Einführung
Grundidee
Algorithmus
Ergebnisse
Erweiterungen des Algorithmus
HITS vs PageRank
Zusammenfassung
Literatur

Hyperlink Induced Topic Search (HITS)

Ekaterina Tikhoncheva

Seminar „Information Retrieval“

Universität Heidelberg
19.01.2014

Agenda

1. Einführung
2. Grundidee
3. Algorithmus
   - Schritt 1
   - Schritt 2
   - Konvergenz
4. Ergebnisse
5. Erweiterungen des Algorithmus
6. HITS vs PageRank
7. Zusammenfassung
   - Modifikationen

Einführung

- HITS ist ein Algorithmus zum Finden und Bestimmen der Rangordnung der zu einer gegebenen Suchanfrage relevanten Seiten im WWW[1]
- Entwickelt von Jon M. Kleinberg, 1997 IBM Research Report RJ 10076
- Idee: finde autoritäre Informationsquellen (engl. **authorities**) zu der gegebenen Suchanfrage
- Mathematischer Hintergrund: Eigenvektoren der mit dem Web-Graphen verbundenen Matrix

[1] in einer Hyperlinked-Umgebung

Drei Typen der Suchanfragen

- spezielle:
  „Does Netscape support the JDK 1.1 - code-signing API?“[7]
- breit angelegte:
  „Find information about the Java programming language“[7]
  - Suche nach ähnlichen Seiten
    „Find pages similar to *java.sun.com*“[7]

Drei Typen der Suchanfragen

- spezielle: Problem der Knappheit
  „Does Netscape support the JDK 1.1 - code-signing API?“[7]
- breit angelegte:
  „Find information about the Java programming language“[7]
  - Suche nach ähnlichen Seiten
    „Find pages similar to *java.sun.com*“[7]

Drei Typen der Suchanfragen

- spezielle: Problem der Knappheit
  „Does Netscape support the JDK 1.1 - code-signing API?“[7]
- breit angelegte: Problem der Vielfältigkeit
  „Find information about the Java programming language“[7]
  - Suche nach ähnlichen Seiten
    „Find pages similar to *java.sun.com*“[7]

Ranking

- Man möchte die meist relevanten und nützlichen Seiten (Autoritätsseiten) aus der Menge aller zu der Anfrage relevanten Seiten finden
- Mögliche Hindernisse:
  - die meist relevanten Seiten werden nicht unbedingt durch ein textbasiertes Ranking vorgezogen
  - es kann sein, dass die relevanten Seiten die Wörter aus der Suchanfrage gar nicht enthalten

Authorities und Hubs I

Annahme

Die Relevanz zwischen zwei Seiten wurde vom Ersteller des Links zwischen diesen Seiten geprüft

Stimmt im Allgemeinen nicht (z.B. Navigationslinks, Werbung)

Aber unter dieser Annahme reicht es nur die Linkstruktur des WWW zu betrachten, um die Autorität einer Seite im Bezug auf eine andere zu bestimmen

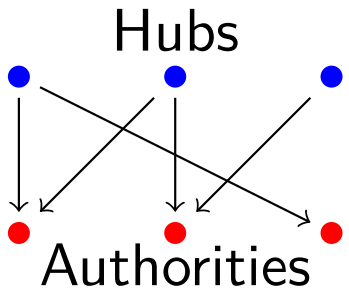
Authorities und Hubs II

Authorities (Autoritätsseiten)

Relevante Seiten, auf die viele weitere relevante Seiten zeigen

Problem: populäre Seiten kommen immer als Authorities vor

Es muss eine Überlappung in den Links, die auf Authorities zeigen, berücksichtigt werden



Hubs

Seiten, die auf viele Authorities zeigen

HITS Algorithmus

Der Algorithmus besteht aus zwei Schritten:

1. Konstruiere einen auf die Suchanfrage fokussierten Teilgraphen des WWW
2. Berechne rekursiv die Authorities- und Hubgewichte für alle Knoten im Teilgraphen aus Schritt 1

Teilgraph vom WWW

Wir stellen das ganze Web als ein gerichteter Graph $G = (V, E)$ dar. Gesucht wird ein auf der Anfrage basierter Teilgraph vom WWW

