【SIGIR2014勉強会】

Session 6: Hashing and Efficiency

担当：渡辺陽介（名古屋大学）

Session6の概要

1. Supervised Hashing with Latent Factor Models
   - P. Zhang (Shanghai Jiao Tong University), W. Zhang (Shanghai Jiao Tong University), W. Li (Nanjing University), M. Guo (Shanghai Jiao Tong University)
2. Preference Preserving Hashing for Efficient Recommendation
   - Z. Zhang (Purdue University), Q. Wang (Purdue University), L. Ruan (Purdue University), L. Si (Purdue University)
3. Load Balancing for Partition-based Similarity Search
   - X. Tang (University of California at Santa Barbara), M. Alabduljalil (UCSB), X. Jin, T. Yang (University of California at Santa Barbara)
4. Estimating Global Statistics for Unstructured P2P Search in the Presence of Adversarial Peers
   - S. Richardson (University College London), I. Cox (University College London)

[1本目]

Supervised Hashing with Latent Factor Models

- 背景
  - 大規模データに対する類似検索
    - 厳密解だけでなく，高速化のために近似解を返すことも時には必要
  - ハッシュを用いた類似検索
    - 類似したデータ同士には，近いハッシュ値が付かなければならない
    - 類似検索向けの既存のハッシュ手法
      - データに依存しない（トレーニングデータ不要）手法
        - Locality-Sensitive Hashing (LSH), SIKH,…
      - データに依存する（トレーニングデータあり）手法
        - 教師なし学習： Spectral Hashing (SH), PCAH, AGH, …
        - 教師あり学習： RBE, SPLH, MLH, KSH,…
- この論文の提案： Latent Factor Hashing (LFH)
  - データ依存，教師あり学習の一種
    - トレーニングデータ間のベクトル空間的な近さではなく，意味的な類似性の関係をラベルとして与える

[1本目]

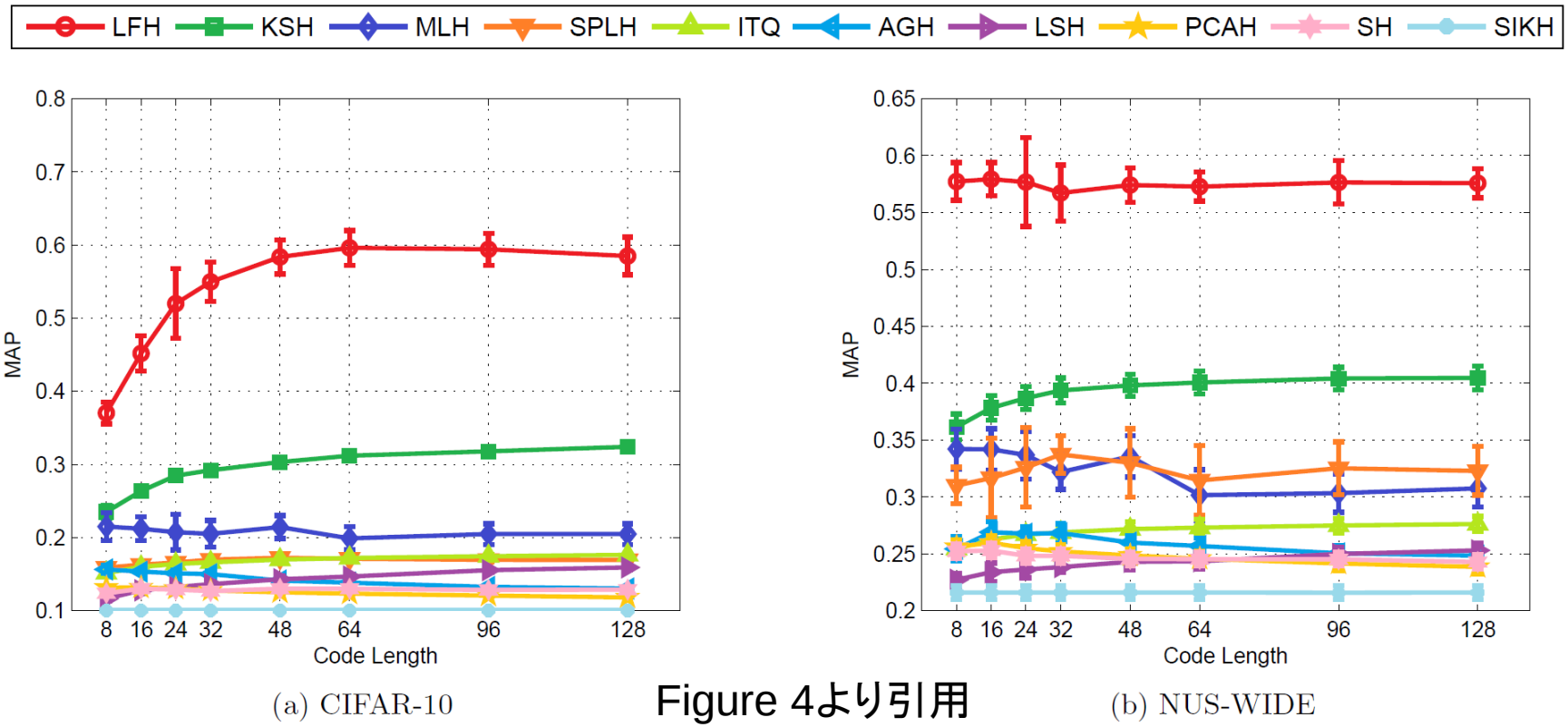
Latent Factor Hashing (LFH)

