Praktikum 5 GraphFrames – Graph Analytics

Ziel des Praktikums ist das Verständnis für Graph-Analysen und den Einsatz von Graph-Algorithmen auf horizontal skalierenden Systemen. Zum Einsatz kommt das Framework *GraphFrames*, das auf Spark und der Komponente *Spark SQL and Data Frames* basiert.

Zur Anwendung von Graph-Analysen wird der **Social Network Graph Pokec** (s.u.) zur Verfügung gestellt. Alle Implementierungen erfolgen in PySpark.

Als Codebeispiele zur weiteren Bearbeitung liegen ein **GraphGenerator** sowie eine Implementierung des **ConnectedComponents-Algorithmus** im Notebook zu Praktikum 5 zur Verfügung.

Visualisiert werden die Graphen mit dem **Visualisierungstool Gephi**.

Vorbereitung

Informieren Sie sich im Foliensatz **BDA_GraphFrames (Kapitel 5)** über die grundlegenden Konzepte des Frameworks *GraphFrames* (dieser Foliensatz wird auch im Rahmen der Vorlesung kurz vorgestellt). Ergänzend zu diesem Foliensatz wird das **Notebook GraphFrames** mit der Pyspark-Implementierung der entsprechenden Beispiele zur Verfügung gestellt, das Sie sich kopieren können zur eigenen Nutzung.

Installieren Sie ggf. das **Visualisierungtool Gephi** auf Ihrem eigenen Rechner (das Tool steht außerdem auf allen Laborrechnern zur Verfügung): https://gephi.org/users/download/

Machen Sie sich vertraut mit den **Pokec-Daten** (s.u.).

Aufgabe 1 muss vollständig implementiert zum Praktikumsbeginn vorliegen.

Pokec Social Network (Slovakia): Schema und Hive-Tabelle
Informationen zur Generierung der anonymisierten Daten sowie der zur Verfügung gestellten Edges und zum **Schema** der Vertices finden Sie hier:
https://snap.stanford.edu/data/soc-pokec-readme.txt

Die Menge der Vertices können Sie als DataFrame aus der Hive-Tabelle ***pokecVertices*** auslesen, die Menge der Edges aus der Hive-Tabelle ***pokecEdges***.

Datenquelle: https://snap.stanford.edu/data/soc-Pokec.html

L. Takac, M. Zabovsky. Data Analysis in Public Social Networks, International Scientific Conference & International Workshop Present Day Trends of Innovations, May 2012 Lomza, Poland.

Durchführung des Praktikums

Loggen Sie sich auf Zeppelin ein und kopieren Sie das Notebook praktikum5/_istaccount. Setzen Sie die Variable *currUser* auf Ihren Benutzernamen.

Aufgabe 1 – Graph Analysen auf dem Social Network-Graph Pokec

a) Lesen Sie die Vertices und die Edges des Pokec-Graphen aus den zur Verfügung gestellten Tabellen als DataFrames ein und erstellen Sie damit eine **GraphFrame-Instanz, deren Vertices alle Properties** besitzen. Generieren Sie ein **zweite GraphFrame-Instanz, deren Vertices nur die Vertex-ID** besitzen.

Bearbeitungsdatum: 12.06.2019 · Mi5y, 13.06.2018 · Do5y und 19.06.2019 · Mi6x

→ Die Teilaufgaben b) – f) führen Sie auf dem Graphen ohne Vertex-Properties durch.

b) Visualisieren Sie die **Gradverteilung des Pokec-Graph**, und ermitteln Sie die IDs der großen „Hubs".

c) Die Hive-Tabelle *is_pokeclpa10* enthält das Ergebnis einer LPA-Anwendung auf den Pokec-Graphen mit 10 Iterationen (**LPA = Label Propagation Algorithm** – vgl. auch Kapitel 5).

   In welchen Communities (= gleiches Label) sind die drei großen Hubs des Pokec-Graphen enthalten?

d) Wählen Sie die größte Community, die mindestens einen der drei großen Hubs enthält und extrahieren Sie diese Community als **Subgraph**.