Sei eine Suchanfrage σ gegeben. Wir bezeichnen mit $\mathbf{Q}_\sigma$ das Ergebnis der textbasierten Suche

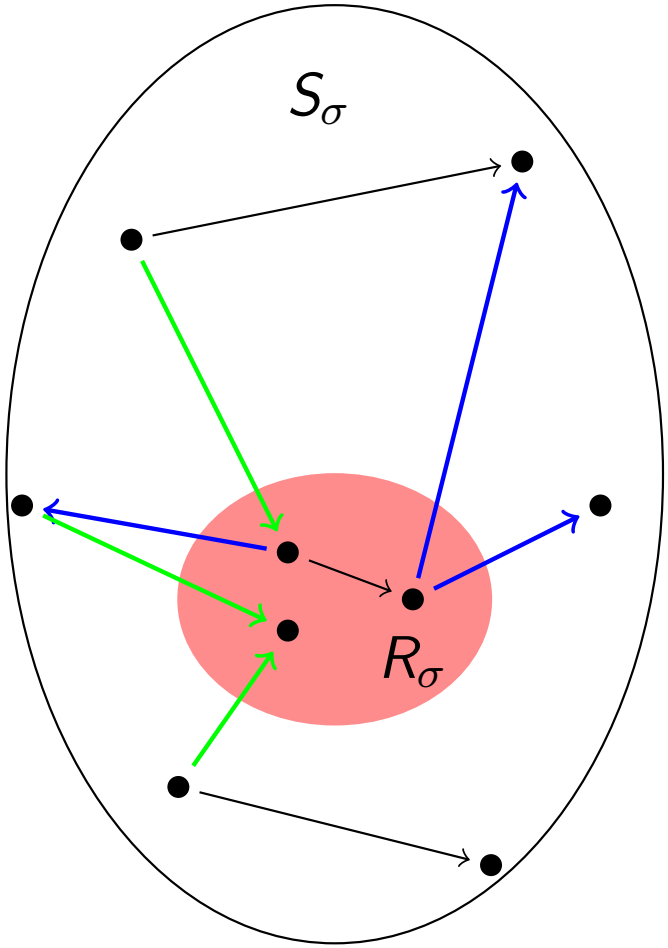
- Q_σ ist groß
- Gute Authorities müssen nicht in Q_σ liegen

Gesucht wird eine Menge $\mathbf{S}_\sigma$ von Seiten, die

- klein ist
- relevante Seiten enthält
- viel gute Authorities enthält

Algorithm 1 Teilgraphen(σ, E, t, d)

1: R_σ ist die Menge der t besten Ergebnisse des textbasierten Suchalgorithmus E angewendet auf die Anfrage (engl. **root set**) σ
2: $S_\sigma := R_\sigma$
3: **for** p in R_σ **do**
4: $S_\sigma = S_\sigma \cup \{Menge\ der\ Seiten,\ auf\ die\ \mathrm{p}\ zeigt\}$
5: $S_\sigma = S_\sigma \cup \{Menge\ der\ \mathrm{d}\ Seiten,\ die\ auf\ \mathrm{p}\ zeigen\}$
6: **end for**



Bemerkungen

Algorithm Teilgraphen(σ, E, t, d)

1: R_σ ist die Menge der **t** besten Ergebnisse des textbasierten Suchalgorithmus E angewendet auf die Anfrage (engl. **root set**) σ
2: $S_\sigma := R_\sigma$
3: **for** p in R_σ **do**
4: $S_\sigma = S_\sigma \cup \{Menge\ der\ Seiten,\ auf\ die\ p\ zeigt\}$
5: $S_\sigma = S_\sigma \cup \{Menge\ der\ \mathbf{d}\ Seiten,\ die\ auf\ p\ zeigen\}$
6: **end for**

In der Implementierung vom Autor:

