Beleg 3

Das Ziel dieser Belegarbeit ist es die Grundlagen der Programmierung von Spark Anwendungen mittels Resilient Distributed Datasets (RDDs) zu üben. Der Beleg ist in 3 Teile geteilt: Der Erste Teil sollte als ein Spark Tutorial dienen, in dem grundlegende RDD Operationen auf schon bekannte, simple Probleme angewendet werden sollten. Im zweiten Teil sollen diverse Statistiken für einen kleinen Twitter Datensatz berechnet werden. Der Twitter-Datensatz beinhaltet Tweets unserer Bundestagsabgeordneten aus dem Zeitraum Anfang Januar bis Ende Mai. Im letzten Teil soll der Page Rank-Algorithmus unter Verwendung von Spark implementiert werden. Dieser kann dann auf einen kleinen aus Wikipedia gescrapten Datensatz zum Thema Star Wars angewendet werden.

Aufgabe 1:

Vervollständigen sie die Funktionen der Klassen *Section0, Section1, Section2* aus dem *package introduction* und beachten Sie dabei die Anweisungen in den Kommentaren. In der Einführung geht es um die Erzeugung eines Spark-Kontextes und das Kennenlernen von einfachen Actions und Transformationen.

Aufgabe 2:

In dieser Aufgaben sollen Sie für einen kleinen Twitterdatensatz, der die Tweets der Bundestagsabgeordneten aus dem Jahre 2021 enthält, Statistiken erstellen.

a) Machen Sie sich mit den Klassen und Methoden aus den Paketen twitter und IOUtils vertraut und implementieren sie die Methode parse der Klasse TwitterUtilities. Sie soll aus einem String im Twitter JSON Format einen Tweet erzeugen. Welche Informationen extrahiert werden sollen, entnehmen Sie bitte aus dem Kommentar über der Methode. Testen sie die Methode anschließend mit Hilfe der Klasse ParsingTest.

b) Implementieren sie die fehlenden Methoden der Klasse TwitterAnalyzer d.h. *countTweetsPerUser, findPartyNames, findMemberNamesPerParty, countTweetsPerParty, countMembersPerParty, averageTweetsPerPartyAndMember, tenMostFamousHashtags, fiveMostFamousHashtagsPerParty* und *getDaysAndFrequenciesOfHashtag,* . Informationen und Tipps entnehmen Sie bitte aus den Kommentaren über den jeweiligen Methoden. Testen sie diese Anschließend mit Hilfe der Klasse TwitterAnalyzerTest.

c) Schauen Sie sich die Klasse TwitterAnalyzerDataFrames an und versuchen Sie die Funktionsweise von Spark-SQL nachzuvollziehen. Implementieren Sie die Funktion countTweetsPerUser. Konsultieren Sie bei Interesse die Dokumentation zum Thema SparkSQL(https://spark.apache.org/docs/latest/sql-getting-started.html).

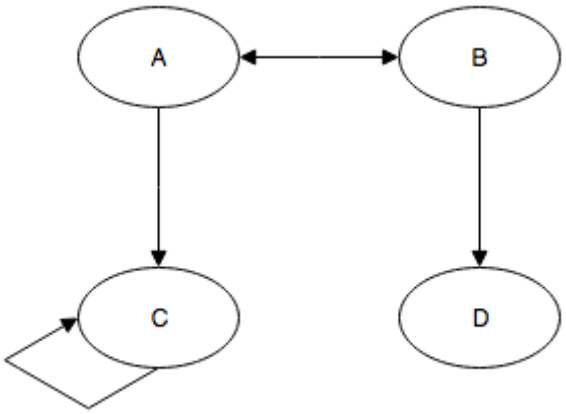
d) Denken Sie sich selbstständig mindestens drei Anfragen aus und implementieren Sie diese entweder in der Klasse TwitterAnalyzer oder TwitterAnalyzerDataFrames. Verwenden Sie dabei entweder RDDs oder DataFrames (SparkSQL). Implementieren Sie nicht nur die Methode sondern schreiben Sie dafür auch Tests, in dem Sie die entsprechende Testklasse erweitern. Bei Interesse können Sie auch den Cluster verwenden sowie die Tweets der gesamten Legislaturperiode. Auf

Anfrage stelle ich Sie Ihnen gerne bereit, bzw. kümmere mich um Zugriffsrechte für den Cluster.

