UNIVERSITÄT LEIPZIG

Institut für Informatik
Lehrstuhl für Rechnernetze und Verteilte Systeme

Praktikum Computational Advertising
Prof. Dr.-Ing. Christoph Lindemann
Sommersemester 2011

Betreuer: **Dipl.-Inform. Jan Friedrich, friedrich@rvs.informatik.uni-leipzig.de**
Dipl.-Inform. Michael Petrifke, petrifke@rvs.informatik.uni-leipzig.de

Aufgabe 3: SimRank++

Übersicht

Advertiser sind nicht in der Lage Bid Phrases für alle möglichen Suchanfragen anzugeben. Die Vielfältigkeit der Sprachen und die semantische Ähnlichkeit macht dies unmöglich. Aus diesem Grund ist man dazu übergegangen aus manuell vorgegebenen Bid Phrases automatisch neue Kandidaten zu generieren. Dies kann sowohl online geschehen, wenn die Suchanfrage eines Nutzes ad hoc in eine Menge von Suchanfragen umgeschrieben wird, als auch offline, wenn nach der Eingabe der Bid Phrases durch den Werbetreibenden diese durch passende Vorschläge erweitert werden. Im Folgenden betrachten wir den online Fall.

Zum Einsatz kommt der SimRank++ Algorithmus [1]. Dieser baut auf dem bekannten SimRank auf und berechnet die Ähnlichkeit zweier Suchanfragen. Hierzu werden Klick Daten verwendet. Es wird ein bipartiter Graph aufgebaut der Suchanfragen mit Ads verbindet. Die berechneten Ähnlichkeiten werden abschließend genutzt um ein Query Rewriting durchführen zu können.

Die Bearbeitungsfrist beträgt „Christi Himmelfahrt“ geschuldet für dieses Arbeitsblatt drei Wochen.

Aufgabe

Teil a: Simrank

Auf der Praktikumswebsite finden Sie einen Klick-Datensatz. Implementieren Sie das Frontend entsprechend dem Interface aus Aufgabe 2. Jede Anfrage wird einem Query Rewriting unterzogen. Die dafür notwendige Analyse des Klick-Graphen und die genaue Implementierung des Rewriting sollen ähnlich der Aufgabe 2 zur Laufzeit bestimmt werden. Die dort verwendeten Plugin-Konzepte sind nun auch hier wieder anzuwenden.

In dieser ersten Teilaufgabe basiert die Implementierung schlicht auf Simrank. Bestimmen Sie

für jede Suchanfrage des Datensatzes die fünf ähnlichsten Queries und speichern Sie diese in einer Datenbank. Dies erfolgt in einem Vorverarbeitungschritt der in `analyzeClickGraph` einzubetten ist. Die Methode `analyzeDemographicFeatures` hat noch keine Bedeutung und kann einfach als leere Funktion definiert werden.

Die Simrank Implementierung wird detailliert im entsprechenden Paper [1] erläutert. Die für Sie wichtigsten Formeln seien hier noch einmal kurz erwähnt:

$$s(q,q') = \frac{C_1}{N(q)N(q')} \sum_{i \in E(q)} \sum_{j \in E(q')} s(i,j) \qquad s(\alpha,\alpha') = \frac{C_2}{N(\alpha)N(\alpha')} \sum_{i \in E(\alpha)} \sum_{j \in E(\alpha')} s(i,j)$$

Dabei ist $C_1 = C_2 = 0.8$ zu wählen und $k = 5$. Allerdings werden Sie zur Lösung der Aufgabe die schnellere und leichtere Matrizen-Version des Algorithmus verwenden:

Algorithm 1 Simrank Computation

Require: transition matrix P, decay factor C, number of iterations k
Ensure: similarity matrix S

1: $[N,N] = \text{size}(P)$;
2: $S = I_N$;
3: **for** $i = 1 : k$, **do**
4: $\quad \text{temp} = C * P^T * S * P$;
5: $\quad S = \text{temp} + I_N - \text{Diag}(\text{diag}(\text{temp}))$;
6: **end for**

Nutzen Sie soweit möglich die boost uBLAS Bibliothek [2].

Bauen Sie nun eine Konsolen-Applikation, in der ihr Frontend mit ihrem Backend verbunden ist. Die Applikation soll neben den Parametern aus Aufgabe 2 nun noch folgende Kommandos akzeptieren:

```
query <query>

    Suche des passenden Ads zur Suchanfrage <query term> mithilfe
    des Frontends; Ausgabeformat analog zu matchad


load_click_data <click file>

    Start der offline Auswertung der Klick-Daten und Ausgabe der
    fünf ähnlichsten Queries zu jeder Query aus dem Datensatz (mit
    dem genauem Ähnlichkeitswert)
```

Teil b: Simrank++

Als Verbesserung zur ersten Version ist nun Simrank++ umzusetzen. Dazu werden nun Kantengewichte berechnet (Häufigkeit des Auftretens einer Query – Ad-Verbindung in den Klick-Daten).

Algorithm 2 Simrank++ Computation

Require: weighted transition matrix P', evidence matrix V, decay factor C, number of iterations k
Ensure: similarity matrix S'

```
[N,N] = size(P');
S' = I_N;
for i = 1 : k, do
    temp = C * P'^T * S' * P';
    S' = temp + I_N − Diag(diag(temp));
end for
S' = V. * S';
```

Die Berechnung der „evidence“ und der gewichteten Übergangswahrscheinlichkeiten W, die sich aus dem „spread“ und „normalized_weight“ zusammensetzen, sind ausführlich im Paper beschrieben.

Abgabe

Bitte schicken Sie pro Gruppe genau eine Lösung bis zum **17.06.11, 23:59 Uhr** per Email an Jan Friedrich (*friedrich@rvs.informatik.uni-leipzig.de*). Die Abgabe sollte den gesamten Quellcode enthalten sowie ihre Build-Skripte (z.B.. autotools, cmake, qmake, Custom Makefile, vcproj, ...). Geben Sie auch an in welcher Umgebung die Aufgabe angefertigt wurde (Betriebssystem, Compiler, inkl. Version). Wir empfehlen die Aufgabe mit GCC unter einem Unix-kompatiblem Betriebssystem oder der CygWin-Umgebung [3] zu bearbeiten. Sollten Sie dennoch unter Windows arbeiten wollen, können wir bei Problemen eventuell nur begrenzt Hilfe anbieten. Zudem müssen Sie zum Reviewgespräch einen Laptop mitbringen, auf dem der Code sowohl kompiliert als auch ausgeführt werden kann.

Die Termine für die Reviewgespräche sind am **20.06.**, **21.06.** und **22.06.** jeweils von **15 bis 17 Uhr**. Kommen Sie bitte innerhalb dieser Zeiten, wenn möglich mit eigenem Laptop, unangemeldet mit beiden Gruppenmitgliedern zu Jan Friedrich (Raum 04-06). Die Reviews werden ca. 15 Minuten dauern.

Referenzen

[1] Simrank++: Query Rewriting through Link Analysis of the Click Graph, I. Antonellis, H. Garcia-Molina, C. Chang, Proc. VLDB Endowment, 2008

[2] http://www.boost.org/doc/libs/release/libs/numeric/ublas/doc/index.htm

Informationen zur Abgabe

- **Siehe Webseite: http://rvs.informatik.uni-leipzig.de/de/lehre/SS11/praktika/ca/**

Viel Erfolg!