自然言語処理入門
「現実的なご利益：情報検索」

東京大学　情報基盤センター
（情報理工学系研究科、情報学府　兼担）
中川裕志

情報要求

- ユーザがある問題を解決するために、現有の知識だけでは不十分であると感じている状態が情報要求(information need) な状態。

1. 直感的要求（visceral need)： 要求を認識しているが言語化できない状態
2. 意識された要求(conscious need)： あいまいな、あるいはまとまりのない表現でしか言語化できない状態
3. 形式化された要求(formalized need)： 具体的な言語表現
4. 調整済み要求(compromised need)： 問題解決に必要な情報源が同定できるくらいに具体化された要求

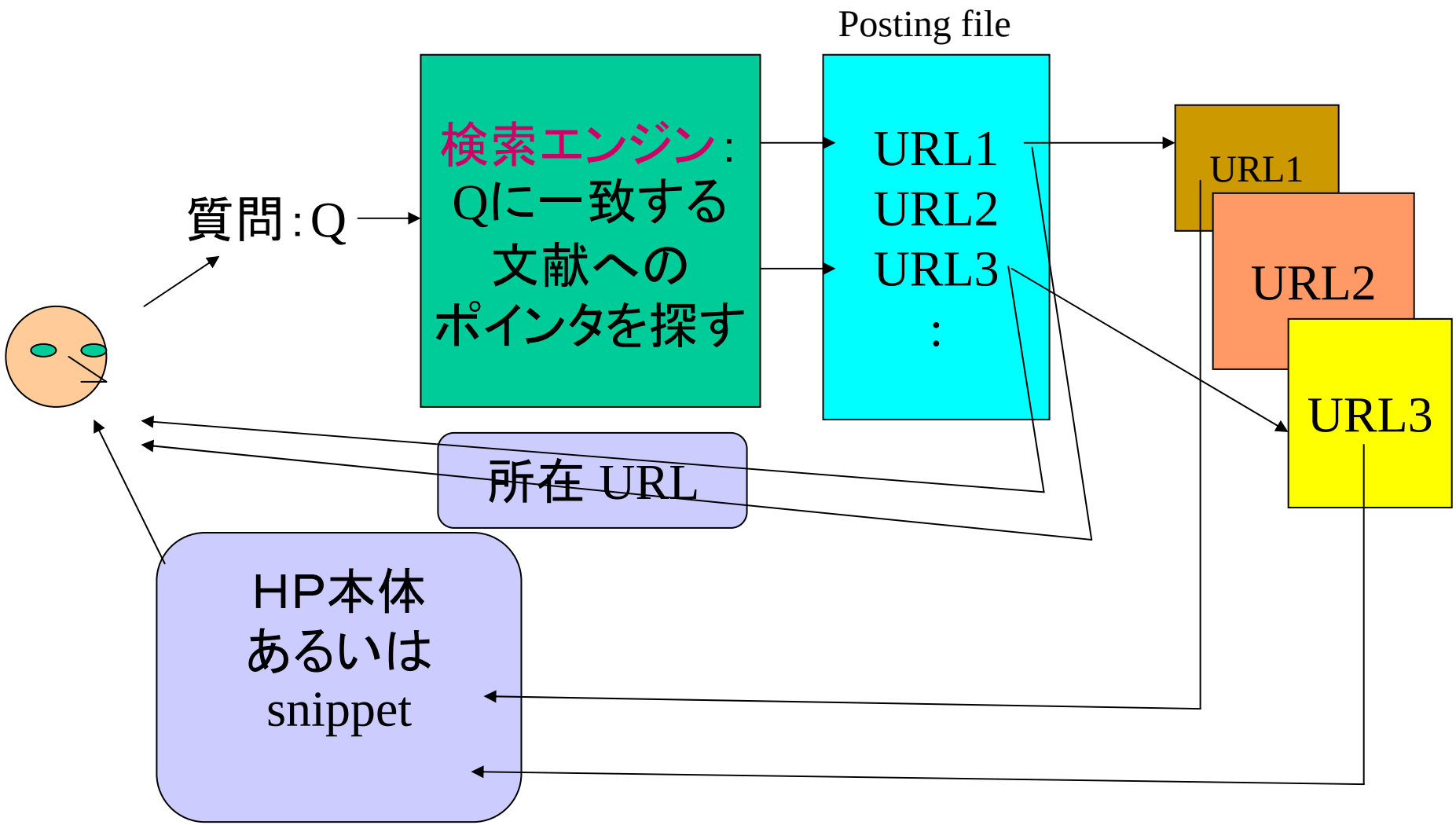
情報要求

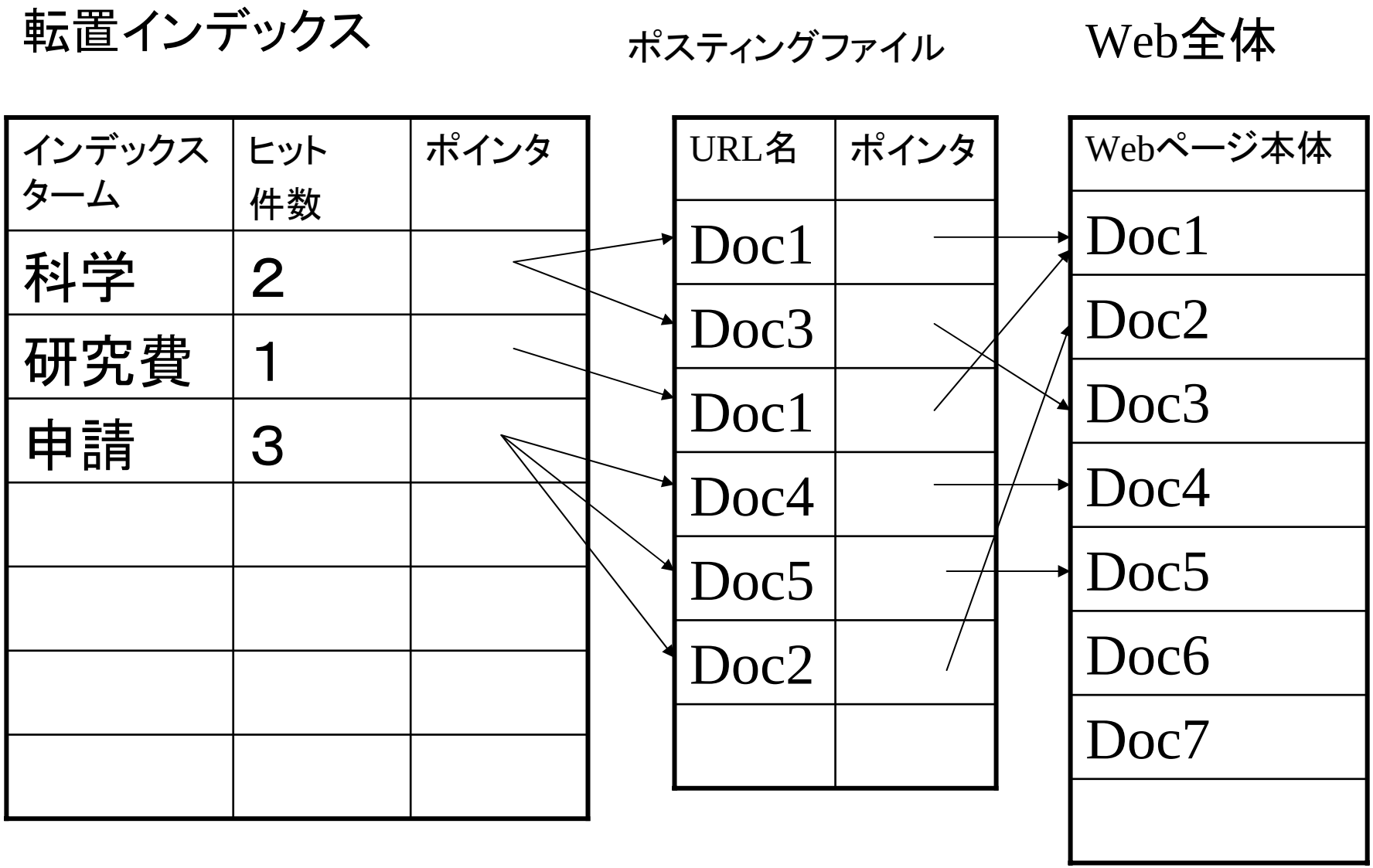
- ユーザがある問題を解決するために、現有の知識だけでは不十分であると感じている状態が情報要求(information need) な状態。
- 広義の情報検索：ユーザの持つ問題を解決できる情報を見つけ出す事。問題はユーザが与える。
- 狭義の情報検索：ユーザの検索質問に適合する文書を文書集合中から見つけ出す事
- 情報の解釈をユーザが行なう= 情報検索
- ユーザがある問題を解決するために、現有の知識だけでは不十分であると感じている状態が情報要求(information need) な状態。
- 情報の解釈をシステムが行なう= 人工知能
- データベースの場合、ユーザはデータベースの構造、内容まで知った上で定型的な検索質問を行なう。つまり、文書集合を対象とする情報検索よりもさらにユーザは情報内容に通じていなければならない。実際、ユーザにとっては大変な負担であり、データベース検索の専門家が必要になったり、典型的な質問例を利用したりすることになる。