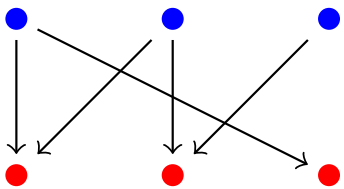
- t = 200
- d = 50
- $|S_\sigma| \sim O(10^3)$
- Lösche Link zwischen den gleichen Domains (Navigationslinks)
- Beschränke Anzahl der Seiten aus einer Domain, die auf die gleiche Seite verweisen (z.B. Werbung)

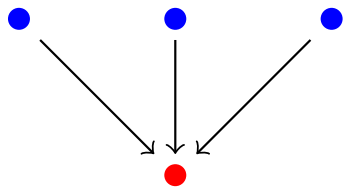
Berechnen von Authorities und Hubs

Der erste Schritt erlaubt es uns, sich auf die Menge S_σ der Seiten zu konzentrieren, die viele relevante und autoritäre Seiten enthält und gleichzeitig klein genug ist, um die nicht trivialen Algorithmen anwenden zu können

Annahmen

- Ein guter Hub zeigt auf viele gute Authorities
- Eine gute Autoritätsseite wird von vielen guten Hubs hingewiesen





Sei Graph $G = G[S_\sigma]$ aus dem Schritt 1 gegeben. Wir bezeichnen mit A die Adjazentmatrix des Graphen G

Für jede Seite $p \in V$ werden ein nicht negatives **Autoritätsgewicht x(p)** und nicht negatives **Hubgewicht y(p)** definiert, so dass es $\sum_{p \in V(G)} x^2(p) = 1$ und $\sum_{p \in V(G)} y^2(p) = 1$ gelten

Es werden zwei Operationen definiert:

$$\mathcal{I}(\mathbf{y_i}) = \sum_{q:qp \in E(G)} y_i(q) = A^T y$$

$$\mathcal{O}(\mathbf{x_i}) = \sum_{q:pq \in E(G)} x_i(q) = Ax$$

Algorithm 3 Iteriere(G, k)

1: $z = (1, \ldots, 1) \in \mathbb{R}^n$
   // $n = |V(G)|$
2: $x_0 := z$
3: $y_0 := z$
4: **for** $i = 1, \ldots, k$ **do**
5: $\quad x_i' := \mathcal{I}(y_{i-1})$
6: $\quad y_i' := \mathcal{O}(x_i')$
7: $\quad$ normiere $x_i' \rightarrow x_i$
   $\sum_{p \in V(G)} x_i^2(p) = 1$
8: $\quad$ normiere $y_i' \rightarrow y_i$
   $\sum_{p \in V(G)} y_i^2(p) = 1$
9: **end for**
10: **return** (x_k, y_k)

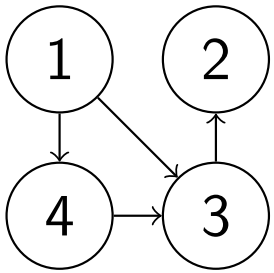
Algorithm 4 Filter(G, k, c)

1: $(x_k, y_k) = Iteriere(G, k)$
2: Authorities := {Seiten mit c größten Einträgen von x_k}
3: Hubs := {Seiten mit c größten Einträgen von y_k}
4: **return** (Authorities, Hubs)

$$\mathcal{I}(y_i) = \sum_{q:qp \in E(G)} y_i(q)$$

$$\mathcal{O}(x_i) = \sum_{q:pq \in E(G)} x_i(q)$$

Beispiel



Adjazenzmatrix $A = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

x_0	(1,1,1,1)	y_0	(1,1,1,1)
x_1	(0, 0.41, 0.82, 0.41)	y_1	(0.80, 0, 0.27, 0.53)
x_2	(0, 0.17, 0.85, 0.51)	y_2	(0.84, 0, 0.11, 0.53)
x_3	(0, 0.07, 0.85, 0.52)	y_3	(0.85, 0, 0.04, 0.53)
x_4	(0, 0.03, 0.85, 0.53)	y_4	(0.85, 0, 0.02, 0.53)
x_5	(0, 0.01, 0.85, 0.53)	y_5	(0.85, 0, 0.01, 0.53)
x_6	(0, 0, 0.85, 0.53)	y_6	(0.85, 0, 0, 0.53)
x_7	(0, 0, 0.85, 0.53)	y_7	(0.85, 0, 0, 0.53)

