Universität Hamburg
Department Informatik
Wissenschaftliches Rechnen
Dr. J. Kunkel, H. Hatzel, J. Gresens

Übungsblatt 8 zur Vorlesung
Big Data Analytics
im WiSe 2015/2016
Abgabe: 11.12.2015, 420 P

Sollten Probleme auftauchen, wenden Sie sich bitte an die Mailingliste der Veranstaltung:

`bd1516@wr.informatik.uni-hamburg.de`

1 Datenimport (90 P)

1.1 Einrichtung

Die neuste Neo4j Version befindet sich in `/home/bigdata/neo4j`. Starten werden wir Neo4J auf ABU1. Zur Vorbereitung erstellen Sie einen Ordner entsprechend Ihrer Gruppen-Nummer in `/tmp` und extrahieren Sie dort die bereitgestellte neo4j Version. Ändern Sie vor dem Starten von Neo4j in der Konfigurations-Datei `neo4j-community-2.3.1/ conf/neo4j-server.properties` den Port für HTTPS und HTTP als auch die Database Location. Eine geänderte Beispielkonfiguration finden Sie unter `/home/bigdata/neo4j/`
Verwenden Sie den Port 100<ZweistelligeGruppenNr>0 also Beispielsweise für Gruppe 9: 10090 und für Gruppe 19: 10190. Die letzte Ziffer können Sie selbst innerhalb Ihrer Gruppe festlegen bzw. für HTTP und HTTPS. Den Datenbank-Server können Sie nun mit einem Aufruf von `./neo4j start` im `bin` Ordner starten.
Das Neo4j Webinterface visualisiert die Ergebnisse ihrer Anfragen. Um das Webinterface lokal zu benutzen, während der Neo4j Prozess auf einem der Cluster-Knoten läuft, müssen Sie den Port mit SSH weiterleiten (In diesem Beispiel Wurde der Port 10090 gewählt): `ssh -L 10090:abu1:10090 cluster`

1.2 Datenimport

In dieser Aufgabe wollen wir Neo4j nutzen um einen Graphen der Links zwischen einzelnen Wikipedia Artikeln aufzubauen. Dabei soll jeder Knoten einen Artikel und jede Kante zwischen zwei Knoten die entsprechende Verlinkung zwischen den Artikeln darstellen. Dafür verwenden wir Datensätze der Wikipedia von `https://dumps.wikimedia.org`, die bereits für Sie in CSV-Dateien konvertiert wurden.
Da das Importieren von Daten sehr lange dauern kann verwenden für den Datenimport die kleinere Datenbank des englischen wikiqoute: `https://dumps.wikimedia.org`.
Die CSV-Dateien für den Import finden Sie in `/home/bigdata/neo4j` auf dem Cluster.
Verwenden Sie den Batch Importer um die Artikel von Wiki-quote zu importieren und für jeden Artikel eine Node zu erstellen. Daraufhin verwenden Sie Cypher um Verlinkungsrelationen in die Datenbank zu importieren. Sie werden feststellen das ein komplett naiver Ansatz, ohne Indizes, sehr langsam ist.
Tipp: Um die Geschwindigkeit zu verbessern lassen sich sich exemplarisch für einen der Links mittels `EXPLAIN` anzeigen wie die Query funktioniert und überlegen Sie was sich verbessern lässt.
Informationen zum Batch-Import gibt es hier: `http://neo4j.com/docs/stable/import-tool.html`
Zusammengefasst führen Sie also die folgenden Schritte durch:

- Über Optimierungen Nachdenken
- Datenbank Initial einrichten
- wikiquote-all.csv per Batch-Import importieren
- wikiquote-links.csv mit Cypher importieren

Abgabe:

`1-import.txt` Eine Beschreibung ihres Vorgehens beim Importieren.

2 Cypher Query Language (60 P)

Nutzen sie Cypher für die folgenden Teilaufgaben:

- Geben sie die Namen aller Artikel aus, die auf den Artikel *Artificial_intelligence* verlinken.
- Finden sie alle kürzesten Pfade, die beim Navigieren der Wikipedia über Pagelinks vom Artikel *Alan_Turing* zum Artikel *Gene_Amdahl* führen.
- Wieviele Artikel enthält der Datensatz?
- Welcher Artikel ist am stärksten vernetzt, d.h. hat die meisten eingehenden und ausgehenden Links?
- Erstellen Sie einen Artikel für sich selbst.
- Zusätzlich zu Artikeln soll es nun auch Zitate geben, die jeweils einem Artikel zugeordnet sind und als eigene Objekte in der Datenbank erfasst werden. Weisen Sie einem beliebigen Artikel ein Zitat zu.

Abgabe:

`2-queries.txt` Ihre Cyper Queries.

