

Web-Informationssysteme, WS 2009/10

Seminar-Übung 3: PageRank, HITS, Tabellarische Daten, Inverted Files

Besprechung am Mi 11.11.2009

— SÜ-3.1 —

Was Ihr wissen müßt über “PageRank & HITS”

Wiederholung

PAGERANK

- Was ist und wie wird die **Google-Matrix** (oder PageRank-Übergangsmatrix) berechnet?

Die Google-Matrix (PageRank-Übergangsmatrix) ergibt sich wie folgt:

1. Der Graph wird durch seine transponierten Adjazenzmatrix A^T repräsentiert. $A_{ij}^T = m$ falls es m Links von j nach i gibt.
2. Falls es Knoten ohne ausgehende Links (*dangling nodes*) gibt, wird die zugehörige Zeile durch $(1, \dots, 1)^T$ ersetzt, wobei n die Anzahl der Knoten im Graphen ist. Bei solchen Knoten wird also eine Kante zu jedem Knoten eingefügt.
3. Im nächsten Schritt wird diese Matrix normalisiert zu einer zeilen-stochastischen Matrix H : Sei $\deg(i)$ die Gesamtanzahl aller von i ausgehenden Links nach Schritt (2) (*degree*, Summe der Einträge in der i -ten Zeile von A^T) und D der (Spalten-) Vektor aus diesen Werten, also $D_i = \deg(i)$. Dann ist $H = A^T \cdot D$.

Damit ist $H_{ij} = m/\deg(j)$ der Anteil der Links von j nach i unter allen Links ausgehend von j .

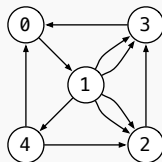
4. Schließlich soll die Matrix noch so verändert werden, dass es eine (typischerweise kleine) Wahrscheinlichkeit zufällig von jedem Knoten zu jedem anderen zu springen (*random leap probability*, α). Die resultierende Matrix ist die Google-Matrix G

$$G = \alpha H + (1 - \alpha) \begin{pmatrix} 1/n \\ \vdots \\ 1/n \end{pmatrix} \cdot (1 \dots 1)$$

Statt $(1/n, \dots, 1/n)^T$ kann man auch eine beliebige andere Wahrscheinlichkeitsverteilung verwenden, um beispielsweise bestimmte Knoten (z.B. mit unerwünschtem Inhalt) zu penalisieren.

EXAMPLE GOOGLE-MATRIX:

Sei der folgende Web-Graph gegeben:



Dann ergibt sich die Google-Matrix G (mit *random leap probability* $\alpha = 0.1$) wie folgt:

$$A^T = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 1 \\ 0 & 2 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}, \quad D = \begin{pmatrix} 1 \\ 5 \\ 1 \\ 1 \\ 2 \end{pmatrix}, \quad H = \begin{pmatrix} 0 & 0 & 0 & 1 & 0.5 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0.4 & 0 & 0 & 0.5 \\ 0 & 0.4 & 1 & 0 & 0 \\ 0 & 0.2 & 0 & 0 & 0 \end{pmatrix},$$

$$G = \begin{pmatrix} 0 & 0 & 0 & 0.90 & 0.45 \\ 0.90 & 0 & 0 & 0 & 0 \\ 0 & 0.36 & 0 & 0 & 0.45 \\ 0 & 0.36 & 0.90 & 0 & 0 \\ 0 & 0.18 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0.02 & 0.02 & 0.02 & 0.02 & 0.02 \\ 0.02 & 0.02 & 0.02 & 0.02 & 0.02 \\ 0.02 & 0.02 & 0.02 & 0.02 & 0.02 \\ 0.02 & 0.02 & 0.02 & 0.02 & 0.02 \\ 0.02 & 0.02 & 0.02 & 0.02 & 0.02 \end{pmatrix} = \begin{pmatrix} 0.02 & 0.02 & 0.02 & 0.92 & 0.47 \\ 0.92 & 0.02 & 0.02 & 0.02 & 0.02 \\ 0.02 & 0.38 & 0.02 & 0.02 & 0.47 \\ 0.02 & 0.38 & 0.92 & 0.02 & 0.02 \\ 0.02 & 0.20 & 0.02 & 0.02 & 0.02 \end{pmatrix}$$

- Zeigen Sie dass es für diese Matrix einen eindeutigen Eigenwert > 0 gibt mit maximalem Betrag unter allen Eigenwerten der Matrix.