Konvergenz

Aus dem Algorithmus folgt $y_k = (AA^T)^k z$

Sei $w_1(AA^T)$ der Haupteigenvektor von AA^T, d.h. Vektor zu dem betragsgrößten Eingenwert
Man kann zeigen, dass der Vektor z zu $w_1(AA^T)$ nicht orthogonal ist und da die Matrix AA^T symmetrisch ist, konvergiert die Reihenfolge $\{y_k\}$ gegen $w_1(AA^T)$ (Power Iteration, Potenzmethode [5]). Außerdem hat $w_1(AA^T)$ nur positive Einträge (siehe Peron-Frobenius Theorem)

Analog konvergiert $x_k = (A^T A)^{k-1} A^T z$ gegen $w_1(A^T A)$

In der Praxis lässt man der Algorithmus nicht bis zu Konvergenz laufen, sondern macht eine feste Anzahl k[2] von Iterationen

[2] k=20 sei ausreichend [7]

Beispiel (Fortsetzung)

$$A = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$A^T A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

Eigenwerte	Eigenvektoren
$\lambda_1 = 0$	(1,0,0,0)
$\lambda_1 = 0.38$	(0,0,-0.53,0.85)
$\lambda_1 = 1$	(0,1,0,0)
$\lambda_1 = 2.62$	(0,0,0.85,0.53)

$$AA^T = \begin{pmatrix} 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Eigenwerte	Eigenvektoren
$\lambda_1 = 0$	(0,1,0,0)
$\lambda_1 = 0.38$	(-0.53,0,0,0.85)
$\lambda_1 = 1$	(0,0,1,0)
$\lambda_1 = 2.62$	(0.85,0,0,0.53)

Somit sehen wir, dass die Reihenfolgen $\{x_k\}$ und $\{y_k\}$ konvergieren entsprechend gegen $w_1(A^T A)$ und $w_1(AA^T)$.

Ergebnisse I

HITS [7]

(Gates) Authorities

.643 http://www.roadahead.com/	*Bill Gates: The Road Ahead*
.458 http://www.microsoft.com/	*Welcome to Microsoft*
.440 http://www.microsoft.com/corpinfo/bill-g.htm	

www.ask.com

http://www.ask.com/wiki/Bill_Gates?lang=en	
www.gates.com/	Gates Corporation
www.gates.com/all-search-tools/automotive-interchange-search-results	Gates Corporation (manufacture)
usmilitary.about.com/od/glossarytermsg/g/g2641.htm	gate
www.gatesfoundation.org/	Bill & Melinda Gates Foundation

Ergebnisse II

www.google.com

www.gates.com/	Gates Corporation
ww2.gates.com/germany/	Gates Corporation
de.wikipedia.org/wiki/Bill_Gates	
http://www.gatesbbq.com	Gates Bar B. Q.
http://www.gatesfoundation.org	Gates Foundation

www.search.yahoo.com

www.gates.com/	Gates Corporation
www.gateschevyworld.com	Gates Chevy World - Mishawaka, Elkhart, South Bend
www.gatecrafters.com/design_your_gates.aspx	Driveway Gates | Automatic Gates | Electric Gates
www.gates.com/industries/automotive	Gates Automotive Aftermarket Solutions | Gates Corporation
http://www.gatesbbq.com	Gates Bar B. Q.

Ergebnisse III

HITS [7]

(java) Authorities

.328 http://www.gamelan.com/	*Gamelan*
.251 http://java.sun.com/	*JavaSoft Home Page*
.190 http://www.digitalfocus.com/digitalfocus/faq/howdoi.html	*The Java Developer: How Do I...*
.190 http://lightyear.ncsa.uiuc.edu/~srp/java/javabooks.html	*The Java Book Pages*
.183 http://sunsite.unc.edu/javafaq/javafaq.html	*comp.lang.java FAQ*

www.ask.com

www.ehow.com	What is Java?
www.ask.com/wiki/Java_(programming_language)?lang=en	Wiki
java.com/download	Download java
java.about.com/	about.com
www.oracle.com/technetwork/java/index.html	ORACLE