Aufgabe 3:

In dieser Aufgabe soll der PageRank-Algorithmus unter Verwendung von Spark implementiert werden. Der PageRank Algorithmus ist ein iteratives Verfahren, was ein Netz von Webseiten verwendet, um den Vertrauenswürdigkeit (PageRank Wert) einer Webseite zu berechnen. Der PageRank Wert einer Webseite, sagt aus, wie hoch die Wahrscheinlichkeit ist, dass sich ein Benutzer nach beliebigen vielen Klicks im Netz auf dieser befindet. Der Algorithmus wird in der Lehreinheit erklärt, in diesem Text wird auf die Funktionsweise der Methoden eingegangen. Weitere Informationen können Sie auch den Tests in der Klasse PageRankTests entnehmen.

a) Im ersten Schritt muss die Datenstruktur aufgebaut werden. Dies erfolgt in der Methode extractLinksFromPages. Eine Page (Klasse models.Page) wird durch einen eindeutige pageid, einen eindeutigen Namen sowie eine Menge von Links, auf die die Webseite zeigt (Outlinks). Jeder Link wird dabei repräsentiert über einen Namensraum (Namespace) sowie einen Titel. Der im Bild gezeigte Webgraph wird folgendermaßen repräsentiert:



Das oben dargestellte Netz beinhaltet 4 Webseiten - A, B, C und D. Die Seite “A” verlinkt Seite ”C”, „C“ hat einen Link auf sich selbst, „A“ und „B“ verlinken sich gegenseitig und „B“ verlinkt „D“. Das Netz kann tabellarisch folgendermaßen dargestellt werden:

A ----> (B,C)
B ----> (A,D)
C ----> (C)
D ----> ()

Links vom Pfeil sind die Pages, und Rechts ihre Links (die Seiten, auf die sie verlinkt). Beinhalten Pages keine Verlinkungen (wie D), so ist die Menge leer. Genau diese Darstellung soll in der Methode extractLinksFromPages extrahiert werden.

Die Funktion bekommt ein RDD von Pages als Parameter und hat als Ergebnis ein RDD von Tupeln (String, List[String]), wobei der String der Titel der Page ist und die Liste die Titel der Links beinhaltet, auf die die Page zeigt.

Die Extraktion kann mit den folgenden Schritten vorgenommen werden (die Vorgehensweise muss nicht angewendet werden):

1. Erzeugen Sie Tupel Wert-Paare der Form: Title -> Set(links.titles)
2. Extrahieren Sie alle Links aus den Seiten und erstellen Sie für jeden Link einen Eintrag der Form link.title -> Set() (Somit wird sichergestellt, dass auch alle Seiten berücksichtigt werden).
3. Vereinigen Sie die beiden Mengen aus 1 und 2 und reduzieren Sie diese über den Schlüssel.

b) Machen Sie sich mit der Methode computePageRank vertraut – sie implementiert das PowerIteration-Verfahren (Potenzmethode) zur Ermittlung der Eigenvektoren einer Matrix. Sie ist rekursiv und benutzt die folgenden Methoden:

- *computeContributions:* Ermittelt den Wertteil, der sich aus den Outlinks der einzelnen Seiten ergibt.
- *computeNewRanks:* Berechnet den neuen Rank, in dem alle Beiträge zu einem neuen Rank summiert werden und dann der Teleportationsanteil dazugerechnet wird.
- *computeDifference:* Ermittelt die Differenz der beiden Vektoren (L1-Norm).