検索システムおよびインデクシング

検索エンジンの仕掛け

検索システムの構造


Posting file
検索エンジン:
Qに一致する
文献への
ポインタを探す
質問:Q
URL1
URL2
URL3
:
URL1
URL2
URL3
所在 URL
HP本体
あるいは
snippet


転置インデックス
ポスティングファイル
Web全体
インデックスターム
ヒット件数
ポインタ
科学
２
研究費
１
申請
３
URL名
ポインタ
Doc1
Doc3
Doc1
Doc4
Doc5
Doc2
Webページ本体
Doc1
Doc2
Doc3
Doc4
Doc5
Doc6
Doc7

検索システムの構造

1. 検索対象の文書各々は、それに現れる複数の語(以下「ターム(term) 」と呼ぶ) によって特徴付けられる。タームによる文書の特徴付けをインデクシング(indexing) という。
2. 利用者は自分が欲しい文献を特定する記述を質問Q として検索システムに与える。質問は、1 個以上のキーワード、あるいは自然言語文で記述される
3. 検索エンジンは、Q を解釈して、それに適合する文書を検索する。適合する文書は一般的には複数存在する。したがって、検索結果は、ポスティングファイルへのポインタの集合である。ポスティングファイルは、(文書識別子、文書の格納されているアドレス) というペアからなる。
4. 文書本体は検索システム内で集中管理されている場合と、インターネット上に散在する場合がある。後者の場合、ポッスティングファイルの文書格納アドレスは、URL あるいはURI である。
5. 利用者には検索結果として検索された文書あるいは文献名、URL などのアドレスが返される。多くの場合、文献名には文献のURL へアクセスできるリンクが張ってあり、クリックすればURL にアクセスして文献が表示できる。

ターム と 検索

1. 統制ターム: 予め決められたタームだけを使う。タームの質は統制により維持されるが、利用者が統制タームしか質問に使えない。
2. 自由ターム: 任意のタームを質問に使える。後に述べる全文検索では文書中のタームを網羅するので、自由タームになる

- 完全一致検索（ブーリアン検索）
- 最良優先検索（ベクトル空間法など）

ブーリアン検索

- 質問は論理式。例えば、Q= 科学or(研究費and 申請)
- 転置ファイルを検索して、タームt に対応する文書の集合D(t) が得られるとしよう。すると、質問の論理式(1) によって検索した結果は次のようになる。
- D(科学) or (D(研究費) and D(申請))
- 質問論理式を作ることが素人には難しい。
- 検索結果の文書数が多過ぎたり、あるいは0 だったりすると、論理式を調整して適当な大きさの結果集合にするのだが、この調整が非常に難しい。

転置インデックスの高速検索

- 膨大な数のキーワードの中から質問として与えられたキーワードを高速に検索することが重要

- B木
- トライ
- PATRICIA木

全文検索

- メモリ中にあたかも全文が格納されているかのような状態で検索
- 本当に全文が格納されている場合＝長大な文字列マッチ処理
- 本当はなんでもかんでもインデックスされた場合＝インデックスの効率化
- 全文検索でできそうな検索質問
  「ターム「情報」の前後n 文字以内の範囲にターム「収集」が現れる」
  「ターム「分析」と「統合」が同じ段落に現れる」
- インデックスの作り方

- 文字列を十分に長くとれば全ての半無限長文字列は異なるものになる。その上で文字列を辞書式にソートすると言語の統計で説明したKWIC になる。そして、葉が反無限長文字列の開始位置をデータに持つTrie やPATRICIA 木を作れば、KWIC は全文検索向きのインデクスとして使える。
- 半無限長文字列の代わりに1 単語を葉とするTrie やPATRICIA 木にするとコンパクトになるが、葉には文書において複数回現れるその単語開始位置を全て記述しておく。これもインデクスとして使える。