- 問題定義
  - トレーニングデータ
    - 特徴ベクトル：xi　(D次元のベクトル)
    - 類似度ラベル：sij（xiとxjが意味的に似ているなら1，それ以外0)
  - ハッシュ
    - ベクトルxiのバイナリコード：bi ($\in \{-1, 1\}^D$)　(Dビット)
- 学習
  - トレーニングデータの類似度ラベルとバイナリコードの関係に基づき，特徴ベクトルからバイナリコードへの変換行列を求める
    - 類似度ラベル sijが1のとき，ベクトルxiとxjのバイナリコード biとbjの距離（ハミング距離）は可能な限り小さくなければならない
    - 類似度ラベル sijが0のとき，ベクトルxiとxjのバイナリコード biとbjの距離（ハミング距離）は可能な限り大きくなければならない
  - 離散値の制約のままではNP-hardなので，実数制約に緩和して問題を解いている

[1本目]

評価実験

- データセット
  - CIFAR-10: 60,000枚の画像，NUS-WIDE: 269,648枚の画像
- 比較対象（提案手法はLFH)
  - データ依存，教師あり学習：KSH, MLH, SPLH,
  - データ依存,教師なし学習：SH, PCAH, ITQ, AGH
  - データ独立：LSH, SIKH



(a) CIFAR-10

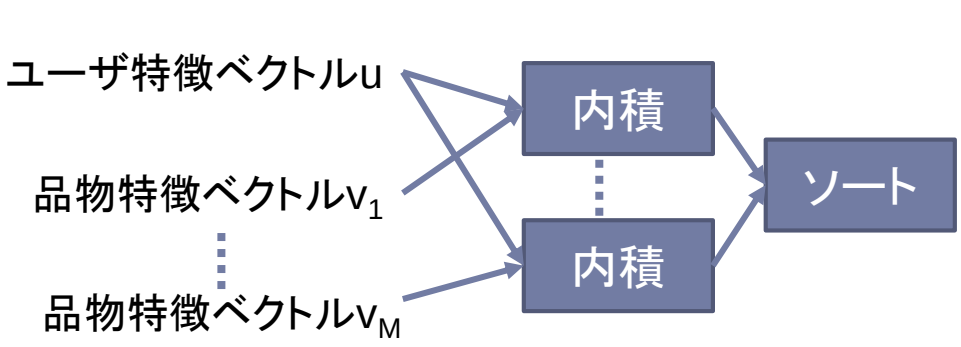
Figure 4より引用

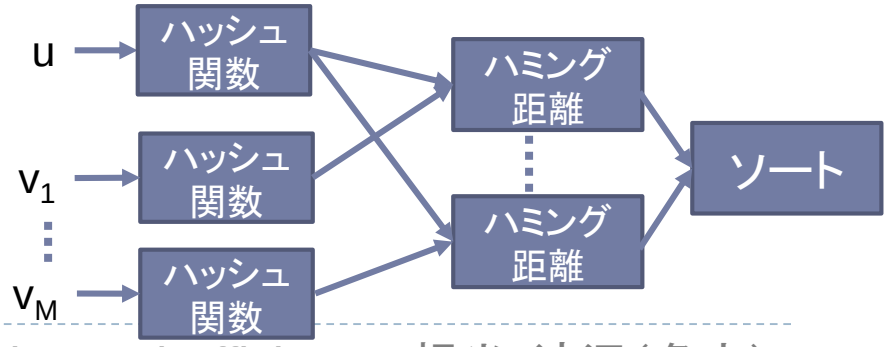
(b) NUS-WIDE

[2本目]

