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

Übungsblatt 6 zur Vorlesung
Big Data Analytics
im WiSe 2015/2016
Abgabe: 27.11.2015, 540 P

Bitte dokumentieren Sie die benötigte Bearbeitungszeit für die einzelnen Aufgaben in `bearbeitungszeit.txt`. Bitte denken Sie auch daran uns Feedback zur Veranstaltung zu geben:

`http://goo.gl/forms/B01IIDHvW2`

Bei der Abgabe von Source Code kommentieren Sie diesen bitte.

1 Analyse des IMDb Datensatzes mit R (210 P)

In dieser Aufgabe werden wir Ihre geparsten IMDb Daten (`imdb[movie|actors].csv`) mit R analysieren. Hierbei gehen wir wie folgt vor:

- Explorieren Sie Ihre Daten. Drucken Sie hierfür statistische Informationen aus, erstellen Sie Plots und beschreiben Sie Ihre Erkenntnisse.
- Analysieren Sie einen möglichen Zusammenhang der Bewertung zu der Anzahl der Bewertungen, Produktionsland, Anzahl Schauspielern, Keywords, Genre oder Zitaten. Verwenden Sie hierfür den Korrelations-Koeffizienten.
- Versuchen Sie ein lineares Model für die Vorhersage der Bewertung aufzustellen. Verwenden Sie hierfür beliebige Parameter d.h. unabhängige Variablen die Ihnen so einfallen.

1.1 Hinweise

Verwenden Sie die Bibliothek *rjson* um das CSV zu verarbeiten.

1.1.1 Code Gerüst für R

```r
library(rjson)

readIMDB = function(){
    # read the CSV file, defines the classes for the columns
    i = read.csv("imdbmovie.csv", header=F, colClasses=c("character", "integer", "character", "character"))
    colnames(i) = c("title","year","country","actors")

    # convert the JSON strings in actors to a list
    i$actors = lapply(i$actors,  fromJSON)

    return (i)
}
```

Abgabe:

`1-imdb-analyse.R`	Ihr kommentiertes R-Programm, welches alle Analyseschritte noch einmal durchführt.
`1-imdb-analyse.pdf`	Die Ausgaben von R und dabei erstellte Grafiken. Fügen Sie eine kurze Beschreibung Ihrer Ergebnisse ein.

2 Identifikation von Film-Zitaten für die Wikipedia (180 P)

In dieser Aufgabe wollen wir ein Python-Programm schreiben, welches zu einen Wikipedia-Artikel „passenden“ Zitate aus Filmen von der IMDb ausgibt. Verwenden Sie hierfür Ihre Daten von der `imdb-[movie|actors].csv` und `wiki-clean-frequency.csv` und speichern Sie das Ergebnis unter `wiki-movie-quotes.csv` mit der ArtikelID und der Liste der dazu passenden Zitate.
Das Erstellen eines guten Modells für passende Zitate ist der schwierigste Teil dieser Aufgabe. Sie können verschiedene Modelle entwerfen, ein Beispiel sei hier über die Worthäufigkeiten gegeben:

- Bezeichnen wir nun die relative Wortfrequenz eines Wortes W in einem Text T als $f(W,T)$.
- Sei A der Text des Wikipedia-Artikels mit dem Dictionary der relativen Worthäufigkeiten ($W1 = f(W1,A)$,$W2 = f(W2,A)$, ...). Hierbei sollen Stoppworte entfernt werden und die Wörter auf Ihren Wortstamm reduziert werden (verwenden Sie hierfür ihre `stoppwortlist.csv` oder laden Sie eine einfache Stoppwortliste aus dem Internet). f_i enthält die relative Frequenz, also bspw. 1% der Worte des Artikels sind das Wort mit dem Wortstamm „Computer“.
- Sei Q eine Liste mit Zitaten ($q_1, ..., q_n$) und deren Worthäufigkeiten in der Liste aus Dictionaries QF=(q1=($wq1 = f(wq1, q1)$,...), q2=...). Diese sind ebenfalls um Stoppworte bereinigt und auf ihren Wortstamm reduziert.
- Wählen das Zitat q_s für den Artikel A aus, welches den Abstand der relative Worthäufigkeiten zwischen A und q_s minimiert, d.h. verwenden Sie die Funktion $m(A,Q) = min_{q_s \in QF}(\sum_{w \in A \cup q_s}(|f(w,A) - f(w,q_s)|))$ um die Übereinstimmung (Score) zu berechnen. Der maximale Score beträgt hierbei 2 (kleiner ist besser).
- Transformieren Sie nun die Worthäufigkeiten der Zitate zuerst in einen Vektor, der alle Wörter des Artikels enthält (und andere Worte des Zitats ignoriert), dann können wir ebenfalls leicht andere Distanzfunktionen nutzen oder auch Clustering Algorithmen. Die Score unterscheidet sich hierbei jedoch, sofern im Zitat andere auch Wörter auftreten, welche nicht im Artikel vorhanden sind. Dies lässt sich jedoch leicht beheben.
- Lassen Sie sich bei der Implementation von m die ersten 5 Zitate inklusive Score zurück geben.
- Eine Annahme könnte sein, dass längere Wörter wichtiger sein sollten. Gewichten Sie also die Abstandsfunktion entsprechend der Wortlänge.

