Contexte du projet

Développer un système de recommandation d'hôtels en utilisant des **Graph Neural Networks (GNN)**. Vous disposez de deux jeux de données comprenant des avis clients et des notes :

1. Le premier jeu de données est obtenu via web scraping, collecté par vos collègues.
2. Le deuxième jeu de données est le **HARD (Hotel Arabic-Reviews Dataset)**, disponible sur **GitHub**.

Vous devrez utiliser l'un des deux datasets pour construire votre modèle.

Objectifs du projet

1. Effectuer une **analyse de sentiment** sur les avis afin d'enrichir le dataset choisi.
2. Combiner les scores d'analyse de sentiment avec les notes des utilisateurs pour créer des **scores finaux** plus représentatifs, et l'utiliser pour entraîner un système de recommandation basé sur les **GNN**.

Étapes à suivre :

1. Prétraitement des données

· Nettoyage et préparation des données

· **Analyse de sentiment** : Utiliser des modèles de type transformer récents, tels que AraBERT ou CAMeL-BERT, pour l'analyse des sentiments des avis.

· **Calcul des scores finaux** :
Score final=(0.5×score normalise)+(0.5×sentiment-score)×max-rating

Cette formule combine les scores normalisés des utilisateurs avec les scores d'analyse de sentiment (+1 avis positif et -1 avis négatif).

· **Division des données** : Séparer le jeu de données en deux sous-ensembles :

✔ **Ensemble d'entraînement (80%)** : Pour entraîner le modèle.
✔ **Ensemble de test (20%)** : Pour évaluer les performances du modèle après l'entraînement.

2. Construction du graphe

L'entrée pour un modèle GNN est un **graphe** composé de **nœuds** et **d'arêtes**, enrichis par des informations supplémentaires.

 Les nœuds :

· **Utilisateurs** : Chaque utilisateur est représenté par un nœud.
· **Hôtels** : Chaque hôtel est également représenté par un nœud.

Ces nœuds peuvent contenir des **attributs supplémentaires**, comme :

- Pour les **hôtels** : localisation, catégorie, prix moyen, etc.
- Pour les **utilisateurs** : identifiant, historique des avis, etc.

 Les arêtes :

- Les **arêtes** relient les utilisateurs aux hôtels qu'ils ont évalués, et **ces arêtes sont pondérés** par les scores finaux calculés précédemment (qui combinent l'analyse de sentiment des avis et les notes des utilisateurs).

3. Entraînement du modèle GNN

- Testez différentes variantes de GNN pour la recommandation, telles que **GCN**, **GraphSAGE**, et **GAT**.
- **Suivi de l'apprentissage** : Pour chaque modèle GNN, présentez la courbe d'erreur durant l'apprentissage pour détecter tout **sur-apprentissage**.
- **Comparaison du temps d'apprentissage** : Évaluez également le temps d'apprentissage nécessaire pour chaque modèle afin de choisir celui qui est le plus performant.

4. Évaluation des performances

Sur la base de test (20%) , évaluez la performance de chaque système de recommandation à l'aide des métriques suivantes :

- **RMSE** (Root Mean Square Error)
- **MAE** (Mean Absolute Error)
- **Précision**, **Rappel**, et **F-score**

5. Variantes de GNN à explorer

- **GCN (Graph Convolutional Network)** : Une des variantes les plus classiques et simples à mettre en œuvre, utilisant des convolutions sur les graphes pour agréger les informations des voisins.
- **GraphSAGE (Graph Sample and Aggregation)** : Utilise un échantillonnage des voisins pour gérer des graphes très grands.
- **GAT (Graph Attention Network)** : Applique un mécanisme d'attention pour donner plus de poids à certains voisins, améliorant ainsi l'agrégation des informations.
- **Autres variantes** : Vous pouvez également explorer des architectures GNN plus générales ou avancées qui combinent différentes techniques d'agrégation.

Livrables attendus :

1. **Code source complet** (avec l'executable) pour le prétraitement, l'analyse de sentiment, et la création du système de recommandation basé sur GNN.
2. **Rapport détaillé** qui couvre :
    - Les étapes du projet

- Les paramètres détaillés de chaque modèle GNN (fonction d'activation, méthode d'agrégation, nombre de couches, etc.)
- Les performances obtenues

3. **Comparaison des variantes de GNN** avec des graphiques montrant la performance de chaque modèle, le temps d'exécution et l'évolution de l'erreur pendant l'entraînement.

4. **Discussion riche**

Langue de rédaction : vous pouvez utiliser l'anglais. C'est fortement recommandé.

Remarque importante

Voici comment vous faites la **Division des données**

1. **Séparer les données** (80/20) :
   - 80% : données d'entraînement
   - 20% : données de test
2. **Validation croisée** sur l'ensemble d'entraînement (80%) : Sur les 80% des données d'entraînement, appliquez la **validation croisée**. Cela consiste à diviser cet ensemble d'entraînement en plusieurs sous-ensembles (5 ou 10 fold), et pour chaque pli (fold), entraînez vos modèle sur les autres plis et le testes sur le pli restant. Cela permet d'ajuster les hyperparamètres du modèle et non de tester sa performance.

   La validation croisée aide à éviter le sur-apprentissage en s'assurant que le modèle se généralise bien sur différentes parties des données d'entraînement.

3. **Tester les performances sur les données de test** : Une fois que le modèle a été formé et validé (en ajustant les hyperparamètres via la validation croisée), il sera testé sur les **20% de données restantes** (ensemble de test), qui n'ont jamais été utilisées pendant l'entraînement. C'est là ou on mesure les performances du modèle avec des métriques comme **RMSE**, **MAE**, **précision**, **rappel.**