Preference Preserving Hashing for Efficient Recommendation

- 背景
  - ユーザの嗜好に合った品物を推薦する推薦システム
    - Matrix factorization (MF)法[14]：ユーザの嗜好を表すベクトルと，品物の特徴を表すベクトルの間の内積の計算が必要
- 論文の提案
  - ベクトル同士の内積計算の代わりに，ハッシュ値同士のハミング距離を用いた近似計算で高速化
    - 元の空間での内積の大小関係を引き継げるハッシュ手法が必要
  - Preference Preserving Hashing (PPH) (データ依存，教師なし学習)
    - Constant Feature Norm (CFN)
    - Magnitude and Phase Quantization (MPQ)





[2本目]

評価実験

- 実験データ: MovieLens(百万件, 一千万件), Netflix
- 比較対象: MedThresh, ITQ
- 提案手法: MPQとCFNの組み合わせが本命

Table 1: Performance of Hashing Code on MovieLens-1M

NDCG@10	MedThresh				ITQ				MPQ			
dim	16	24	32	40	16	24	32	40	16	24	32	40
CFN	0.5564	0.5582	0.5583	0.5609	0.5598	0.5629	0.5637	0.5659	**0.637***	**0.642***	**0.651***	**0.652***
L2	0.527	0.5229	0.5275	0.5284	0.5256	0.525	0.5291	0.5292	0.5828$_\circ$	0.5777$_\circ$	0.5809$_\circ$	0.5796$_\circ$
SumZero	0.5403	0.5387	0.5369	0.536	0.5402	0.5385	0.5355	0.5355	0.5782	0.5681	0.5613	0.5578

Table 2: Performance of Hashing Code on MovieLens-10M

NDCG@10	MedThresh				ITQ				MPQ			
dim	16	24	32	40	16	24	32	40	16	24	32	40
CFN	0.5643	0.5683	0.5724	0.5737	0.5651	0.571$_\circ$	0.574$_\circ$	0.577$_\circ$	**0.623***	**0.628***	**0.638***	**0.641***
L2	0.549	0.5529	0.5554	0.5559	0.5527	0.5557	0.5549	0.5533	0.5356	0.5426	0.5436	0.5337
SumZero	0.542	0.5395	0.5391	0.532	0.5324	0.5398	0.5391	0.5338	0.5789$_\circ$	0.5623	0.5536	0.5534

Table 3: Performance of Hashing Code on Netflix

NDCG@10	MedThresh				ITQ				MPQ			
dim	16	24	32	40	16	24	32	40	16	24	32	40
CFN	0.5554	0.5676	0.5673	0.5687	0.5557	0.5647	0.5682	0.571$_\circ$	**0.622***	**0.629***	**0.641***	**0.642***
L2	0.5589	0.56	0.5594	0.554	0.5435	0.5461	0.5497	0.5471	0.5845$_\circ$	0.571$_\circ$	0.5811$_\circ$	0.5705
SumZero	0.5581	0.5549	0.554	0.5443	0.5591	0.555	0.5558	0.5457	0.5605	0.5637	0.577	0.5576