Suffix Array

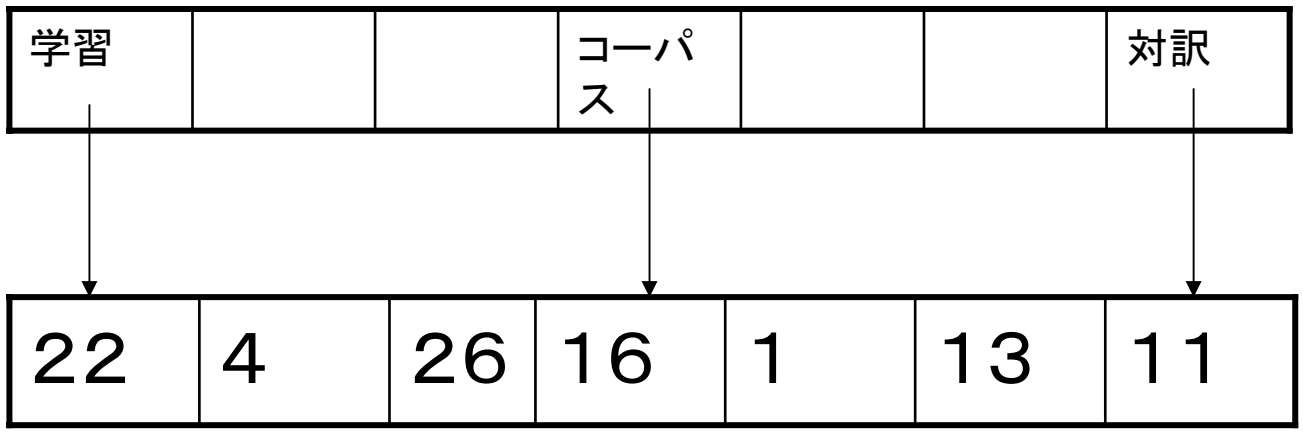
最近の機械翻訳では、対訳辞書をコーパスから学習する研究が多い

1　4　11　13　16　22　26

- この例文の Suffix Array は辞書式にならべた各単語の開始位置を記載した配列

22	4	26	16	1	13	11

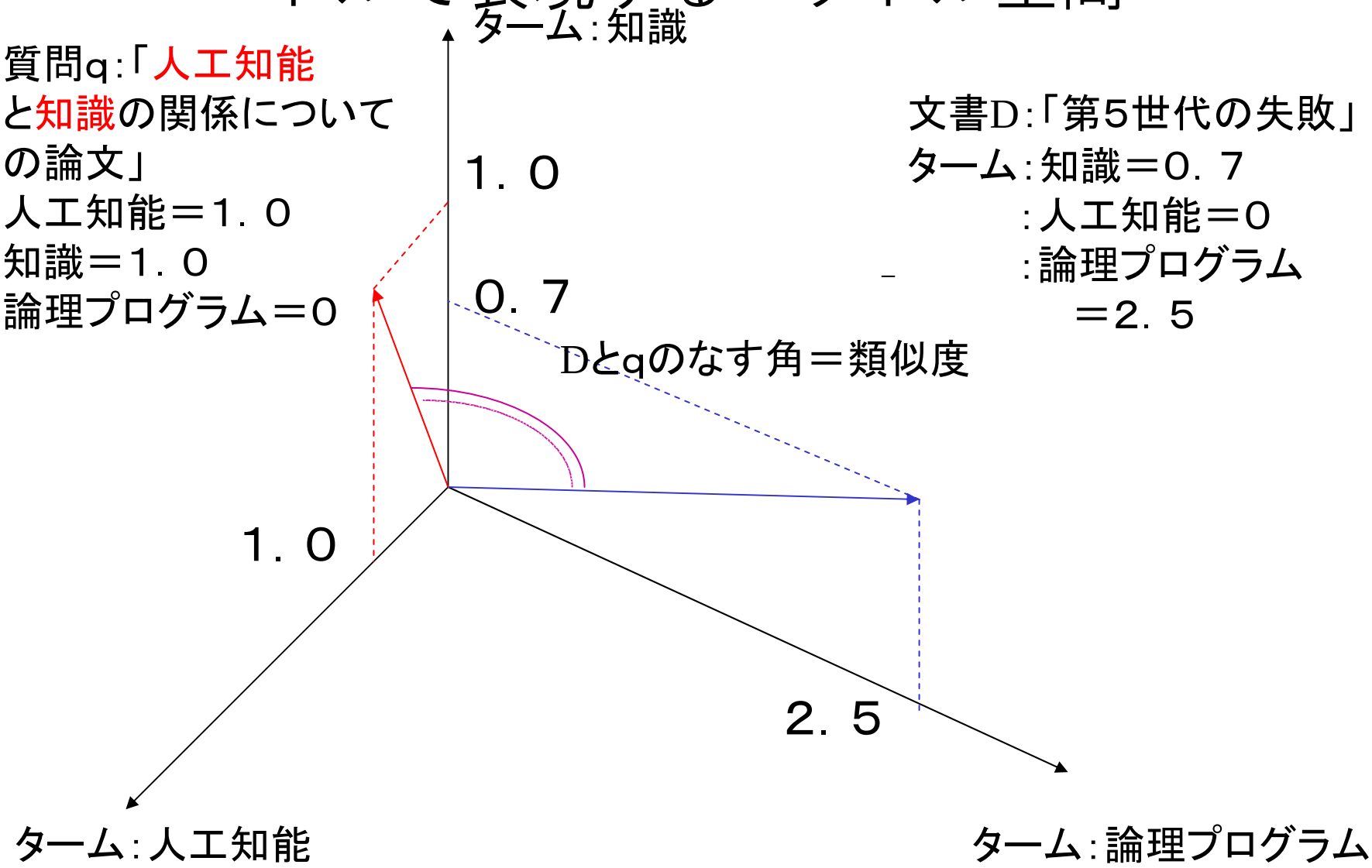
- 2分探索で高速サーチ
- Supra Index



ベクトル空間法

- 最良優先検索
- タームの重み付けと類似度
- パッセージ検索

各タームを次元にし、質問と文書をベクトルで表現するベクトル空間



タームの重み　その1ターム頻度

- タ ー ム頻度(Term Frequency: tf)

$$\mathrm{tf}_{\mathrm{j}}^{\mathrm{i}} = \mathrm{freq}(\mathrm{i},\mathrm{j})$$

- freq(i; j) = 文書Dj におけるタームt i の出現頻度。
- 変形版tf

$$\mathrm{tf}_{\mathrm{j}}^{\mathrm{i}} = K + (1-K)\frac{\mathrm{freq}(\mathrm{i},\mathrm{j})}{\max_{\mathrm{i},\mathrm{j}}(\mathrm{freq}(\mathrm{i},\mathrm{j}))}$$

$$\mathrm{tf}_{\mathrm{j}}^{\mathrm{i}} = \frac{\log(\mathrm{freq}(\mathrm{i},\mathrm{j})+1)}{\log(\text{文書 j 中の総ターム種類数})}$$

タームの重み その2 文書頻度

- 文書頻度 Document frequency $df_i = Dfreq(i)$
- ただし、Dfreq(i)はタームtiが出現する文書数
- 実際はその逆数 idf_i を使う

- 文書総数Nによる正規化

$$idf_i = \log \frac{N}{Dfreq(i,j)} + 1$$

タームの重み　その３　tf ·idf

- 文書Djに現れるタームtiの重みwijは、Djには数多く現れ、他の文書にはあまり現れないという性質をもつべき。つまり、文書Djをよく特徴つけることが大切。そこで、前記のtfとidfをかけたものがよい。つまり、　tf ·idf

$$w_j^i = tf_j^i \times idf_i$$

文書ベクトルと質問ベクトルとそれらの類似度　その1

- このようにしてタームtiの重みが決まったので、文書Djのベクトルは、各タームを次元に割り当てた多次元空間におけるベクトルとして表現できる。つまり、

$$\mathbf{D}_j=(w_j^1,w_j^2,....,w_j^m)$$

- 一方質問ｑもタームtiを含めば１、含まなければ０という値にしてベクトルで表現できる。つまり

$$\mathbf{q}=(q_1,q_2,...,q_m)$$

- ただし、mは文書集合における全ての異なりターム数

文書ベクトルと質問ベクトルとそれらの類似度　その２

- さて、情報検索とは、質問ｑに対して類似度の高い文書Ｄｊを探すことなので、類似度simを以下に定義する。これは、ベクトル空間におけるｑとＤ$_j$のなす角θが0に近いほど類似度が高いと考える方法。

$$\mathrm{sim}(\mathbf{q},\mathbf{D}_j)=\frac{q_1 w_j^1+.....+q_m w_j^m}{\sqrt{q_1^2+...+q_m^2}\times\sqrt{(w_j^1)^2+...+(w_j^m)^2}}=\cos\theta$$

- sim の大きい順に検索結果をに並べて質問者に提示する。

ページ間リンクを利用する重みつけ

- Webのホームページはリンクが張られている。これは図書の引用索引に似ている。
- **基本原理**: **多くの良質なページからリンクされているページは、良質なページである**
- **この原理によるアルゴリズム**として有名なのが GoogleのPageRank algorithm と HITS　algorithm