Die Matrix ist irreduzierbar (sie kann als Adjazenzmatrix eines gewichteten Graphen interpretiert werden) und alle ihre Einträge sind (strikt) positiv. Damit ist das Frobenius-Theorem anwendbar und es gibt einen Eigenvektor $\lambda > 0$, so dass $Gv\lambda v$ und $\lambda > |\mu|$ für jeden Eigenwert μ von G .

Sie ist irreduzierbar (gewichtete Adjazenzmatrix eines stark-verbundenen Webs). Alle Einträge sind positiv. Damit ist Frobenius Theorem anwendbar. Der zugehörige Eigenvektor ist der Perron- oder Google- (Eigen-) Vektor.

- Skizzieren Sie wie der zugehörige Eigenvektor iterativ berechnet werden kann!

Sei G die obige Google-Matrix und \mathbf{p}^0 ein beliebiger Startvektor, typischerweise $\mathbf{p}^0 = (1, 0, \dots, 0)$ ein Basisvektor des Vektorraums.

Eine Näherung des PageRank-Vektors v zum obigen Eigenwert wird iterativ berechnet durch die Formel:

$$\mathbf{p}^{k+1} = G\mathbf{p}^k$$

Dies kann z.B. eine vorgegebene Anzahl von Iterationen (je größer das Web desto größer sollte diese Zahl sein, um eine ordentliche Approximation zu erhalten).

Ansonsten wird abgebrochen, wenn sich der Vektor nicht mehr oder nur noch wenig ($\|\mathbf{p}^{k+1} - \mathbf{p}^k\| < \epsilon$) verändert.

- Die Google-Matrix wird aus der *transponierten* Adjazenzmatrix gebildet. Zeigen Sie wie man den obigen Eigenvektor iterativ berechnen kann ohne diese Transponierung.

Zuerst erinnere man sich, dass $(AB)^T = B^T A^T$. Aus $\mathbf{p}^{k+1} = G\mathbf{p}^k$ folgt $(\mathbf{p}^{k+1})^T = (G\mathbf{p}^k)^T$ und damit $(\mathbf{p}^{k+1})^T = (\mathbf{p}^k)^T G^T$. Wenn wir zusätzlich Zeilen-Vektoren ($1 \times m$) statt Spalten-Vektoren ($m \times 1$) betrachten, ergibt sich damit eine attraktive Alternative für die Berechnung des PageRank-Vektors.

HITS

- Was ist das wesentliche Unterscheidungsmerkmal des HITS- im Vergleich zum PageRank-Algorithmus.

Der HITS-Algorithmus ist Anfrage-spezifisch. Der Grund ist, dass er angewandt auf das gesamte Web schlecht funktioniert: Er wird dominiert durch "allgemeine" Hubs (wie yahoo.com), themen-spezifische Hubs treten fallen kaum ins Gewicht. Darüber hinaus ist die Trennung in Hub und Autorität in manchen Fällen ebenfalls themen-spezifisch.

- Wie wird der Anfangsgraph im HITS Algorithmus berechnet?

Das Originalpaper schlägt vor dazu klassischen IR (z.B. Vektorraum-Modell) zu verwenden. Es werden beispielsweise alle Seiten, in denen eines der Anfragekeywords vorkommt (sowie die Kanten dazwischen) betrachtet. Diese Startmenge wird um alle Knoten erweitert, auf die aus dieser Menge direkt verwiesen wird, sowie um alle Knoten die darauf verweisen (möglicherweise gedeckelt).

- Skizzieren Sie den iterativen HITS Algorithmus. Erklären Sie dabei die beiden Update-Operationen.