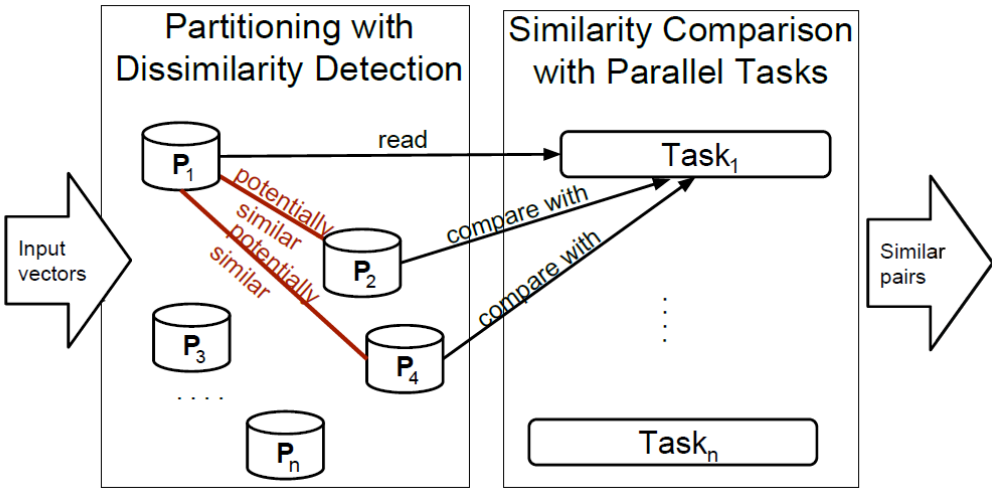
Table1, 2, 3 を引用

[3本目]

Load Balancing for Partition-based Similarity Search

- 背景
  - All Pairs Similarity Search (APSS)
    - 全データのペアのうち，類似度が高いものを探す問題
    - とにかく計算コストが高い
      - 不要な比較計算を減らしたい．分散処理で高速化したい（負荷分散必要）
- アプローチ：データ集合に対する2段階の比較処理
  1. 類似度の上限を大雑把に見積り，データ集合をパーティションへ分割
     - 同じパーティションのデータは同じ計算ノードへの配置される
  2. 各パーティションを分散配置した状態での類似度計算
     - 1の上限値に基づいて，類似度の低そうなデータしかないパーティション間の類似度計算を省く

Figure1より引用

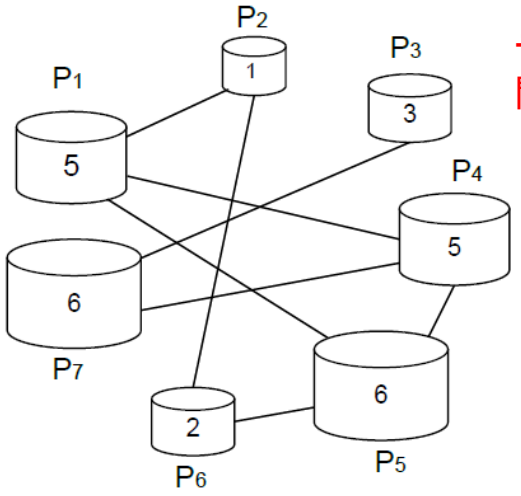


[3本目]

負荷分散

- 各ノードで行われる類似度計算
  - パーティション内のデータのペアの類似度計算（ローカル）
  - パーティション間のデータのペアの類似度計算（データ転送→計算）
    - Similarity Graphに基づいて，類似度の低そうなデータしかないパーティションのデータは類似度計算の対象にしない
    - 計算が必要なパーティションの場合，「データを受信し自分で計算する」か「データを送信し相手に計算してもらう」が選べる
      - Comparison Graphの構築→比較コスト・IOコスト見積もり（負荷分散）