Es ist nicht nötig, dass Sie für alle Artikel der Wikipedia Zitate erzeugen. Wählen Sie einige Artikel aus und dokumentieren Sie diese zusammen mit dem Programmaufruf als Kommentar im Programm.

Abgabe:

`2-imdb-quotes.py`	Ihr kommentiertes Python-Programm.
`2-imdb-quotes-beispiele.txt`	Einige Artikeltitel mit den dazu ermittelten Zitaten.

3 TEZ DAGs: Verarbeitung der Wikipedia (150 P)

In dieser Aufgabe schreiben Sie noch einmal ein Programm zur Analyse der Worthäufigkeiten von `wiki-clean.csv`. Diesmal verwenden wir jedoch TEZ. Hiermit sollte es möglich sein zusätzlich in einem einzelnen Programmablauf die Häufigkeiten aller Worte über alle Artikel zu aggregieren und in einer weiteren Datei abzulegen.
Ihre Ausgabedatei sollte wie folgt aufgebaut sein:

```
ArtikelID1,"Name1",Wortanzahl,"{\"Wort\":1,\"Wort2\":4,\"Wort3\":3, ...}"
ArtikelID2,"Name2",Wortanzahl,"{\"Wort\":1,\"Wort2\":4,\"Wort4\":4, ...}"
```

Bzw. für die Aggregation:

```
Wort1,2
Wort2,4
...
```

Zur Durchführung der Analyse auf mehrere Dateien wollen wir ebenfalls TEZ-Sessions verwenden. Schauen Sie sich hierfür die Einführung hier an: `http://hortonworks.com/blog/introducing-tez-sessions/` Testen Sie den Leistungsgewinn, indem Sie das Programm mehrfach auf den Eingabedatensatz starten.

3.1 Hinweise

Aufgrund eines Problems mit dem Klassenpfad müssen Sie Ihr JAR vor der Ausführung an folgende Stelle kopieren: `/home/kunkel/bigdata/offentlich/jars/<GRUPPEN_NR>.jar`. Die Gruppen Nummer wurde Ihnen bereits zuvor mitgeteilt.
Weitere Beispiele finden Sie unter `https://github.com/apache/tez/tree/master/tez-examples/src/main/java/org/apache/tez/examples`.

3.1.1 Code-Gerüst für Java

Aufgrund des Umfangs des Code-Gerüsts für Java finden Sie dieses im bereitgestellten Archiv unter `6-TezWikipedia.java`, welches großteils ein sinnvoll gewähltes Beispiel aus den Tez-Beispielen entspricht. Das dort gegebene Code-Gerüst zählt die Worthäufigkeiten in einer Eingabedatei und ist ein leicht modifiziertes TEZ-Beispiel (`Wordcount.java`). Ebenfalls finde Sie die Skripte zum kompilieren und Ausführen unter `6-compile-run-tez.sh`. Passen Sie dort jedoch Ihre Gruppennummer an!

Abgabe:

`3-tez.java` Ihr kommentiertes Java-Programm für die Analyse der Worthäufigkeiten in den Artikeln.