Otto-Friedrich-Universität Bamberg
Lehrstuhl für Medieninformatik
Prof. Dr. Andreas Henrich
Dipl.-Wirtsch. Inf. Daniel Blank

Information Retrieval 1

Wintersemester 2014/15

- Teilleistung 2 -

- Beachten Sie zu den Teilleistungen stets die Hinweise im Onlinekurs. Verfolgen Sie hierzu auch die Diskussionen in den Foren des Kurses.
- Geben Sie Ihre Lösung als **ein ZIP-Archiv** ab.
- Die Abgabe erfolgt durch den Upload des ZIP-Archivs über das entsprechende Formular zu dieser Teilleistung im Onlinekurs.
- Benennen Sie das Dokument mit den Lösungen nach dem Schema:

***Nachname-Vorname*-TL2.zip**

- Alle Aufgaben ohne Implementierungsaspekte bzw. praktische Anwendung lösen Sie bitte in **EINER zusammenhängenden PDF-Datei**, die Sie mit ins ZIP-Archiv packen.
- Verwenden Sie bei der Programmierung keine weiteren als die angegebenen Bibliotheken.
- Bei der Programmierung bietet es sich an diversen Stellen an mit Exceptions zu arbeiten. „Fehlerfälle“ können Sie sehr einfach behandeln, indem Sie stets eine *RuntimeException* werfen.
- Abgabetermin ist der

Freitag – 19.12.2014, 23:55 Uhr (Achtung: es zählt die Serverzeit).

Wir wünschen Ihnen viel Spaß und Erfolg bei der Bearbeitung der Aufgaben!

Hinweis 1: *Da der Onlinekurs mit halben Punkten nicht umgehen kann, wird es pro Teilleistung 40 Moodle-Punkte geben, die den eigentlichen 4 Punkten pro Teilleistung entsprechen. Am Ende des Kurses werden die Moodle-Punkte Ihrer Teilleistungen addiert, durch 10 dividiert und anschließend auf halbe Punkte gerundet. Sollten Sie z.B. 117 Moodle-Punkte erreicht haben, entspricht dies 11,5 Bonuspunkten. 108 Moodle-Punkte würden bspw. 11 Bonuspunkten entsprechen. Weitere Informationen finden Sie hier: http://www.uni-bamberg.de/index.php?id=10596*

Hinweis 2: *Sofern Sie bei der Bearbeitung der Aufgaben auf Schwierigkeiten stoßen bzw. Aufgabenstellungen aus Ihrer Sicht unklar formuliert sind, möchten wir Sie bitten, intensiv und umgehend die Kommunikationsmöglichkeiten über das Forum oder per E-Mail zu nutzen und ggf. nachzufragen. Ein direktes Feedback soll Ihnen und auch Ihren Kommilitoninnen und Kommilitonen den Zugang zu den Aufgaben erleichtern, der aufgrund unterschiedlicher Vorkenntnisse ggf. ausschließlich auf Basis der Aufgabenstellung nicht immer für Alle mit dem gleichen Aufwand möglich sein kann.*

Aufgabe 1 (IR-Modelle, 4 Punkte):

Gegeben sei die Anfrage „Fußball Weltmeister Deutschland“. Für die drei Begriffe gelten folgende Häufigkeiten:

	in D_1	in D_2	in der Kollektion insgesamt
Fußball	4	1	1000
Weltmeister	2	3	30.000
Deutschland	0	4	100.000

Dokument D_1 besteht aus 80 Wörtern und D_2 aus 40 Wörtern. Die Kollektion insgesamt umfasst 750.000 Wörter.

Bestimmen Sie ein Ergebnisranking der Dokumente im Hinblick auf die Anfrage und rechnen Sie die entsprechenden Scores aus. Wenden Sie dazu Sprachmodelle in drei Varianten an:

- einfache Maximum-Likelihood-Schätzung
- mit Jelinek-Mercer Glättung
- mit Dirichlet-Glättung

Setzen Sie λ zu 0,3 und μ zu 150 und verwenden Sie den natürlichen Logarithmus.

Aufgabe 2 (IR- Modelle, 2 Punkte):

a) Berechnen Sie den Score von Dokument D für die Anfrage $Q =$ „Universität Bamberg“ (qf_i hat jeweils den Wert 1) nach BM25 für folgende Situation:
   - es existieren keine Relevanzinformationen
   - 250.000 Dokumente existieren insgesamt
   - „Universität“ tritt insgesamt in 4.000 Dokumenten auf
   - „Bamberg“ tritt insgesamt in 2.000 Dokumenten auf
   - „Universität“ kommt in D 10-mal vor
   - „Bamberg“ kommt in D 20-mal vor
   - D enthält 900 Wörter während die Dokumente im Durchschnitt 1.200 Wörter enthalten.
   - es gelte: $k_1 = 1{,}2$; $b = 0{,}75$ und $k_2 = 125$

[Verwenden Sie bei der Rechnung bitte auch hier den natürlichen Logarithmus.]

b) Was wirkt sich stärker auf den Score des Dokumentes in Teilaufgabe a) aus: Wenn ein Vorkommen von Universität zusätzlich in D eingefügt würde, oder wenn ein Vorkommen von Bamberg zusätzlich in D eingefügt würde? Begründen Sie Ihre Antwort anhand der Formel. Sie müssen die sich ergebenden neuen Werte nicht ausrechnen!