Similarity Graph



データ送受信
関係の導出

Comparison Graph

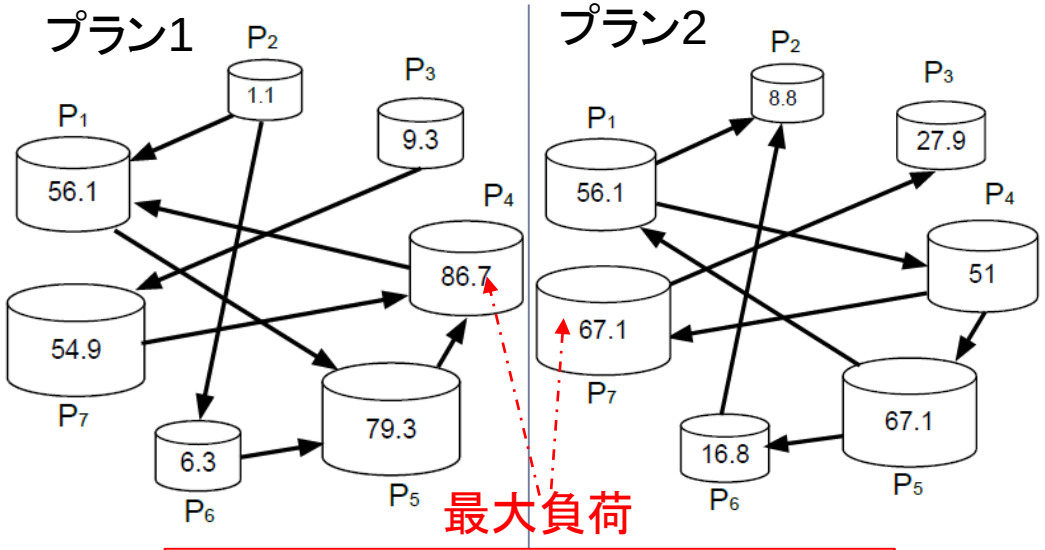


Figure3より引用

頂点：パーティション（数字はデータサイズ）
辺：類似度の高いパーティションの関係

頂点：パーティション（数字はコスト）
辺：データの送信の関係

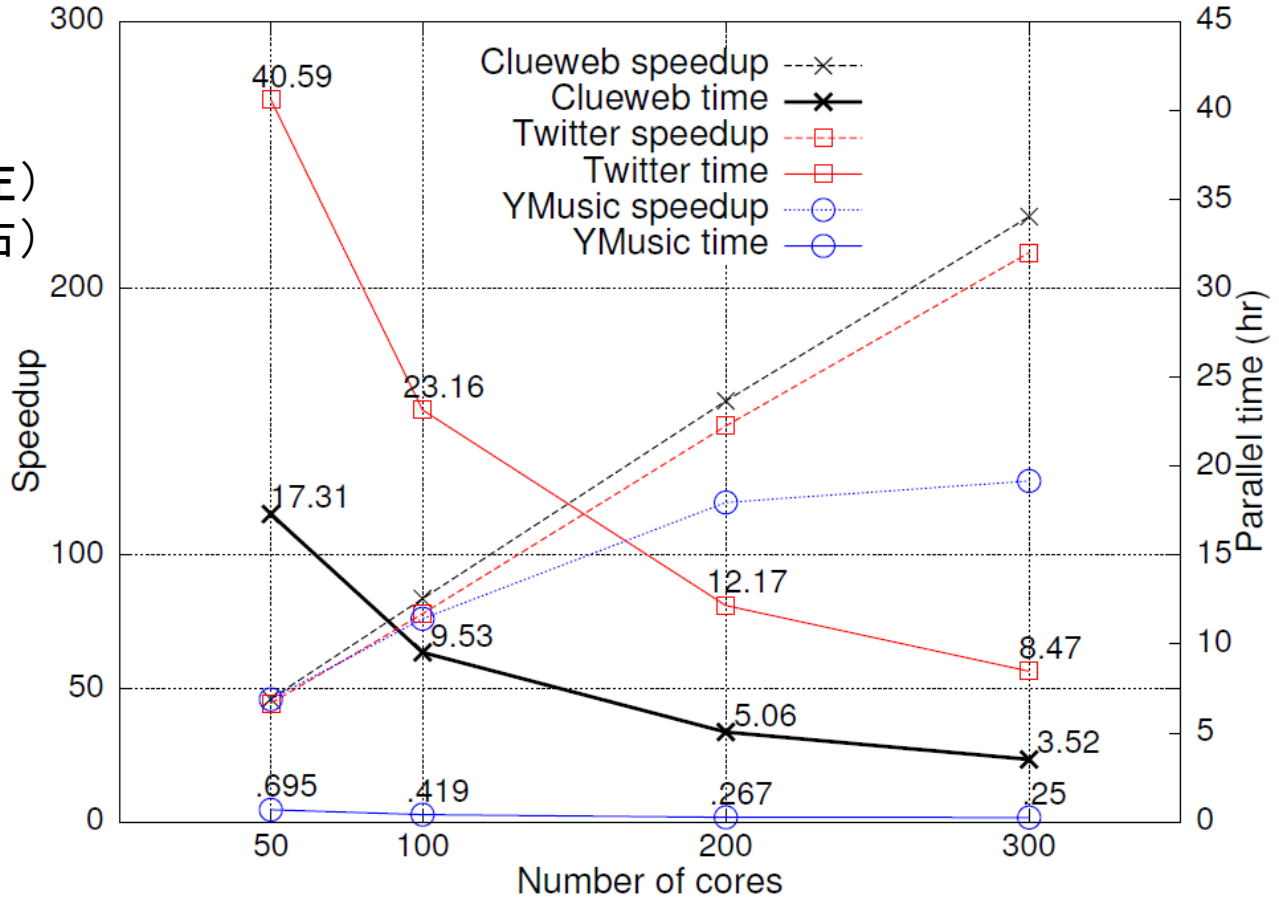
[3本目]

