseaux Neuronaux Siamois (SNN). Le SNN est une méthode efficace pour les scénarios de correspondance complexe, utilisant une architecture de réseau neuronal conçue pour évaluer la similarité entre deux entrées.

1. Préparation des Données

1. **Chargement des Données**: Les données sont chargées depuis des fichiers CSV situés dans le répertoire `input_data/`.
2. **Nettoyage des Données**: Suppression des colonnes avec un nombre élevé de valeurs NaN (> 60%).
3. **Filtrage des Données**: Seules les colonnes nécessaires sont conservées, selon les mappings définis dans les fichiers JSON.
4. **Normalisation**: Les colonnes de chaînes de caractères sont converties en minuscules et les caractères non alphanumériques sont supprimés. Les adresses sont normalisées via la géocodification.

2. Mapping des Données

1. **Chargement du Dictionnaire de Mapping**: Lecture des règles de mapping depuis un fichier JSON.
2. **Application du Mapping**: Renommage des colonnes du DataFrame selon les règles de mapping.
3. **Filtrage des Colonnes**: Conservation uniquement des colonnes spécifiées dans le dictionnaire de mapping.

3. Normalisation des Données

1. **Prétraitement des Colonnes de Chaînes**: Conversion en minuscules et suppression des caractères non alphanumériques.
2. **Géocodification des Adresses**: Utilisation d'un service de géocodification pour normaliser les adresses.
3. **Exportation des Résultats**: Sauvegarde des données normalisées dans un fichier CSV.

4. Architecture du Modèle SNN

Le modèle SNN utilise deux sous-réseaux identiques pour traiter deux entrées en parallèle et calculer leur similarité.

1. **Réseau de Base**:
    - **Couche d'Entrée**: Reçoit les vecteurs de caractéristiques des entités.

- **Convolution 1D**: Deux couches de convolution pour extraire les caractéristiques importantes.
- **MaxPooling**: Deux couches de pooling pour réduire la dimensionnalité et capturer les caractéristiques essentielles.
- **Couche Dense**: Deux couches denses pour réduire davantage la dimensionnalité et apprendre des représentations complexes.
- **Dropout**: Utilisée pour éviter le surapprentissage en désactivant de manière aléatoire certaines unités pendant l'entraînement.
- **Couche de Sortie**: Génère des embeddings pour les entités d'entrée.

2. **Module de Distance**:
   - **Calcul de la Distance Euclidienne**: Calcule la distance entre les embeddings des deux entités.
   - **Couche Lambda**: Applique la fonction de distance pour obtenir un score de similarité.
3. **Perte Contrastive**:
   - Utilisée pour minimiser la distance entre les paires similaires et maximiser la distance entre les paires dissimilaires.