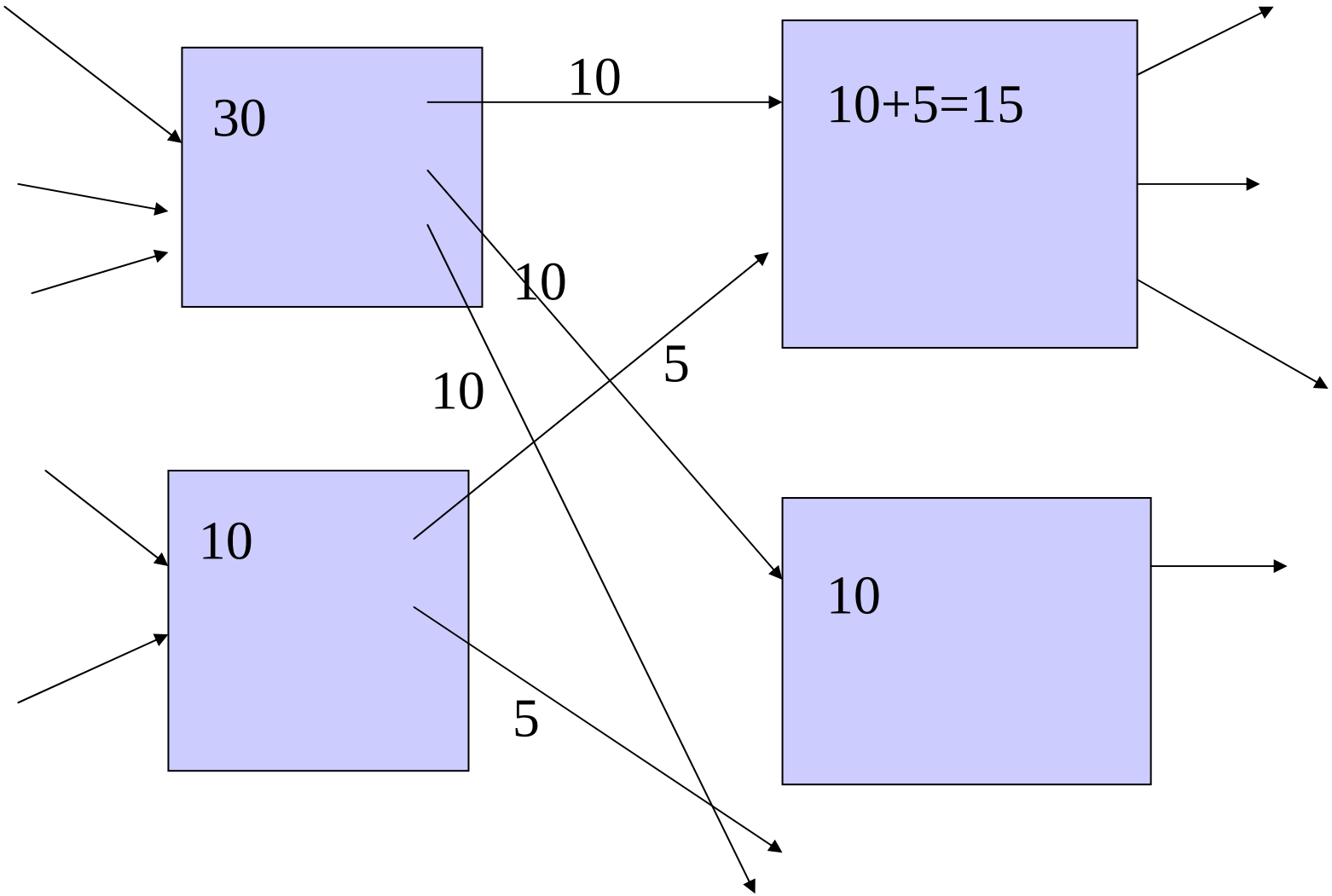
Webページのランキング

- Webページ検索の特異性
  - 文書DBからの検索の場合、使える情報は文書中のテキスト、単語
  - tf×idfによるベクトル空間の類似度ではない重み付けはできないか？
  - Webページの検索の場合、テキスト、単語に加え、リンク情報が使える
  - リンクは（テキストや単語と異なり）、Web（＝データベース）の大域的構造
  - リンクを利用する検索結果のランキングについて説明する。代表的なPageRankについて説明する。

PageRank algorithm のアイデア

- 多くの良質なページからリンクされているページは、やはり良質なページである
- 利用者がページをリンクをたどってサーフする状況を模擬する。(良いページには沢山の利用者がいると考えればよい)
  - →
  - ページの重みが決められている
  - リンク元ページからの流入する重み総和がそのページの重み
  - リンク先へ流入する重みは、そのページの重みをリンク先の本数で割ったもの

ページ重みの概念図



ところが問題は

- あるページPiの重みはPiへのリンク元のページの重みが確定しないと計算できない。
- めぐりめぐって、Piの重みが確定しないとPiへのリンク元のページの重みが確定しないと計算できない。
- →
- 各ページの重みを要素とし、全ページ数の次元を持つベクトル R を考え、このベクトルを数学的に求めることにする

ページ重みの計算法

- 接続行列A=(aij)で表現
  - if ページiからページjにリンク then aij=1/Ni
    (Niはiからでるリンク数)
    otherwise aij=0
- ページのベクトル R は以下の式を満足する
- R=cAR つまり

$$\begin{bmatrix} r_1 \\ \vdots \\ r_n \end{bmatrix} = c \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} r_1 \\ \vdots \\ r_n \end{bmatrix}$$

しかし、その1

- 2つのページの間だけでループしている状態に他からのリンクがあり、そこに重みがどんどん吸収されてしまう場合
- ページがいくつかのグループにまとまり、グループ間にはリンクがない
- このような場合には、全ページにわたっての重みが均等に行き渡らない。
- →ある割合(15%)で、ランダムにページを探したと仮定
- →Aに全要素が一定値の行列を加算して計算

◆R＝(cA+(1-c)E)R

◆Eは全要素が1の行列、c=0.85としている

しかし、その2

- 流出すれだけで、流入する全くリンクのないページとは
- 勝手に他人にリンクを張っているだけの権威のないページ　→　無視する
- 機械的に張ったリンクの問題
  - PageRankのアイデアの基本は、人間がWebページを評価しリンクを張った状況、すなわち人間の判断を利用しようとしているので、機械が張ったリンクは悪影響を与える