評価実験

Table2
データセットと
単独実行時の
処理時間

Dataset	Twitter		ClueWeb		YMusic
Size	4M	100M	1M	40M	625K
AMD	45	45,157*	50	79,845*	31.95
Intel	26.7	25,438*	29.3	46,946*	17.8
AMD/df-limit	1.27	797*	4.55	7,286*	6.23

Figure9
コア数（横軸）と
速度向上（縦軸左）
処理時間（縦軸右）



[4本目] Estimating Global Statistics for Unstructured P2P Search in the Presence of Adversarial Peers

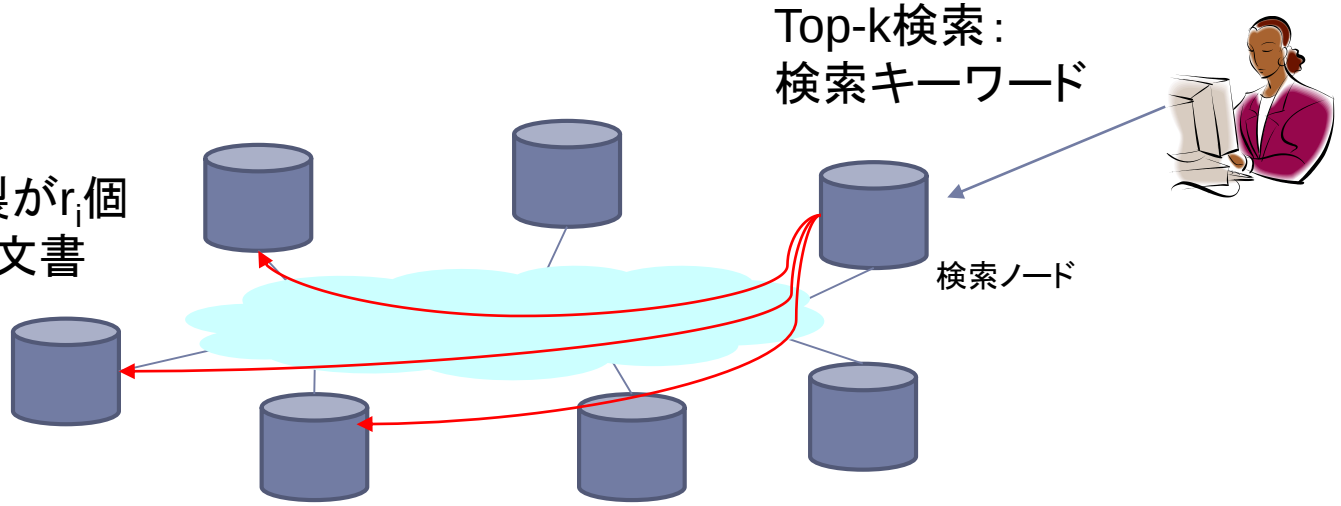
- 構造化されていないP2Pネットワークでの文書検索
  - 文書のスコア計算には，各単語の出現頻度等の情報が必要
    - IDF(Inverse Document Frequency)の計算など
  - しかし，P2Pには全体の統計量を管理している場所がない
- P2Pネットワーク全体で単語の統計情報を推測しながら，文書のスコア計算を見積もる
  - 著者の一人の過去の論文の拡張
    - Probably Approximately Correct Search [10]
- 論文の貢献
  - [10]を拡張し，2つのランキング手法での推測
    - BM25，language model with Dirichlet smoothing
  - 敵対的なノードがいた場合の分析
    - 特定の文書のランクを下げる，上げる，検索結果全体の精度を下げる

[4本目]

PACでの問合せフレームワーク

想定:
- n台のノード
- m種類の文書
- 文書d_iに対して複製がr_i個
- 各ノードにはρ個の文書

Top-k検索:
検索キーワード



1. 検索ノードは，z台のノードへクエリ（キーワード集合）を送信
2. クエリを受け取ったノードでは，ローカルのデータを使ってtop-k'検索を実行
   - ローカルのtop-k'検索には昔収集した統計値を使用
3. 検索ノードは，z台のノードからtop-k'検索結果と単語の統計情報を取得
   - 統計値の推定，スコアの計算，最終的なtop-kの結果を生成