Aufgabe 3 (Reading Exercise zu IR-Modellen, 14 Punkte):

Lesen Sie das Papier „*Graph-of-word and TW-IDF: New Approach to Ad Hoc IR*“, das ein neues IR-Modell vorstellt, das Abhängigkeiten zwischen Termen berücksichtigt. Beantworten Sie anschließend die folgenden Fragen.

Gegeben sind zwei kurze Dokumente und eine Anfrage, d.h. jeweils der Text zwischen den Klammern, der anhand von Leerzeichen aufgesplittet wird:

D_1: (peyton and eli manning are quarterback brothers)

D_2: (the two manning brothers are the only quarterback brothers who won the super bowl)

Q: (manning brothers)

Die Kollektion besteht aus 5000 Dokumenten, *manning* kommt in 80 Dokumenten vor, *brothers* in 150. Verwenden Sie für die Ähnlichkeitsberechnungen bei dieser Aufgabe jeweils die Formel $Score(Q, D_k) = \sum_i q_{ik} * d_{ik}$, wobei q_{ik} der absoluten Vorkommenshäufigkeit eines Terms i in der Anfrage entspricht.

a) Ermitteln Sie durch Anwendung des TF-IDF-Modells, ob D_1 oder D_2 das ähnlichere Dokument zur Anfrage Q ist. Verwenden Sie bei Ihrer Berechnung $d_{ik} = (\log(f_{ik}) + 1) \cdot \log\frac{N}{n_k}$ und somit nur den Zähler der Ihnen bekannten TF-IDF-Formel. Arbeiten Sie mit dem natürlichen Logarithmus.

b) Das zu lesende Papier skizziert das sog. TW-IDF-Modell. Beschreiben Sie kurz die wesentlichen Ideen dieses Modells und grenzen Sie es vom TF-IDF-Modell ab. Erstellen Sie hierzu auch einen Graphen zu Dokument D_1, der für die Ermittlung der Termgewichte benötigt wird. Verwenden Sie eine Fenstergröße von 4.

c) Modifizieren Sie Ihre Berechnungen aus Aufgabenteil a), indem Sie nun das TW-IDF-Modell bei einer Fenstergröße von 4 anwenden. Welches Dokument ist das ähnlichere zur Anfrage? Verwenden Sie die Formel zur Berechnung der d_{ik} Werte aus Aufgabenteil a) und ersetzen Sie den Teil $(\log(f_{ik}) + 1)$ durch den TW-Teil gemäß TWIDF-Modell. Machen Sie deutlich, wie die neuen TW-Gewichte entstehen.

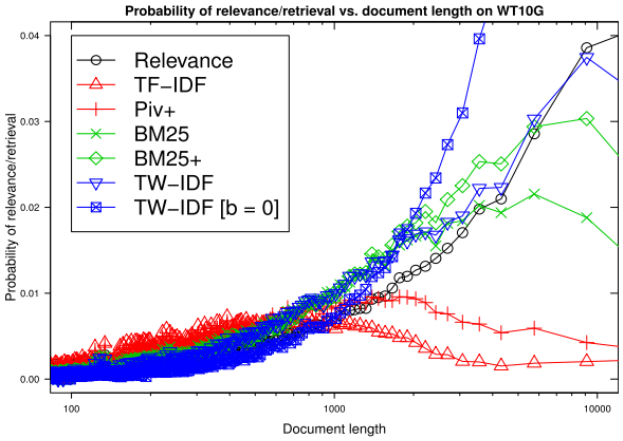
d) Erläutern Sie knapp, ob es bei der Modellierung günstig ist die folgenden Konzepte zu berücksichtigen und begründen Sie anhand der Aussagen im Papier:

   i. einen gerichteten vs. einen ungerichteten Graph

   ii. gewichtete vs. ungewichtete Kanten

   iii. fixe vs. variable Fenstergröße

e) Erläutern Sie Abbildung 3 des Papiers und dabei die wesentlichen Aussagen, die mit der Abbildung transportiert werden sollen. Auf die beiden Systeme Piv+ sowie BM25+ müssen Sie dabei nicht eingehen.



Quelle: F. Rousseau and M. Vazirgiannis. 2013. Graph-of-word and TW-IDF: new approach to ad hoc IR. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, New York, NY, USA, 59-68.

Aufgabe 4 (Ein Suchsystem auf Basis von Apache Lucene, 20 Punkte):

Apache Lucene (http://lucene.apache.org/core/index.html) ist eine mächtige IR-Bibliothek. Sie bildet u.a. die Grundlage für die Wikipedia-Suche. Außerdem basiert die Suche auf vielen Unternehmenswebseiten auf Apache Solr, das wiederum auf Lucene beruht.