Wir bezeichnen mit \mathbf{h}^k den Vektor der Hub-Gewichte, mit \mathbf{a}^k den Vektor der Autoritäts-Gewichte, jeweils der k -ten Iteration.

Initialisierung: $\mathbf{h}^0 = \mathbf{a}^0 = (1, \dots, 1)^T$.

Iteration:

- Autoritäts-Update:** $\mathbf{a}_i^{k+1} = \sum_{(j,i) \in E} \mathbf{h}_j^k$. Der neue Autoritäts-Wert von i ist also die Summe aller Hub-Werte von Knoten j , von denen es eine Kante nach i gibt.
- Hub-Update:** $\mathbf{h}_i^{k+1} = \sum_{(i,j) \in E} \mathbf{a}_j^k$. Der neue Hub-Wert von i ist also die Summe aller Autoritäts-Werte von Knoten j , zu denen es eine Kante von i gibt.
- Normalisierung:** Nach jedem Iterations-Schritt muß erneut normalisiert werden. Jeder \mathbf{a}_i^{k+1} (\mathbf{h}_i^{k+1}) wird also durch $\sum_i \mathbf{a}_i^{k+1}$ ($\sum_i \mathbf{h}_i^{k+1}$) geteilt.

Man kann die beiden Update-Operationen auch als Matrizenmultiplikation ausdrücken. Sei A wieder die Adjazenzmatrix des Web-Graphen. Dann ist

$$\mathbf{a}^{k+1} = L^T \mathbf{h}^k \qquad \mathbf{h}^{k+1} = L \mathbf{a}^k.$$

Setzen wir für \mathbf{h}^k nochmals $L \mathbf{a}^{k-1}$ ein (entsprechend für \mathbf{a}^k) so ergibt sich eine neue Iterationsfolge:

$$\mathbf{a}^{k+1} = L^T L \mathbf{a}^{k-1} \qquad \mathbf{h}^{k+1} = L L^T \mathbf{h}^{k-1}.$$

Man beachte, dass das "Überspringen" von Werten keinen Einfluß auf die Konvergenz hat, ja sogar die Geschwindigkeit der Konvergenz (um einen konstanten Faktor) beschleunigt. Die letzteren Fassung ist kompetitiv, da die verwendeten Matrizen vorberechnet werden können und typischerweise sehr dünn besetzt sind.

— SÜ-3.2 —

Extraktion von Web-Tabellen

Vertiefung

Gatterbauer, W., Bohunsky, P., Herzog, M., Krüpl, B., and Pollak, B. 2007. *Towards domain-independent information extraction from web tables*. In Proceedings of the 16th international Conference on World Wide Web (Banff, Alberta, Canada, May 08 - 12, 2007). WWW '07. ACM, New York, NY, 71-80.

<http://www2007.org/papers/paper790.pdf> oder <http://doi.acm.org/10.1145/1242572.1242583> oder Anlagen Gatterbauer2007WebTables.pdf.

“Traditionally, information extraction from web tables has focused on small, more or less homogeneous corpora, often based on assumptions about the use of <table> tags. A multitude of different HTML implementations of web tables make these approaches difficult to scale. In this paper, we approach the problem of domain-independent information extraction from web tables by shifting our attention from the tree-based representation of web pages to a variation of the two-dimensional visual box model used by web browsers to display the information on the screen. The thereby obtained topological and style information allows us to fill the gap created by missing domain-specific knowledge about content and table templates. We believe that, in a future step, this approach can become the basis for a new way of large-scale knowledge acquisition from the current ‘Visual Web.’”

FRAGENKATALOG

1. Was versteht man unter “web data extraction” (oder “Web information extraction”)?

Die Extraktion von *strukturierten* Daten aus Web-Seiten. Strukturierte Daten sind typischerweise

- a) Meta-Daten über Web-Seiten und ihre Beziehungen (z.B. die PageRank-Matrix);
- b) Daten über “Web-Objekte” also typischerweise tabellarisch dargestellte Informationen im Web, wie z.B. Information über ein Buch bei Amazon, eine Vorlesung des MIT, ein Musikstück bei last.fm, etc.;
- c) allgemeine Fakten (z.B. durch natürlichsprachliche Analyse gewonnen).