単語tが文書に出現する確率(BM25)

$$\hat{P}_{doc}(t) = \frac{\sum_{u \in Z} DF(t, L_u)}{\sum_{u \in Z} |L_u|}$$

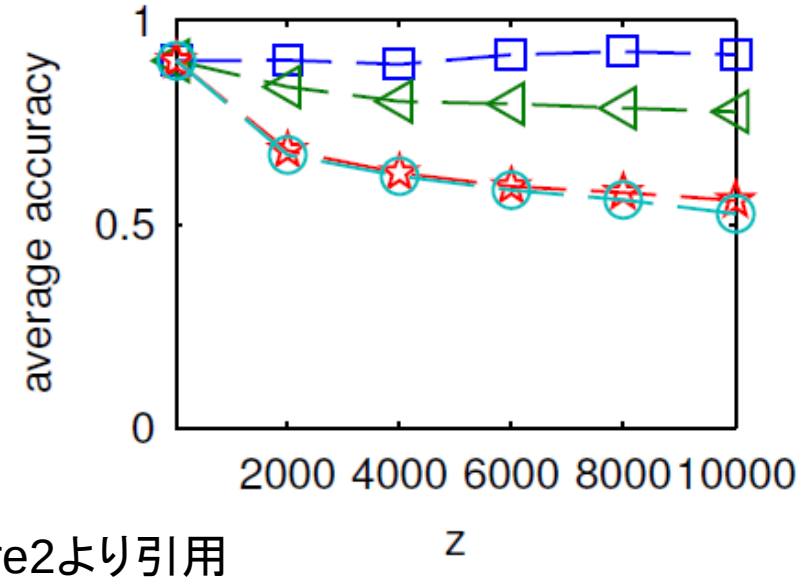
平均文書長の推定(BM25)

$$AV\hat{G}DL = \frac{\sum_{u \in Z} \sum_{d \in L_u} DL(d)}{\sum_{u \in Z} |L_u|}$$

[4本目]

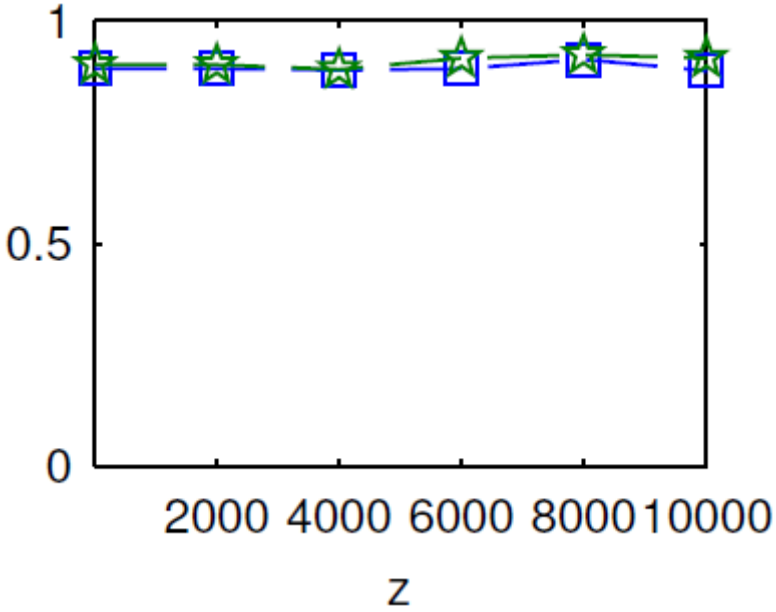
シミュレーション実験（10,000ノード，1,692,096文書）

figure1より引用



- z台のノードの情報での $P_{doc}(t)$の推定値
- z台のノードの情報での AVGDLの推定値
- ノード単体での$P_{doc}(t)$の推定値
- ノード単体の情報でのAVGDLの推定値

figure2より引用



- z台のノードに送るクエリのtop-k'を k'=ρにした場合の精度
- z台のノードに送るクエリのtop-k'を k'=10にした場合の精度

（元のクエリはtop-10検索）