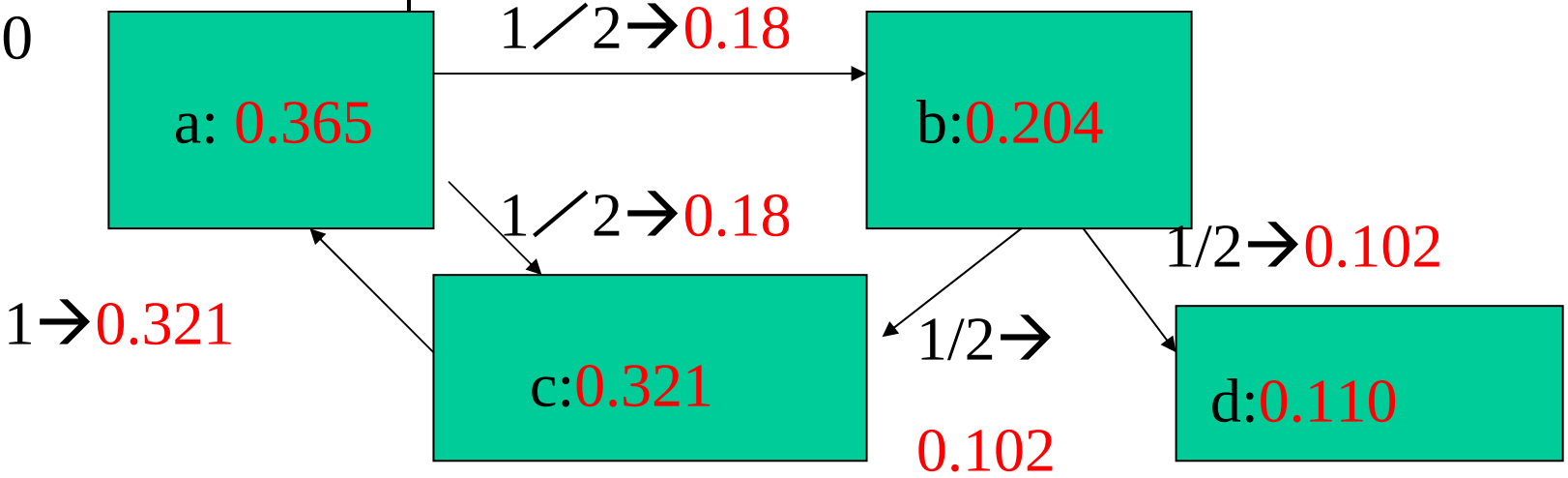
ページベクトルの計算

- R＝(cA+(1-c)E)R
  - Eは全要素が1の行列、c=0.85としてみた
- これはベクトルの固有値問題で Rは固有ベクトル。これを以下の繰り返し計算で解く。
  1. R(0)←適当な初期値
  2. R(i+1)←（cA+(1-c)E) R(i)
  3. D←||R(i)||ー||R(i+1)||
  4. R(i+1)←R(i+1)+dE
  5. δ ← ||R(i+1)ーR(i)||
  6. if δ > ε then goto 2
  - なお ||x||はベクトルの全要素2乗の和の√

計算例

- 簡単な例で繰り返し計算すると(ε =0.001)
- Rは $\sum_{i=1}^{4} r_i = 1$ と確率になるように正規化

$$\begin{array}{c} \\ a \\ b \\ c \\ d \end{array}\begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1/2 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & & & \end{bmatrix} \end{array} \Rightarrow cA+(1-c)E \Rightarrow 0.85A+0.15E$$



考察

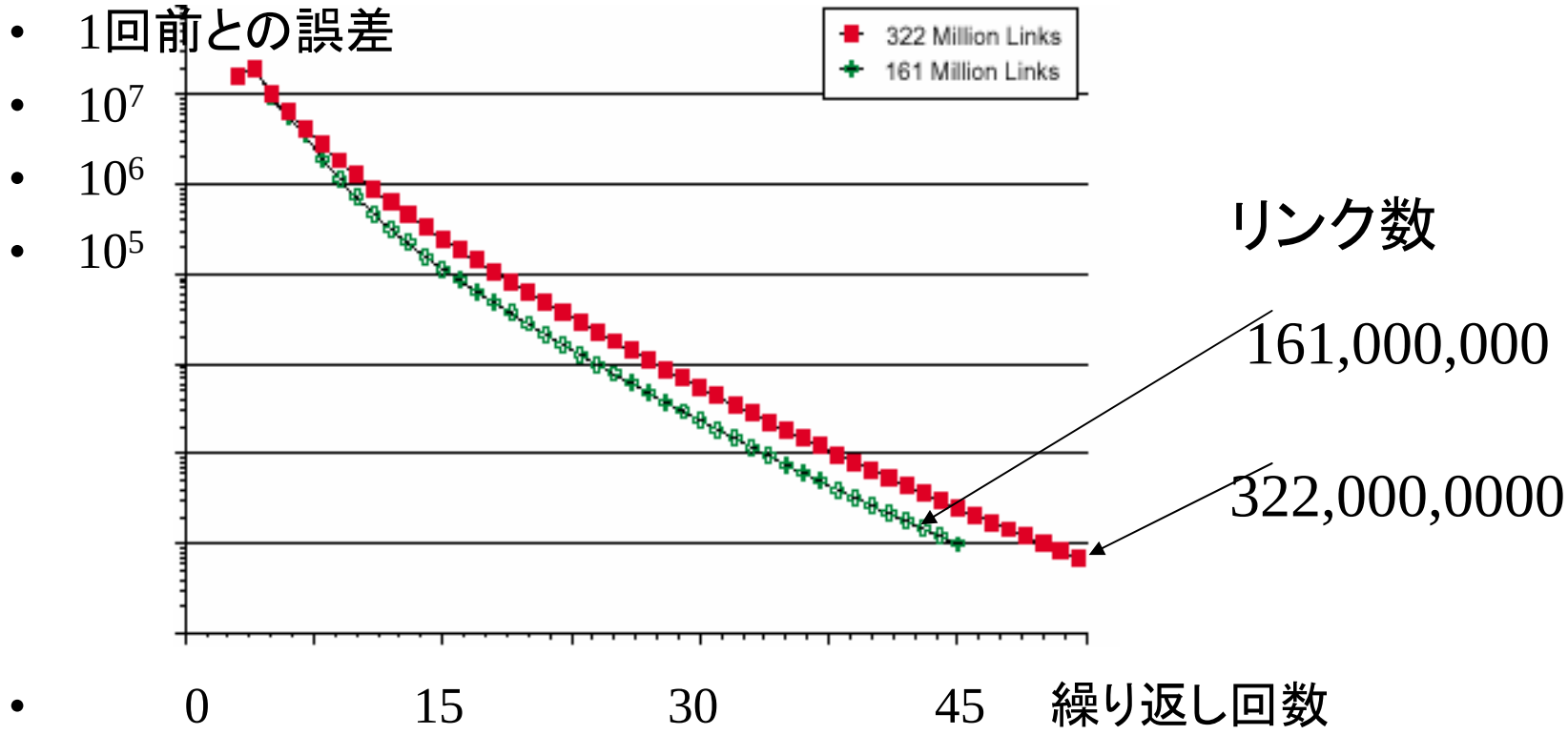
I. a,b,c,d ページともほぼ流れ込むリンクの重みに近い
II. リンク重みとページ重みのズレは、0.15の確率でランダムにページを移動する分
III. 良いページcからリンクされているaは重みが大きい
IV. 多くのページからリンクされているcも重みが大きい
V. あまりたいしたことのないページbだけからリンクされているdは重みが小さい
VI. 以上、ほぼ我々の意図したページの重みになっている。

数値計算の問題

- これまで述べてきたページ重みの繰り返し計算はうまく収束するのか、十分早いのか？
- Googleの10億ページでは100億のリンクがあるので、まっとうに線形代数の計算をしていたのでは無理。
  - 数値計算法ではかなり工夫をしているはず。企業秘密らしく公開されていない
  - Googleのマシンルームの様子からみると、
    - PCクラスタによる並列計算
  - 疎な行列を適当に分解すればまとまったページ群を独立に計算できるかも

収束の問題(以下の論文の数値例グラフ)