3 Klassifikation von titanic.csv (150 P)

Der Datensatz `/home/bigdata/titanic.csv` enthält Informationen über die Passagiere der Titanic. Zu jedem Passagier gibt es die folgenden Informationen: Buchungsklasse, Geschlecht, Alter. Zusätzlich ist vermerkt, ob der Passagier das Unglück überlebte oder nicht.
Gehen Sie wie folgt vor:

- Erstellen Sie zunächst einen Entscheidungsbaum für alle Passagiere und versuchen Sie hierbei sinnvolle Entscheidungsregeln für das Überleben der Passagiere herzuleiten.
- Zur Evaluation der Güte des Baumes teilen Sie die Beobachtungsdaten in zwei Klassen und implementieren Sie eine k-cross-validation[1]. Berechnen Sie die Fehlerrate der Vorhersage für $k = (2, .., 10)$ und stellen Sie die falsch positive und falsch negative Fehlerrate in einem Diagramm dar.
- Erweitern Sie die Validation. Stellen Sie sich vor Sie hätten nur $m = (1/2, 1/4, 1/8, ...)$ Daten zur Verfügung. Erstellen Sie wieder einen Diagramm für die Fehlerrate.

Es ist Ihnen frei gestellt, ob Sie R oder Python für die Analyse nutzen.

Abgabe:

`3-titanic-tree.[R|py]` Ihre Analyseskript der Titanic Daten mit Hilfe von R oder Python.
`3-titanic-tree.pdf` Ihre Analyse des Entscheidungsbaums und der k-cross-validation.

[1] Implementieren Sie das Verfahren zur Aufteilung in Training- und Validierungsset selbst!

4 Clustering der Wikipedia-Artikel mit R (120 P)

In dieser Aufgabe bilden wir Cluster von Wikipedia-Artikel basierend auf den relativen Worthäufigkeiten im Datensatz `/home/bigdata/wikipedia-text-tiny-clean.csv`.
Verwenden Sie den k-means Algorithmus und testen Sie diesen mit unterschiedlich vielen Clustern. Inspizieren Sie die Ergebnisse. Sind die ermittelten Cluster sinnvoll?
Verwenden Sie nun das modellbasierte Clustering und inspizieren Sie die Ergebnisse. Wie viele Klassen wurden erstellt? Diskutieren Sie kurz die Laufzeit Ihres Programms.

4.1 Hinweise

Zu Testzwecken ist es sinnvoll zunächst nur mit einer Teilmenge der Dokumente zu arbeiten und Ihr Skript zu erstellen.

4.1.1 Code Gerüst für R

```r
library(tm)
d = read.csv("/home/bigdata/wikipedia-text-tiny-clean.csv", sep=";", quote="", header=F, stringsAsFactors=FALSE)
colnames(d) = c("title","content")

corpus = Corpus(VectorSource(d$content))

# We can set metadata, e.g. title:
i=0
corpus = tm_map(corpus, function(x) {
   i = i + 1
   meta(x, "title") = d$title[i]
   x
})

# you can do:
# inspect(corpus)

# To access an element in the corpus use:
# corpus[[X]]su

# we apply functions to clean the data
corpus = tm_map(corpus, removePunctuation)
corpus = tm_map(corpus, stripWhitespace)
corpus = tm_map(corpus, removeNumbers)
corpus = tm_map(corpus, content_transformer(tolower))
corpus = tm_map(corpus, removeWords, stopwords("english"))
# Create a sparse matrix for each document the word frequency
dtm = DocumentTermMatrix(corpus)
# To see data: e.g. call inspect(dtm[1:5,1:50] )

# Now apply clustering algorithms

# Inspect results....
```

Abgabe:

`4-wikipedia-clusters.R`	Ihre Analyseskript.
`4-wikipedia-clusters.pdf`	Ihre Analyse der Cluster.

5 Bonusaufgabe Klassifikation von neuen Wikipedia Artikeln (180 P)

Stellen Sie sich vor, Sie müssten für einen Wikipedia-Artikel, der bislang über keine Kategorie verfügt, automatisch eine Kategorie vorschlagen.
Mit Hilfe von überwachtem Lernen könnten wir dies wie folgt erreichen:

- Laden Sie die CSV-Datei, die Sie in einer vorangegangenen Aufgabe erstellt haben, welche die Hauptkategorie für jeden Artikel enthält und den dazugehörigen Text.
- Teilen Sie die Menge an Artikeln in Testset und Validierungsset auf.
- Trainieren Sie einen k-nearest neighbor Klassifikator mit Ihren Testdaten und der Kategorie. Als Features eines Artikeltextes wählen wir die relativen Worthäufigkeiten, d.h. zwei Artikel sollten ähnlich sein, wenn die individuellen Worthäufigkeiten ähnlich verteilt sind.
- Validieren Sie die Güte mit Ihrem Validierungssets.
- Inspizieren Sie einige der Artikel im Testset und vergleichen Sie, ob die ermittelte Kategorie mit den von Benutzern erstellten Kategorien übereinstimmt.

Es steht Ihnen frei R oder Python zu verwenden.

Abgabe:

`5-wikipedia-knn.[R|py]`	Ihre Analyseskript in R oder Python.
`5-wikipedia-knn.pdf`	Ihre Analyse der Güte des Algorithmus und einige Beispiele.