Ergebnisse IV

www.google.com

www.java.com/de/download/	Download der kostenlosen Java-Software
www.java.com/de/	java.com: Java + Sie
www.java.com/download	Download Free Java Software
www.java.com/en/	java.com: Java + You
de.wikipedia.org/wiki/Java_(Programmiersprache)	

www.search.yahoo.com

www.java.com	Java - Official Site
java.com/en/download	Download Java
www.oracle.com/technetwork/es/java/javase/overview/index.html	Java SE - Downloads
windows.microsoft.com/en-US/internet-explorer/install-java	Install Java in Internet Explorer
www.java.net	java.net - Official Site

Ergebnisse V

In seiner Arbeit spricht der Autor über folgende Evaluationsergebnisse:

Es wurden die Suchergebnisse von Yahoo!, CLEVER[3][2] und AltaVista[1] verglichen, indem 37 Freiwillige die Qualität der von verschiedenen Suchmaschinen vorgeschlagenen Seiten zu den fest gewählten Themen abschätzen sollten

Die Ergebnisse sind:
In 50% der Fälle CLEVER war besser.
In 31% Yahoo! und CLEVER waren gleich gut.
In 19% Yahoo! war besser.

[3] Implementation von HITS mit Erweiterungen

Suche nach ähnlichen Seiten

- Die Suchanfrage σ sei eine Internetseite
  $R_\sigma = \{$Menge der Seiten, die auf σ zeigen$\}$
- Wende den HITS Algorithmus an

(www.honda.com) Authorities

.202 http://www.toyota.com/	*Welcome to @Toyota*
.199 http://www.honda.com/	*Honda*
.192 http://www.ford.com/	*Ford Motor Company*
.173 http://www.bmwusa.com/	*BMW of North America, Inc.*
.162 http://www.volvocars.com/	*VOLVO*
.158 http://www.saturncars.com/	*Welcome to the Saturn Web Site*
.155 http://www.nissanmotors.com/	*NISSAN - ENJOY THE RIDE*
.145 http://www.audi.com/	*Audi Homepage*
.139 http://www.4adodge.com/	*1997 Dodge Site*
.136 http://www.chryslercars.com/	*Welcome to Chrysler*

Mehrere Mengen von Hubs und Authorities

- Der betrachtete Algorithmus liefert die am dichtesten vernetzte Menge der Authorities und Hubs im auf der Suchanfrage basierten Teilgraphen des WWWs
- Es gibt eventuell weitere dicht vernetzte Mengen, die zu der Anfrage relevant sind (z.B. mehrdeutige Anfrage)
- betrachte weitere Eigenvektoren, sortiert nach Betragsgrößen
- betrachte positive und negative Teile eines der nicht Haupteigenvektors

HITS

- wird auf einem Teilgraphen des WWW angewendet
- einfacher, aber langsam in Echtzeit
- berechnet Autoritätsseiten und Hubs
- beruht auf textbasierter Suche
- kann „diffundieren“

PageRank

- benutzt ganzen Graphen des WWW
- dauert lang, wird aber nur einmal berechnet
- berechnet nur Autoritätsseiten
- unabhängig von dem Kontext

Zusammenfassung: I

- Gewichtung ist sehr wichtig, besonders für breit angelegte Anfragen
- Als Ergebnis der Suche werden t Autoritätsseiten zurückgeliefert
- Jeder Link enthält einen gewissen Anteil der Autorität des Erstellers des Links
- Autoritätsseiten sind zu der Anfrage relevante Seiten mit vielen eingehenden Links
- Hubs sind Seiten, die auf viele Autoritätsseiten verlinken
- Der ganze WWW-Graph ist zu groß $\Rightarrow$ betrachte auf der Anfrage basierten Teilgraphen