Lawrence Page, Sergey Brin, Rajeev Motwani, Terry Winograd, 'The PageRank Citation Ranking: Bringing Order to the Web', 1998



HITS algorithmのアイデア

- 探し当てたいWebページは必ずしも検索のキーワードを含まない
  - TOYOTAやHONDAのページに自動車製造業なんてキーワードは見当たらない
  - 一方、yahooなどの権威あるページはこういうキーワードをむ
  - キーワードではなくWebのリンク構造を利用 (PageRankと同じ意図)
- HITSは、ある話題に対するauthoritiesと多数のauthoritiesにリンクを張る hub との関係を使う方法

Focused Subgraph　－1

- ある話題に関するページ集合:Sが対象
- Sの持つべき性質は
  - 比較的小さい
  - 関連あるページを十分に含む
  - ほとんど全ての（あるいは多数の）authority page　を含む
- トピック:Qを表わすキーワードで検索エンジンから拾ってきたページ集合では、上記の条件は満たさない。

Focused Subgraph －2

- 例えば、検索結果の上位20ページを集めると
- javaというキーワードで検索すると、別のサイトのページ間には15links
- censorshipというキーワードで検索すると、別のサイトのページ間には28links
- 最大link数＝200×199=39800には遠く及ばない
- そこで、

Focused Subgraph　－3

- 比較的少数のタネ集合:Rからはじめる。
  - Rは質問:Qの検索結果の上位　t　ページ

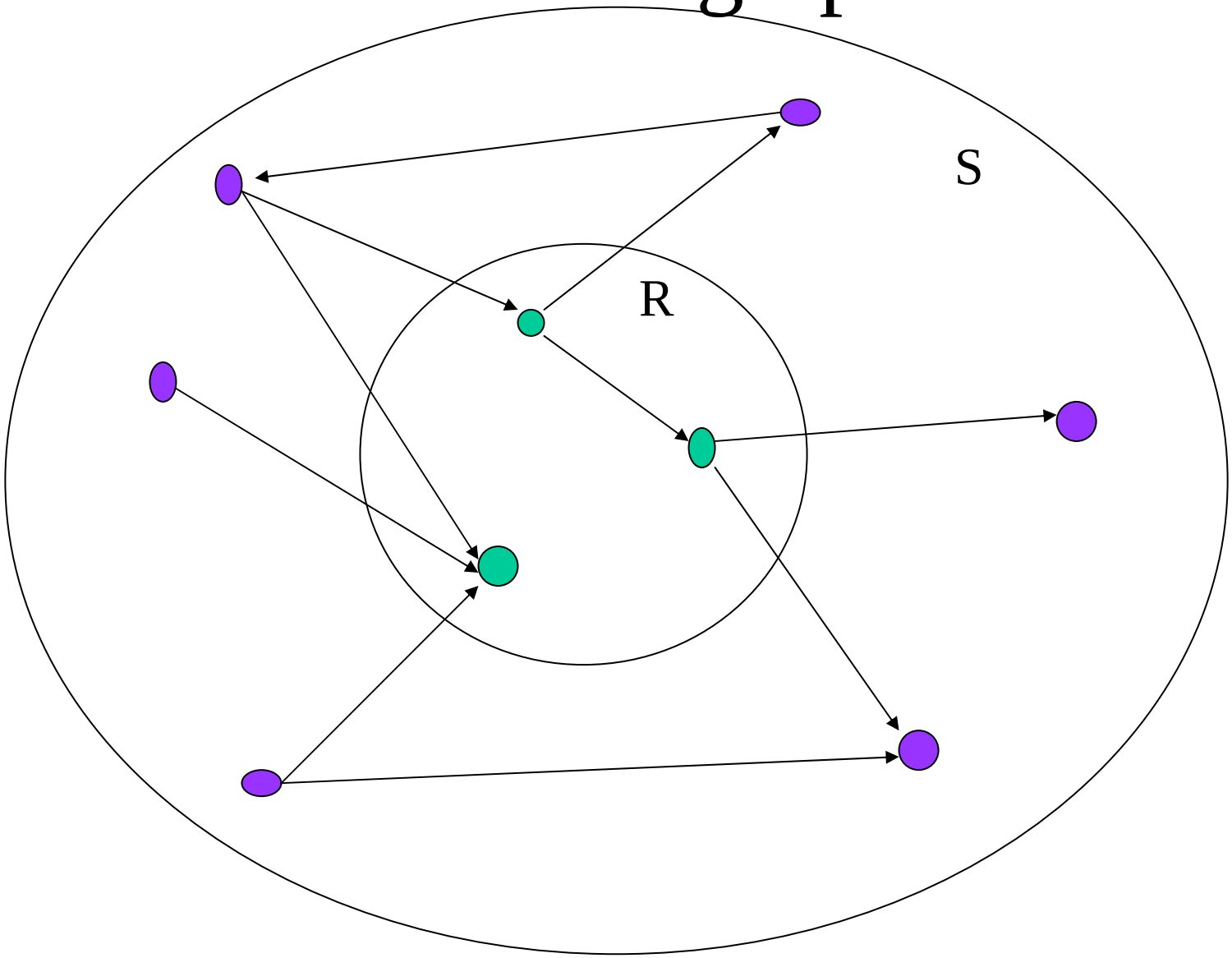
① S=R

② Rに属するページからリンクしている(out going link)ページを S に追加

③ Rに属するページにリンクされている(in coming link)ページのうち d ページをランダムに選んで S に追加

  - Kleinbergの論文では　t=200, d=50でSは1000 ~5000ページ

Focused Subgraph －4



Hubと　Authority　の計算

- S中には、トピック:Qに関係がないのに多くのリンクが入ってくるページがあり、
- 重要なページでないのに、やたらと他のページにリンクを張りまくっているページがある

- 真の authorioty と hub は相互に強化する
  - Good hub pages points to many good authority pages, good authority pages are pointed to by many hubs.
  - 同じdomain名の中でのリンクの張り合い無視（目次や自己参照などだから）

weights

- Authority weight of page: $p = y^{<p>}$
- Hub weight of page: $p = x^{<p>}$
- Link set: $E = \{(p,q) \mid page: p \text{ points to } page: q\}$
- Normalization $\sum_{p \in S} (x^{<p>})^2 = 1, \quad \sum_{p \in S} (y^{<p>})^2 = 1$

iteration

① : $x_0 = (1,1,.....,1) \in \mathbf{R}^n, \quad y_0 = (1,1,.....,1) \in \mathbf{R}^n$

② : 以下③から⑥をk回繰り返す(k＝20くらいで十分)