Das Verfahren wird gestartet, in dem ein Startvektor erzeugt wird, bei dem für jede Komponente der Wert 1/Anzahl der Komponenten gesetzt wird. Dann werden so lange neue Ranks erzeugt, bis die Differenz zwischen dem alten Vektor und dem neuen Vektor sich innerhalb einer zu spezifizienden ε-Umgebung befindet.

Bitte beachten Sie, dass für die Implementierung des Verfahrens die Matrizen nur in Sparse-Repräsentation vorliegen, d.h. es ist nur ein Wert da, wenn auch ein Link zwischen den Seiten vorliegt.

c) Für die Erklärung der computeContributions wird noch einmal der oben dargestellte Graph herangezogen.

A ----> (B,C)
B ----> (A,D)
C ----> (C)
D ----> ()

Daraus ergeben sich die folgenden Zusammenhänge für die Berechnung der neuen Ränge:
A wird aus $1/2*r_B$ und $1/4*r_D$ berechnet
B wird aus $1/2*r_A$ und $1/4*r_D$ berechnet
C wird aus $1*r_C$, $1/2\ r_A$ und $1/4*r_D$ berechnet
D wird aus $1/4*r_D$ berechnet
(Der Wert $1/4\ r_D$ ergibt sich aus der Teleportation, weil D keine Outlinks besitzt)

Diese sogenannten Beiträge sollen aus dem alten Rank sowie der Tabelle berechnet werden. Verbinden Sie hierzu den “alten” Rank mit den Kanten (join-Operator) und berechnen Sie somit die neuen Werte. Damit kein Knoten bei der Berechnung verloren geht (wenn es keine Links gibt, die auf ihn zeigen), fügen Sie einfach einen Standardwert, der den Beitrag 0 hat, hinzu (<Knotenname>, 0). Diesen Beitrag können Sie bei jedem bearbeiteten Knoten hinzufügen, der kein Link auf sich selbst hat (so wird es im Test abgeprüft). Daraus ergeben sich die folgenden Beiträge für unser Beispiel:

//Contributions from A
*("B", 0.5 * 0.25), ("C", 0.5 * 0.25), ("A", 0),*
//Contributions from B

*("A", 0.5 * 0.25), ("D", 0.5*0.25), ("B", 0),*
//Contributions from C
*("C", 0.25 * 1),*
// Contributions from D
*("A", 0.25 * 0.25), ("B", 0.25 * 0.25), ("C", 0.25 * 0.25), ("D", 0.25 * 0.25)*

d) In der Funktion computeNewRanksFromContributions werden die neuen Ranks berechnet, in dem alle Beiträge, die zu einer Page gehören, aufsummiert werden. Der resultierende Wert wird dann mal dem Faktor zur Teleportation genommen und der konstante Wert, der sich aus der Teleportation ergibt dazu addiert.

e) Die Funktion computeDifference soll die L_1-Norm für den alten und den neuen Vektor berechnen, d.h. die absoluten Differenzen der jeweiligen Vektorkompenten ermitteln und diese dann aufsummieren.

f) Sind die Tests alle erfolgreich, können Sie die Applikation pagerank.Application starten. Sie berechnet für den StarWars-Datensatz die Ränge der einzelnen Seiten. Die Laufzeit kann jedoch mehrere Stunden betragen.

Die zweite Belegarbeit kann in Teams (bis 2 Personen) bearbeitet werden. Alle Teilnehmer müssen ein lauffähiges Projekt bis zum 28.06.2021 23:59 Uhr abgeben. Die Belegarbeit muss in der darauf folgenden Übung am 29.06.2021 präsentiert werden.

Literatur:
Der Algorithmus wird ausführlich im Buch Mining Massive Datasets in Kapitel 5 – Link Analysis beschrieben, online kostenlos verfügbar unter: http://mmds.org/