Bei der ersten Art von Daten sind meist am einfachsten zu gewinnen, die zuverlässigen Gewinnung der zweiten Art von Daten ist eines der zentralen, intensive in Praxis und Forschung untersuchten Probleme des Webs, die dritte Art ist noch auf absehbare Zeit nur mit so geringer Genauigkeit erzielbar, dass die resultierenden Daten für automatische Verarbeitung ungeeignet sind (cf. RDF, Semantic Web zur manuellen Strukturierung solcher Daten).

2. Warum spielen gerade Tabellen eine wichtige Rolle bei der *web data extraction*?

Tabellen sind bereits vorstrukturiert, oft sehr einheitlich innerhalb einer Kategorie von Daten und haben eine einfache Repräsentation.

Typische Anwendungen sind Vergleiche von verschiedenen Web-Shops oder anderen Leistungsangeboten, aber auch Meta-Search etc.

3. Was unterscheidet den im Paper vorgestellten Ansatz von typischen Ansätzen auf dem Gebiet? Warum wählen die Autoren diesen Ansatz?

Der Ansatz verwendet die visuelle Darstellung (*rendering*) der Web-Seiten anstatt der Tag-Struktur. Die Autoren argumentieren, dass die Verwendung der visuellen Struktur erlaubt auch mit komplexeren (z.B. Web 2.0) Web-Seiten umzugehen, bei denen die Tabellen-Struktur nicht unmittelbar aus dem Code ersichtlich ist, sondern durch eine Kombination aus CSS, JavaScript etc. erreicht wird. Für eine visuelle Analyse von Tabellen macht es keinen Unterschied, auf welche Weise die Tabelle erzeugt wird.

4. Fallen Ihnen Gegenargumente gegen den von den Autoren vorgeschlagenen Ansatz ein?

Die beiden Hauptprobleme des Ansatzes sind

- der bisher noch enttäuschende *recall* und, schlimmer, *precision*. Diese sind bereits für Tabelle-Extraktion (als nur die Erkennung wo/was eine Tabelle ist) 81% bzw. 68%. Es ist allerdings zu erwarten, dass sich das durch verfeinerte Algorithmen und die Integration von inhaltsbezogenen Schritten (z.B. Ontologie-basierte Analyse der vorkommenden Tabellen-Label) drastisch verbessern lässt.
- Weniger leicht zu überkommen ist die vergleichsweise schlechte Performance. Zum einen muss bei dem Ansatz jede Web-Seite gerendert werden, was je nach Komplexität mehrere Sekunden erfordern kann. Desweiteren ist auch die Analyse vergleichsweise komplex mit einer Komplexität von $O(n \sqrt{n})$ wobei n die Anzahl der Elemente einer Webseite ist.

5. Beschreiben Sie grob, die Schritte des Extraktions-Algorithmus! Kommentieren Sie die Annahmen, die diesem Algorithmus zugrunde liegen.

- a) Filtern der CSS/DOM Eigenschaften auf eine kleine Menge von ausreichend diskriminierenden Eigenschaften.
- b) Nur Elemente mit den Tags `td`, `th` oder `div` werden weiter behalten.
- c) Nur Tabellen, die eine `hyperbox/frame` formen, werden behalten (keine Tabellen mit "ausreichenden" Zellen).
- d) Elemente mit gleichen Koordinaten werden zusammengeworfen (z.B. geschachtelte `div` Elemente).
- e) Im nächsten Schritt werden die eigentlichen Tabellen gefunden, im wesentlichen indem von jedem Element aus versucht wird, es zusammen mit adjazente Elemente als Tabelle zu interpretieren.
- f) Dann werden Tabellen mit überlappenden Zellen eliminiert.
- g) Tabellen müssen 2-dimensional sein und mindestens 3 Zeilen enthalten.
- h) Reine Layout-Zellen werden aus Tabellen eliminiert.
- i) Tabellen mit einer Zelle, die mehr als 40% der Tabelle abdeckt, werden eliminiert.
- j) Tabellen mit zu großen Zellen (mehr als 20 Wörter) werden eliminiert.