③ : $\textbf{for every } p \in S \quad x^{<p>} = \sum_{q:(q,p)\in E} y^{<q>}$

④ : $\textbf{for every } p \in S \quad y^{<p>} = \sum_{q:(q,p)\in E} x^{<q>}$

⑤ : $\{x^{<p>}\} \leftarrow \textbf{normalized} \quad \{x^{<p>}\}$

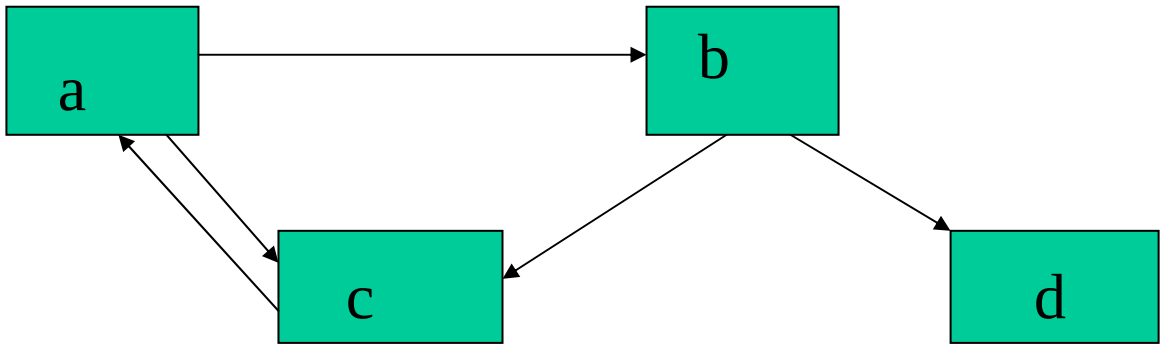
⑥ : $\{y^{<p>}\} \leftarrow \textbf{normalized} \quad \{y^{<p>}\}$

⑦ : 結果は

$\textbf{authority value} = x^{<p>}$

$\textbf{hub value} = y^{<p>}$

計算例



- Authority: a=3.46 × 10^{-6}, b=0.408
  c=0.816 , d=0.408
- Hub: a=0.707, b=0.707,
  c=5.98 × 10^{-6}, d=0

$$A = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

数学的再考－１

- 結局はページ間の接続行列に関連する固有値問題
- 接続行列

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \qquad a_{ij} = \begin{array}{l} 1 \text{ if page i points to page j} \\ 0 \quad \text{otherwise} \end{array}$$

- 繰り返し計算は

$$x_i = \begin{pmatrix} x_i^{<1>} \\ \vdots \\ x_i^{<n>} \end{pmatrix} \quad y_i = \begin{pmatrix} y_i^{<1>} \\ \vdots \\ y_i^{<n>} \end{pmatrix} \qquad x_{i+1} \leftarrow A^t y_i \quad \text{and} \quad y_{i+1} \leftarrow A x_{i+1}$$

数学的再考－2

◆したがって

$$x_k = (A^t A)^{k-1} A^t z, \qquad y_k = (AA^t)^k z, \qquad z = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

◆ゆえに

$A^t A$は$n \times n$の対称行列だから、Golub and Van Loanの線形代数の結果により、

$x^\infty = A^t A$の最大固有値に対応する固有ベクトル

$y^\infty = AA^t$の最大固有値に対応する固有ベクトル

実例
Jon M. KlienbergのAuthorities Sources in a Hyperlinked Environment, JACM 46-5 による

- (java) Authorities
- 0.328: http://www.gamelan.com
- 0.251: http://java.sun.com/
- 0.190: http://www.digitalfocus.com/digitalfocus/faq/howdoi.html
- 0.190: http://lightyear.ncsa.unic.edu/~srp/java/javabooks.html

パッセージ検索

- 文書の内容を特徴付けるのは文書全体よりはむしろ特定の部分
- ベクトル空間モデルを文書ではなく、文書の小さな部分、例えば段落、に適用。この小さな部分をパッセ―ジという。つまり、文書Dの代わりにパッセ―ジPkを使って、パッセ―ジ重みw'_kを計算し、ベクトル空間法を適用
- パッセ―ジの候補としては、