Zusammenfassung: II

- HITS-Algorithmus
  - arbeitet auf dem generierten Teilgraphen
  - ist ein rekursiver Algorithmus
  - konvergiert gegen Haupteigenvektoren der Matrizen $A^T A$ und AA^T[4]
  - liefert Seiten zurück, die den c größten Einträgen des Haupteigenvektors der Matrix $A^T A$ entsprechen
  - kann für die Suche nach ähnlichen Seiten benutzt werden
  - kann die für verschiedene Bedeutungen der Anfrage relevanten Seiten liefern
- HITS wurde in der Suchmaschine CLEVER (IBM Project) implementiert (geschlossen)
- HITS wird in der Suchmaschine Ask.com benutzt ([3])

[4] A ist die Adjazentmatrix des betrachteten Teilgraphen

Modifikationen

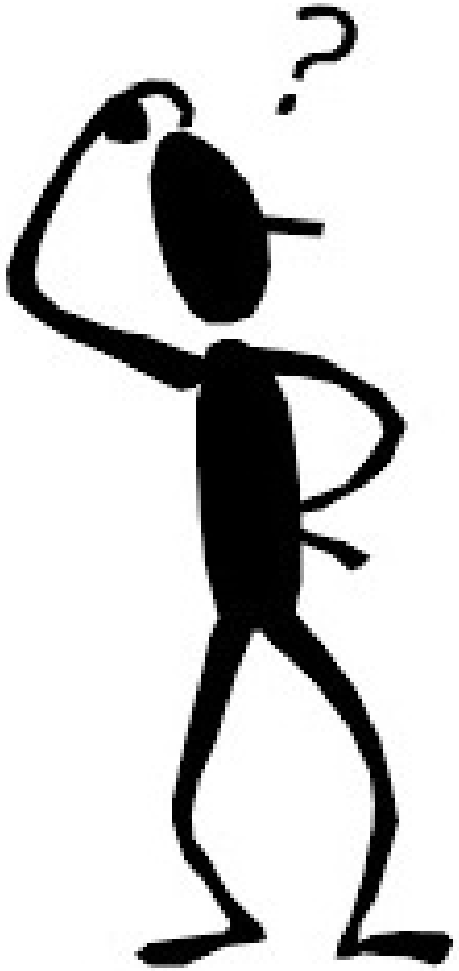
- Problem :

  Topic Drift

  Ergebnisse bei bestimmten Strukturen des Webs sind nicht intuitiv klar [9]
- Modifikationen des Algorithmus
  - Gewichteter HITS-Algorithmus ([4, 8, 9])
  - HITS-Algorithmus mit potenzierter Eingabe ([9])

The End

Vielen Dank für Ihre Aufmerksamkeit!



References I

[1] Altavista. www.de.wikipedia.org/wiki/AltaVista. Accessed: 2014-15-01.

[2] CLEVER project. www.en.wikipedia.org/wiki/CLEVER_project. Accessed: 2014-15-01.

[3] Lecture 4: Hits algorithm - hubs and authorities on the internet. www.math.cornell.edu/~mec/Winter2009/RalucaRemus/Lecture4/lecture4.html. Accessed: 2014-15-01.

References II

[4] Krishna Bharat and M. R. Henzinger. Improved Algorithms for Topic Distillation in a Hyperlinked Environment. *Proceedings of the 21st ACM SIGIR Conference on Research and Development in Information Retrieval*, 1998.

[5] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

[6] Joni Pajarinen. The pagerank/hits algorithms. `www.cis.hut.fi/Opinnot/T-61.6020/2008/pagerank_hits.pdf`. Accessed: 2014-15-01.

[7] John M. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.

References III

[8] Yi Shang Li, Longzhuang and Wei Zhang. Improvement of HITS-based algorithms on web documents. *Proceedings of the 11th international conference on World Wide Web. ACM*, 2002.

[9] J. C Miller, G. Rae, and F. Schaefer. Modifications of Kleinberg's HITS Algorithm Using Matrix Exponentiation and Web Log Records. *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieva*, pages 444–445, 2001.

[10] Jannik Strötgen. Beamer theme. `http://dbs.ifi.uni-heidelberg.de/?id=79`. Accessed: 2014-15-01.

References IV

[11] Zhe Zhao. Authoritative sources in a hyperlinked environment, Jon M. Kleinberg. `web.eecs.umich.edu/~michjc/eecs584/notes/lecture19-kleinberg.pdf`. Accessed: 2014-15-01.