Offensichtlich ist dieser Algorithmus, wie die Autoren ausdrücklich sagen, vorläufig und ad-hoc. Er ist insbesondere auf *object data* Extraktion spezialisiert, also z.B. Produktlistings.

6. Wie wird das Ergebnis der Tabellenextraktion in diesem Ansatz dargestellt?

Das Ergebnis wird als generalisiertes (oder schema-loses) n -Tuple, u.U. mit angereichert mit probabilistischen Informationen, dargestellt. Diese Darstellung wird in XML serialisiert. ^{tisch}

Warum? Zuordnung von Tabellen-Zellen zu Dimensionen bei Tabellen die höher-dimensionale Daten repräsentieren oft abhängig von der Interpretation der Daten (also der Domäne).

— SÜ-3.3 —

Inverted Files im Information Retrieval

Vertiefung, Wiederholung

Zobel, J. and Moffat, A. 2006. *Inverted files for text search engines*. ACM Comput. Surv. 38, 2 (Jul. 2006), 6. Für die Beantwortung der Fragen ist **nur Abschnitt 1–3 notwendig** (Seiten 1–12).

<http://doi.acm.org/10.1145/1132956.1132959> oder <http://www.cs.mu.oz.au/~jz/fulltext/compsurv06.pdf> oder Anlagen Zobel2006InvertedFiles.pdf.

“The technology underlying text search engines has advanced dramatically in the past decade. The development of a family of new index representations has led to a wide range of innovations in index storage, index construction, and query evaluation. While some of these developments have been consolidated in textbooks, many specific techniques are not widely known or the textbook descriptions are out of date. In this tutorial, we introduce the key techniques in the area, describing both a core implementation and how the core can be enhanced through a range of extensions. We conclude with a comprehensive bibliography of text indexing literature.”

FRAGENKATALOG

1. *Once more with feeling*: Erläutern Sie den Unterschied zwischen *matching* (und damit Antworten) in relationalen Datenbanken und in Information Retrieval Systemen.

RDB: Antworten müssen logische Bedingungen **exakt** erfüllen.

IR: Antworten haben “Relevanz” für die Anfrage. Dabei wird oft eine Form von Ähnlichkeit zwischen Anfrage und Dokument verwendet.

2. Erklären Sie welche Rolle Ähnlichkeitsmaße in einem IR-System spielen.

Sie bestimmen wie ähnlich ein Dokument zu einer Anfrage ist und damit wie relevant es ist im Vergleich zu anderen Dokumenten (je ähnlicher, desto relevanter). Typische Ähnlichkeitsmaße sind das Cosinus-Maß oder probabilistische Maße, die Dokumente bei denen die Verteilung eines Terms deutlich von der durchschnittlichen abweicht als für diesen Term relevanter betrachten.

3. Wie muss bei großen Dokumentsammlungen das Ähnlichkeitsmaß aussehen, um zu vermeiden, dass alle Dokumente betrachtet werden müssen.

Alle Dokumente, die einen Term nicht enthalten, sollten bzgl. dieses Termes Ähnlichkeit 0 haben, so dass, z.B. über einen invertierten Index direkt nur die Dokumente, die den Term enthalten, betrachtet werden können.

4. Welche Rolle spielt eine Dokument-Index in einem IR-System?

Er soll schnellen Zugriff auf nur diejenigen Dokument ermöglichen, die einen Term enthalten (für einen Term relevant sind). Dies sind üblicherweise signifikant weniger als die Gesamtzahl der Dokumente.

5. Erklären Sie das Prinzip eines *inverted files* (invertierten Index)! Warum heißt dieser Index *invertiert*?

Die übliche (Vektorraum-) Darstellung ist, zu jedem Dokument die enthaltenen Terme zu speichern, z.B. in Form der Dokument-Term-Matrix.