  1　固定長に分割したテキストの部分

  2　形式段落

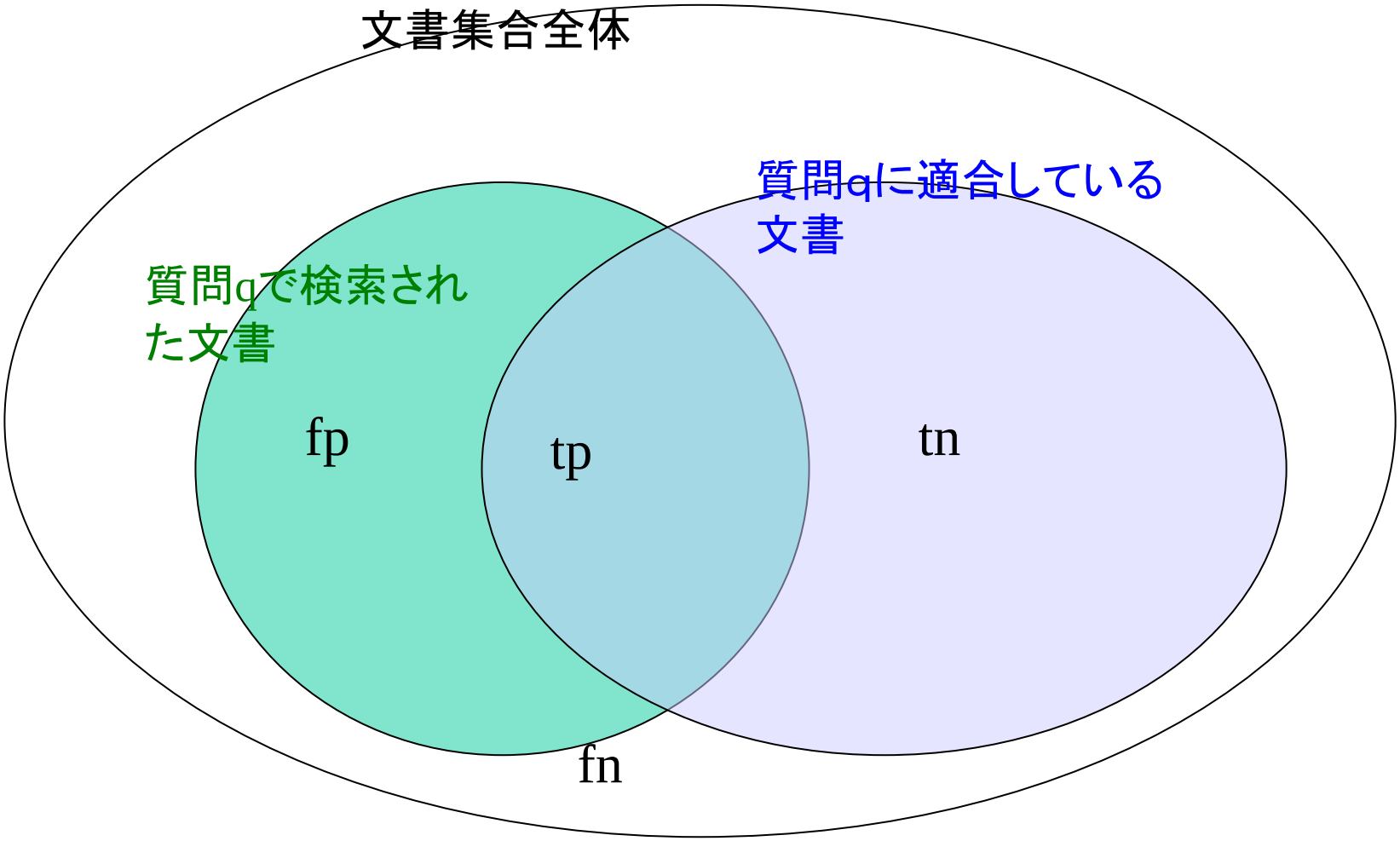
  3　形式的な節、章

情報検索システムの評価法

- 評価法
- 再現率、適合率
- F値など

一般的な検索結果の状態

- 質問qで結果の文書集合が得られた。しかし、結果の中には間違いもあるし、得られなかった文書の中にも正解がありうる。

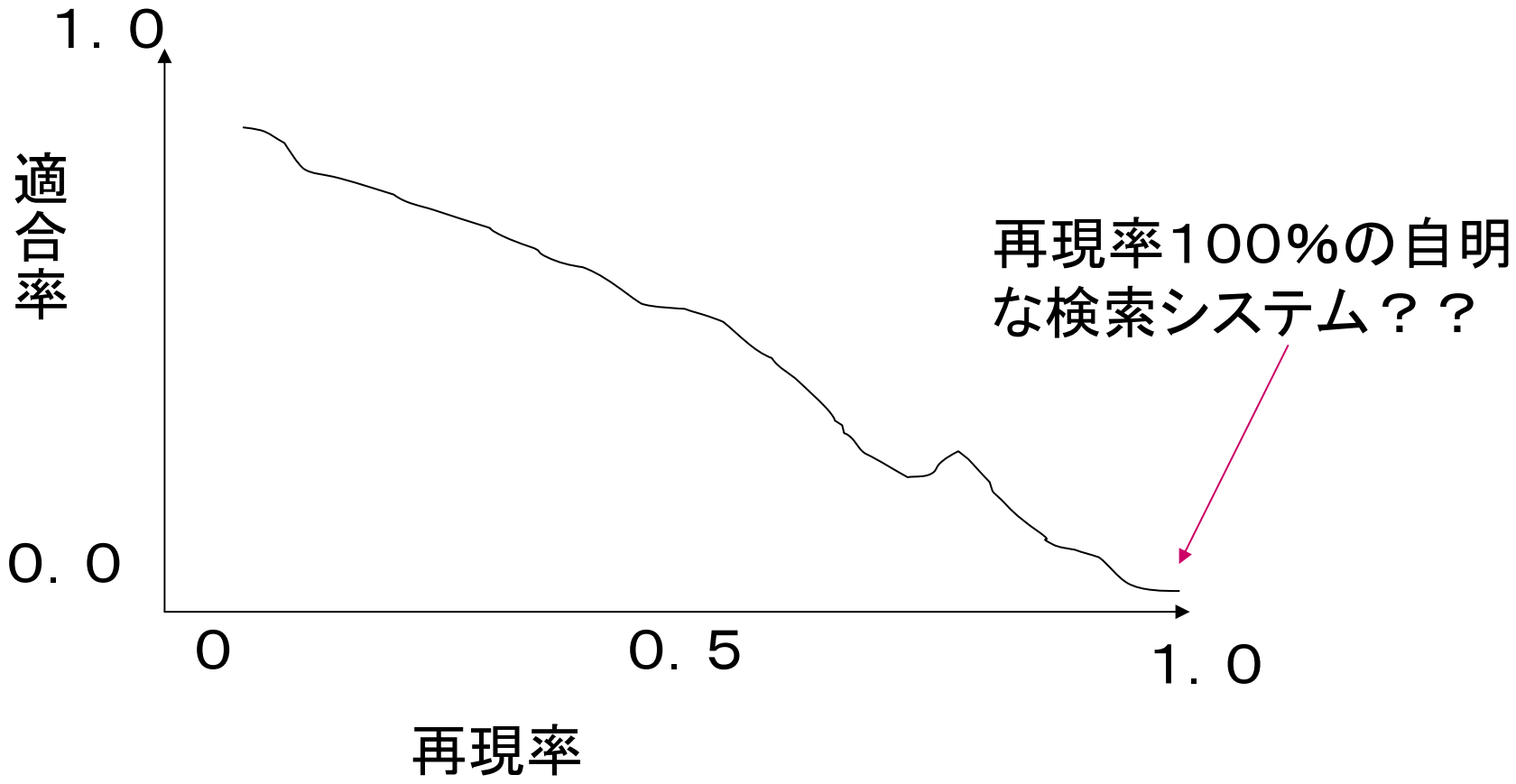


検索エンジンの性能評価

- 再現率 $$recall = \frac{tp}{tp + tn}$$
- 適合率あるいは精度 $$precision = \frac{tp}{tp + fp}$$
- フォールアウト $$fallout = \frac{tn}{tn + fp}$$
- 一般性 $$generarity = \frac{tp}{tp + tn + fp + fn}$$

再現率 vs 適合率

- よく使う評価の表現法



再現率　vs　適合率に関連した尺度

- Break even point　　再現率と適合率が一致する点
- 11点平均適合率　再現率＝0.0 , 0.1,　0.2, ….. 0.9, 1.0 の11点における適合率の平均値
- F値　　ただし、bは適合率が再現率よりどれだけ重視されているかを示すパラメタ―　b＝1がよく使われる。

$$F = \frac{(1 + \mathrm{b}^2) \times \mathrm{P} \times \mathrm{R}}{\mathrm{b}_2 \times \mathrm{P} + \mathrm{R}}$$

$$F = \frac{2PR}{P + R} = \frac{2}{\frac{1}{P} + \frac{1}{R}}$$

順位つき検索結果の評価

- ブーリアン検索では検索結果は全て同等
- ベクトル空間法やPageRank、HITSでは検索結果が質問に適合した順番に並ぶ。（表示も適合順）
- この場合の評価法について

Recall , Precision

- 質問qに適合する結果（以下、正解、という）の数: $|D_q|$
- 検索エンジンの順位つけられた結果: $(d_1 \ldots\ldots d_n)$
- d_i が質問qへの正解なら $r_i=1$、 そうでなければ $r_i=0$ とする。すると、
- 第k順位まで拾ったときの

$$\mathrm{Recall}(k) = \frac{1}{|D_q|} \sum_{1 \le i \le k} r_i$$

$$\mathrm{Precision}(k) = \frac{1}{k} \sum_{1 \le i \le k} r_i$$

平均適合率：average precision

$$\text{AveragePrecsion} = \frac{1}{|D_q|} \sum_{1 \le k \le N} r_k \times \text{precision}(k)$$

ただし、Nは正解が最後に現れた順位

- 例：

$$AvPrec = \frac{1}{2}\left(\frac{1}{1} + \frac{2}{4}\right) = 0.75$$

順位	正解か
1	○
2	
3	
4	○
5	
6	

平均逆順位：Mean Reciprocal Rank(MRR)

$$MRR = \frac{\sum_{n=1}^{N} (if \quad n-位の結果が正解 \quad \frac{1}{n} \quad else \quad 0)}{\sum_{n=1}^{N} \frac{1}{n}}$$

ただし、 Nは正解数

もし、正解がひとつも　現れなければ　　$MRR = 0$

- 例

$$MRR = \left(\frac{1}{1} + \frac{1}{4}\right) / \left(\frac{1}{1} + \frac{1}{2}\right) = 0.833$$

順位	正解か
1	○
2	
3	
4	○
5	
6	

テストコレクション

- (a) 文書集合、(b) 多数の質問、(c)各質問に対する適合文書の集合、を組にしたデータベースをテストコレクションと呼び、情報検索システムの性能評価において必須の資源である

- 正解集合を作ることは大規模テストコレクションでは困難
- Pooling method:、 同一の文書集合に対して、多数の検索エンジンで同じ質問を出し、上位N 個の検索結果を全て集める。N の値として、100 程度が多い。この結果に対してのみその適合性を人手で判断し、それを文書集合全体における適合した文書とする