Unter http://www.manning.com/hatcher3/Sample-1.pdf finden Sie eine kurze Einführung zu Lucene. Lesen Sie das PDF. Im PDF sind bereits Code-Fragmente enthalten, die eine einfache Suche auf Basis von TXT-Dateien beschreiben. Die beiden Klassen (*Indexer.java* sowie *Searcher.java*) finden Sie an die aktuelle Lucene-Version 4.9 angepasst im bereitgestellten Eclipse-Projekt im Paket *sheet2*. Beide Klassen sind direkt lauffähig. Sie können dies direkt ausprobieren, indem Sie beliebige selbsterstellte TXT-Dateien indexieren und nach den Inhalten suchen. Erzeugen Sie sich hierzu ein paar TXT-Dateien.

Teil 4.1 – Indexierung der CACM Kollektion

Im Unterordner *cacm* in Ihrem Eclipse-Projekt finden Sie eine Testkollektion, die wir nun indexieren wollen.

Wir werden verschiedene IR-Modelle miteinander vergleichen, die Lucene anbietet und die wir im Kurs kennengelernt haben. Als Testkollektion verwenden wir die CACM Kollektion (http://ir.dcs.gla.ac.uk/resources/test_collections/cacm/). Machen Sie sich zunächst mit der Kollektion vertraut, indem Sie die readme-Datei lesen.

Indexieren Sie die Kollektion, indem Sie die Methode *indexFile(File file)* der Klasse *CacmIndexer.java* implementieren. Dabei wollen wir folgende Informationen verwenden, die jeweils in einem eigenen Feld indexiert werden sollen:

- *.T* der Titel des Textes
- *.W* der Textinhalt bzw. Inhalt der Abstracts

Diese Feldinhalte sollen später durchsuchbar sein. Zudem soll jedes Dokument durch eine DokumentID repräsentiert werden, die dem *sequential identifier* (*.I*) der Kollektion entspricht. Die Feldnamen entnehmen Sie bitte aus den als *final* deklarierten Variablen der Klasse.

Teil 4.2 – Erstellung der QREL-Datei

Im 2-seitigen Papier *thetrecfiles.pdf* wird ein Evaluationstool vorgestellt, das verschiedene Evaluationskennzahlen wie z.B. MAP und P@20 berechnen kann. Lesen Sie zunächst das Papier, das Sie auch unter http://dl.acm.org/citation.cfm?id=2009916.2010164 finden, sofern Sie das Tool unter http://thetrecfiles.nonrelevant.net/ noch nicht kennen.

Erstellen Sie nun für die Kollektion eine *qrel*-Datei *cacm.qrel*. Informationen zum Format finden Sie hier: http://trec.nist.gov/data/qrels_eng/. Verwenden Sie als Grundlage die Datei *qrels.text* der CACM Kollektion. Implementieren Sie die Methode *StringBuilder generateQRels(String filename)* der Klasse *CacmHelper.java*, die ein Objekt vom Typ *StringBuilder* zurückgeben soll, auf Basis dessen innerhalb der Main-Methode die Ausgabedatei mit der Endung *qrel* erstellt werden kann.

Teil 4.3 – Anfragebearbeitung mit verschiedenen IR-Modellen

Implementieren Sie nun die Methode *search(String indexDir, Similarity sim, Analyzer analyzer)* der Klasse *CacmSearcher*. Beachten Sie dabei die folgenden Aspekte:

- Es sollen alle Anfragen aus der Datei *query.text*, die mit der CACM Kollektion verteilt wird, durchgeführt werden.
- Die Suche soll auf allen indexierten Feldern stattfinden.
- Eine Gewichtung der Felder soll nicht vorgenommen werden.
- Aus Vereinfachungsgründen soll unsere Suche nur maximal die ersten 1000 Ergebnistreffer analysieren.
- Ergebnisse sollen in eine TREC-Datei geschrieben werden, sodass diese von dem in Aufgabe 4.2 genannten Evaluationstool ausgewertet werden können.
- Hinweise zum Aufbau einer TREC-Datei finden Sie unter http://ir.iit.edu/~dagr/cs529/files/project_files/trec_eval_desc.htm
- Alle Informationen, die für eine *TREC*-Datei wichtig sind, sollen zunächst in der Variable *builder* erfasst werden. Daraus wird abschließend eine Datei erstellt, die mit Hilfe des Evaluationstools später evaluiert werden kann.

Teil 4.4 - Vergleich verschiedener IR-Modelle

Ermitteln Sie für die folgenden IR-Modelle unter Anwendung der beiden genannten *Analyzer* jeweils die MAP, P@10 und P@20-Werte und stellen Sie diese geeignet in Form einer Tabelle dar. Welche(s) Verfahren schneidet am besten ab? Überrascht Sie dies?

Similarity:

- *LMJelinekMercerSimilarity(0.2f)*
- *LMDirichletSimilarity()*
- *BM25Similarity()*
- *DefaultSimilarity()*

Analyzer:

- *StandardAnalyzer* (enthält Stoppworteliminierung aber kein Stemming)
- *MyStemAnalyzer* (führt zusätzlich ein Stemming durch)