In einem *inverted file* werden hingegen zu jedem Term die Dokumente (genau: deren IDs) gespeichert, die diesen Term enthalten. Dadurch ist es möglich direkt nur diese Dokument (z.B. bzgl. der Berechnung der Ähnlichkeit zur Anfrage) zu betrachten, anstatt über alle Dokumente der Sammlung iterieren zu müssen.

In manchen Fällen wird auch die Position im Dokument mit gespeichert, insbesondere um Nachbarschafts-Anfragen zu ermöglichen (Term t Nahe zu = höchsten x Positionen entfernt Term t').

6. Skizzieren Sie wie mit Hilfe von *inverted files* eine Anfrage aus mehreren Termen bearbeitet wird (z.B. “dark keep night”).

Sei $w_{q,t}$ das Gewicht des Terms t in der Anfrage q (bzw. $w_{d,t}$ im Dokument d). Sei $W_d = \sqrt{\sum_t w_{d,t}^2}$, $f_{d,t}$ die Anzahl der Vorkommen von t in d .

To use an inverted index to rank a document collection with regard to a query q and identify the top r matching documents:

- (1) Allocate an accumulator A_d for each document d and set $A_d \leftarrow 0$.
- (2) For each query term t in q ,
 - (a) Calculate $w_{q,t}$, and fetch the inverted list for t .
 - (b) For each pair $\langle d, f_{d,t} \rangle$ in the inverted list
 - Calculate $w_{d,t}$, and
 - Set $A_d \leftarrow A_d + w_{q,t} \times w_{d,t}$.
- (3) Read the array of W_d values.
- (4) For each $A_d > 0$, set $S_d \leftarrow A_d / W_d$.
- (5) Identify the r greatest S_d values and return the corresponding documents.

Fig. 4. Indexed computation of cosine similarity between a query q and a text collection.

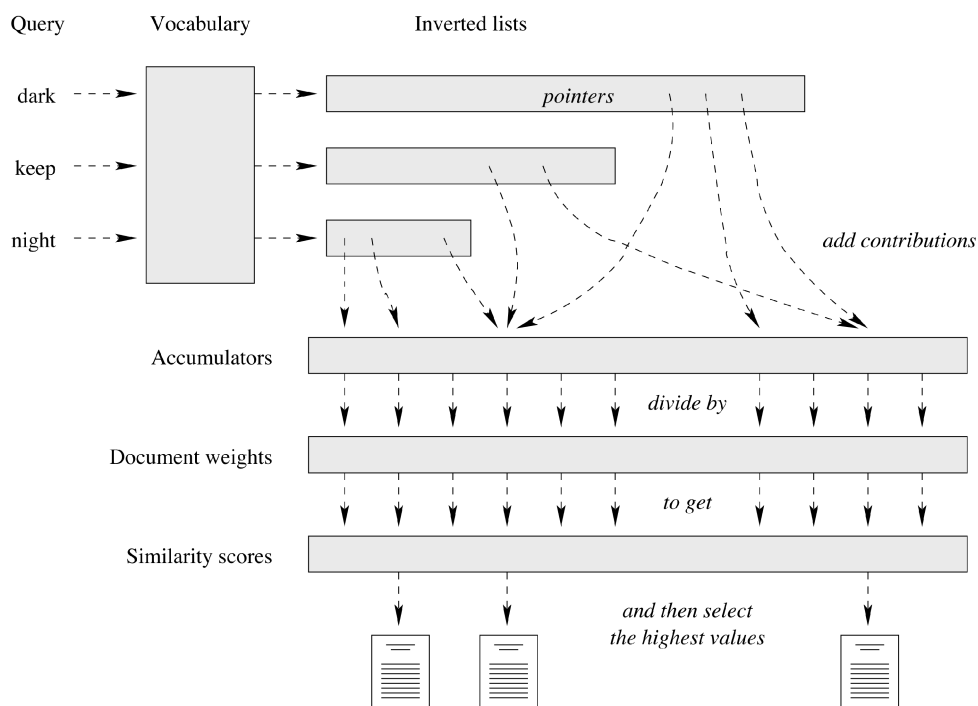


Fig. 5. Using an inverted file and a set of accumulators to calculate document similarity scores.