   Untersuchen Sie für ausgewählte, isolierte Vertices, wie hoch deren Grad im Ausgangsgraph ist, und erklären Sie, warum deren Grad in der Community = 0 ist.

e) Ermitteln Sie die **Gradverteilung** in der Community.

f) Ermitteln Sie den **Clustering-Koeffizienten** der Community und interpretieren Sie den Wert.

g) Generieren Sie nun einen **Property-Graphen für die extrahierte Community**, der alle Properties des Pokec-Graphen enthält.

h) Lassen Sie sich für einen beliebigen Vertex zunächst alle Properties ausgeben – wählen Sie am besten einen Vertex mit einer hohen *completion_percentage* und einem nicht zu großen Grad.

   Generieren Sie dann für **diesen Vertex und alle direkt befreundeten Vertices** mit den verbindenden Kanten einen Subgraph. Reduzieren Sie dabei die Properties auf einige wenige Properties Ihrer Wahl, sodass eine Visualisierung sinnvoll möglich ist.

   Generieren Sie für diesen Graph eine DOT-Datei und visualisieren Sie diese im svg-Format (vgl. hierzu den Code im Notebook).

Aufgabe 2 – Analysen auf synthetisch generierten Graphen

a) Generieren Sie mit Hilfe des vorgegebenen Codes im Zeppelin-Notebook die **synthetischen Daten für zwei verschiedene Graphen**. Geben Sie dabei jeweils einen passenden Pfad in Ihrem Benutzerverzeichnis an.

b) Visualisieren Sie die Verteilung der Vertex-Degrees in Zepplin.

c) Laden Sie sich die .dot Dateien über den Explorer (http://141.100.62.85:50070/explorer.html) herunter und importieren Sie diese in Gephi. Wählen Sie beim Import als Graphen-Typ ungerichtet und für „Strategie zur Kantenverschmelzung" minimum.

d) **Visualisieren** Sie die Graphen in Gephi und wählen Sie das „Yifan-Hu" Layout auf der linken Seite im Reiter Übersicht. Wie unterscheiden sich die beiden Graphen strukturell grundsätzlich?

e) Generieren Sie drei PowerLaw-Graphen mit je 100, 1000 und 10000 Knoten. Wie lange dauert jeweils der Aufruf von connectedComponents()? https://graphframes.github.io/graphframes/docs/_site/api/python/graphframes.html. Wie würden Sie die Laufzeiten erklären?

f) Schauen Sie sich den Code der Methoden *generateRandomEdges* und *generatePowerEdges* genauer an. Diese rufen wiederum unterschiedliche Methoden zur Generierung der Kanten auf. Was genau passiert in den beiden unterschiedlichen Generatormethoden im Hinblick auf das Generieren der Kanten? Wie funktioniert das Generieren und wie wird dieses durch die Parameter beeinflusst?

g) Generieren Sie über die Veränderung des Parameters für den PowerLaw-Generator einen Graphen mit 1000 Knoten, der mehrere Connected Components mit möglichst mehreren Knoten in jeder Component enthält und visualisieren Sie diesen in Gephi.

Aufgabe 3 – Fehlersuche zur Implementierung von ConnectedComponents

Schauen Sie sich die Implementierungen der Funktion myWCC zur Bestimmung der weakly connected components an und versuchen Sie diese zu verstehen. Dokumentieren Sie Ihr Verständnis entsprechend. Eine der letzten beiden Codezeilen vor dem *return* der Methode enthält einen Fehler, der den korrekten Ablauf des Algorithmus verhindert. Korrigieren Sie diesen. Testen Sie die Berechnung der Connected Components und validieren Sie die Ausgabe z.B. durch Gegenüberstellung der Ergebnisse in Gephi oder durch die Verwendung der GraphFrames-eigenen Methode *connectedComponents.* Dokumentieren Sie Ihre Validierung.

Testat: Sie erhalten das Testat, wenn Sie den Dozenten die Ergebnisse zu allen Aufgabenstellungen in zufriedenstellendem Detailgrad am Ende des Praktikums zeigen und erklären können.