Rapport sur les Travaux Pratiques (TP) : TD-03 Spark - Amis en Commun & Mini Projet (Suite du TP3)

Introduction

Les deux travaux pratiques présentés dans ce rapport portent sur l'utilisation de **Apache Spark** pour résoudre un problème d'analyse de réseau social : la détection des **amis communs entre utilisateurs**.

- **TD-03 Spark – Amis en Commun** : implémentation Scala avec Spark pour générer la liste des amis communs entre toutes les paires d'utilisateurs.
- **Mini Projet (Suite du TP3)** : réalisation d'une solution similaire en Python (**PySpark**) pour identifier les amis communs **entre deux utilisateurs spécifiques** : Mohamed (ID 2) et Sidi (ID 1).

Ces TP visent à développer des compétences en :

- Programmation distribuée avec Spark.
- Paradigmes MapReduce dans un environnement Big Data.
- Manipulation de graphes sociaux sous forme de fichiers texte.
- Développement de scripts performants pour l'analyse relationnelle.

I. TD-03 : Spark – Problème des Amis en Commun

Objectif

Implémenter un programme Spark en **Scala** permettant de détecter la liste des amis communs entre chaque paire d'utilisateurs dans un graphe social représenté par un fichier texte : **soc-LiveJournal1Adj.txt**.

Données d'entrée

- **Fichier** : soc-LiveJournal1Adj.txt
- **Format** :
- <User_ID><TAB><Liste_Amis_IDs>

où la liste des amis est séparée par des virgules.

- Les relations sont **mutuelles** (non orientées) : si A est ami avec B, alors B est ami avec A.

Approche algorithmique

1. **Chargement des données**
2. val data = sc.textFile("soc-LiveJournal1Adj.txt")
3. **Nettoyage et préparation**
    - Suppression des lignes mal formatées.
    - Extraction des paires (userID, friendsList).
4. **Génération des paires d'amis**
    - Pour chaque utilisateur, générer toutes les combinaisons d'amis possibles.
    - Tri systématique des paires sous la forme (minID, maxID) afin d'éviter les doublons.
5. **Calcul des amis communs**
    - Utilisation de transformations Spark comme flatMap pour créer les paires.
    - Réduction par clé et intersection des listes d'amis pour trouver les amis communs.
6. **Formatage et sauvegarde**
    - Conversion des résultats au format souhaité.
    - Sauvegarde des résultats dans HDFS ou en local.

Exemple de sortie

<userA> <userB> <friend1, friend2, ...>

Analyse du code fourni

Le code Scala utilise efficacement plusieurs transformations Spark :

- map
- flatMap
- filter
- reduceByKey
- collect

Il inclut aussi des tests spécifiques, par exemple pour rechercher les amis communs entre les utilisateurs 0 et 4.

II. Mini Projet – Suite du TP3 Spark

Objectif

Développer un projet en **PySpark** afin de trouver la liste des **amis communs entre deux utilisateurs spécifiques** : Mohamed (ID 2) et Sidi (ID 1), en se basant sur le même fichier soc-LiveJournal1Adj.txt.

Tâches réalisées

1. **Chargement des données**
    - Lecture du fichier texte avec SparkContext.
    - Nettoyage des lignes mal formatées.
2. **Génération des paires d'amis**
    - Création de paires (minID, maxID) pour éviter les duplications.
    - Transformation de chaque ligne en une liste de paires d'amis.
3. **Calcul des amis communs**
    - Agrégation des amis par utilisateur.
    - Jointure des amis pour chaque paire d'utilisateurs.
    - Intersection des listes d'amis pour extraire les amis communs.
4. **Filtrage des résultats**
    - Recherche spécifique des amis communs entre Mohamed (ID 2) et Sidi (ID 1).
    - Normalisation de la paire sous la forme (1, 2).
5. **Affichage final**
   Format des résultats :
6. 1<Nom1>2<Nom2> liste_amis_communs
7. **Livrables**
    - Code bien structuré et commenté.

- Documentation (README.md) détaillant :
  - Objectifs
  - Étapes techniques
  - Instructions d'exécution
- Dépôt GitHub contenant :
  - Code source
  - Résultats

Technologies utilisées

- **Apache Spark (PySpark)**
- **Python**
- **Git / GitHub** pour le versioning et la livraison

III. Comparaison des deux approches

Critère	TD-03 (Scala/Spark)	Mini Projet (Python/PySpark)
Langage	Scala	Python
Objectif	Calcul de tous les amis communs entre toutes les paires	Calcul ciblé pour deux utilisateurs spécifiques
Données	soc-LiveJournal1Adj.txt	soc-LiveJournal1Adj.txt
Structure	Programme complet avec tests intégrés	Projet modulaire avec documentation
Livrable	Code + explications	Projet sur GitHub + README

IV. Compétences acquises

Compétences techniques

- Maîtrise des transformations Spark :

    - map, flatMap, filter, reduceByKey, join
- Gestion des données semi-structurées (fichiers texte).
- Programmation fonctionnelle appliquée à l'analyse de graphes sociaux.
- Manipulation de structures de données complexes (listes, tuples, paires).
- Développement en PySpark dans un contexte Big Data.

Compétences méthodologiques

- Structuration d'un projet Big Data.
- Rédaction d'une documentation technique claire et détaillée.
- Utilisation des outils de versioning (Git/GitHub).
- Optimisation et analyse des performances Spark.

Conclusion générale

Ces deux travaux pratiques illustrent parfaitement l'application d'**Apache Spark** à des problématiques concrètes d'analyse de réseaux sociaux. Spark se révèle particulièrement efficace pour traiter des volumes importants de données grâce à son modèle distribué.

- Le **TD-03** offre une vue d'ensemble sur la détection des amis communs avec une implémentation en Scala, langage natif de Spark.
- Le **Mini Projet** apporte une solution plus ciblée en **PySpark**, adaptée à des analyses spécifiques et à une intégration aisée dans des projets Python modernes.

Ces exercices constituent une excellente base pour explorer des domaines plus avancés, tels que :

- Analyse de graphes sociaux.
- Recommandation collaborative.
- Intelligence artificielle appliquée aux réseaux sociaux.

Ils développent également des compétences transversales précieuses telles que la gestion de projet, la collaboration et l'intégration continue.