

【VLDB2011勉強会】

Session 24: Searching and Ranking

担当: 大島裕明(京都大学)

Searching and Ranking

[1] Fast Incremental and Personalized PageRank

- ▶ B. Bahmani (Stanford Univ.), A. Chowdhury (Twitter Inc.), A. Goel (Stanford Univ., Twitter Inc.)

[2] Efficient Diversification of Web Search Results

- ▶ G. Capannini, F.M. Nardini, R. Perego, F. Silvestri (ISTI-CNR)

[3] Keyword Search on Form Results

- ▶ A. Ramesh (Stanford Univ.), S. Sudarshan (IIT Bombay), P. Joshi (IIT Bombay)

Fast Incremental and Personalized PageRank

▶ 研究の目的

- ▶ PageRankやSALSAの計算の高速化

▶ 特徴

- ▶ モンテカルロ法
- ▶ 全ページでスコアを更新する手法よりも高速

▶ PageRank

- ▶ Webにおいてランダムウォークするユーザの存在確率
- ▶ 確率 ϵ でランダムジャンプが発生

▶ SALSA

- ▶ HITSとPageRankの中間
- ▶ 二部グラフ上で前進／後退を繰り返して計算するPageRank
 - ▶ 前進時に確率 ϵ でランダムジャンプが発生

Fast Incremental and Personalized PageRank

従来手法

- 全ページのPageRank値を保持
 - 初期値は $1/n$
- ランダムウォークとランダムジャンプでPageRank値を更新
- 収束するまで上記を繰り返す

モンテカルロ法による提案手法

- 各ノードにユーザを配置する
- いずれかのページにユーザを動かす
 - 平均長 $1/\epsilon$ まで動かす (Short Random Walk)
- 集約する
 - $\ln(n/\epsilon)$ 回とか1回とかで良い結果が得られる

Efficient Diversification of Web Search Results

▶ 研究の目的

- ▶ Web検索結果になるべく多様な情報を掲載する
- ▶ 従来手法よりも高速な手法の提案

▶ Search Results Diversification

- ▶ 検索クエリ「Apple」



▶ 検索クエリ「Harry Potter」

- ▶ 本？映画？キャラクター？

Efficient Diversification of Web Search Results

- ▶ 解くべき問題はDiversify(k)
 - ▶ 以下の数式を最大化する検索結果文書集合Sを求めること

$$P(S|q) = \sum_c P(c|q) \left(1 - \prod_{d \in S} (1 - V(d|q, c)) \right)$$

クエリqにおけるカテゴリc
の出現確率

少なくとも1つの文書 $d \in S$ がカテゴリc
に関係しているということ

- ▶ 既存手法ではこれをそのまま解こうとしている
 - ▶ NP困難
 - ▶ $O(nk)$ の計算量が必要
 - ▶ カテゴリ数がkよりも小さい場合でも全てのカテゴリが網羅される保証はない
 - ▶ qの主たるカテゴリが「くだらない」ものだった場合、検索結果は「くだらない」ものであふれかえることになる

Efficient Diversification of Web Search Results

▶ 提案手法のキモ

▶ クエリログからの「Specialization」のマイニング

▶ ちょっと寄り道して「Query Reformulation」のお話

▶ Error Correction

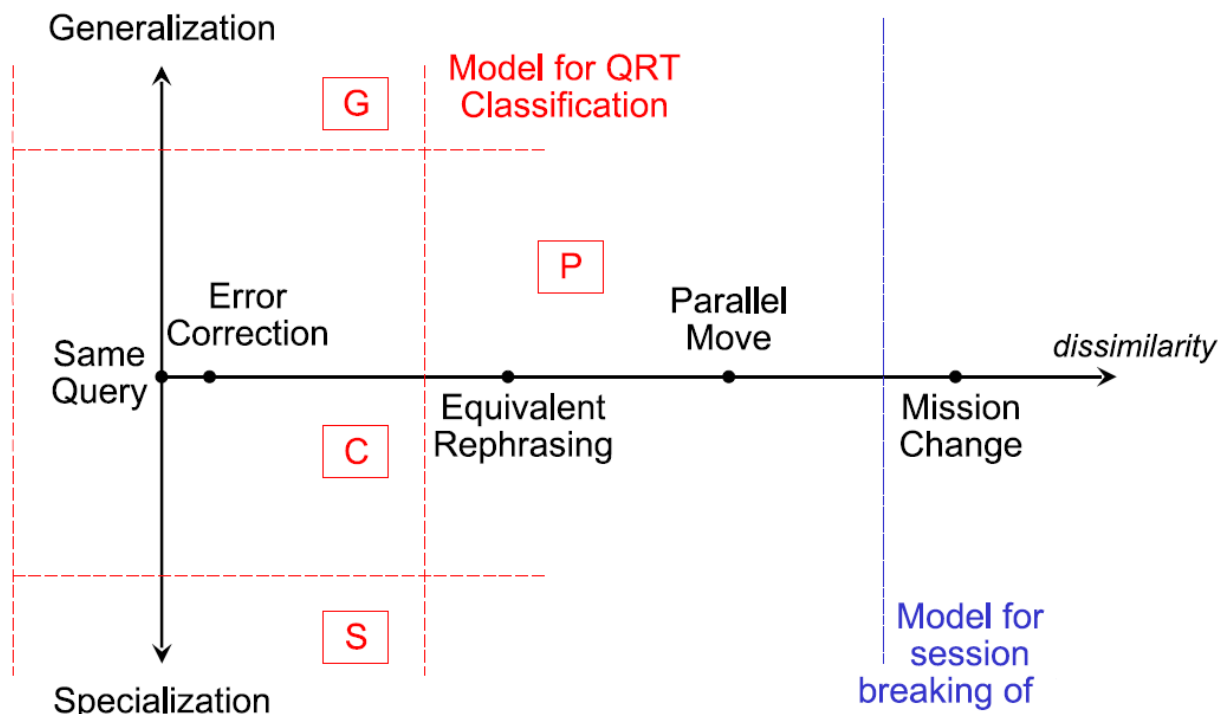
▶ Generalization

▶ Specialization

▶ Equivalent
Rephrasing

▶ Parallel Move

▶ Mission Change



From "Dango" to "Japanese Cakes" :Query Reformulation Models and Patterns, Boldi et al., WI-IAT 2009.

Efficient Diversification of Web Search Results

▶ 提案手法

- ▶ クエリログから「Specialization」をマイニング(手法問わず)
- ▶ 各「Specialization」の発生する頻度(確率) = Utility
 - ▶ ジョブズのアップル: 5/12
 - ▶ レコード会社のアップル: 1/3
 - ▶ 果物のアップル: 1/4
- ▶ 頻度に応じて検索結果の内容をDiversifyする

▶ 結果

- ▶ 計算量が理論上 $O(nk)$ から $O(n \log_2 k)$ に減少
- ▶ TREC 2009 WebトラックのClueWeb09のうちのClueWeb-Bを利用して評価
 - ▶ α -nDCGとかIA-Pで同じかより良いか

Keyword Search on Form Results

▶ 研究の目的

- ▶ スキーマを知らないDBでのキーワード検索
- ▶ 実世界志向

こんなのやだね

Rank: 1 Score: 0.29653063 (es=0.33333334 , ns=0.185709)

```
•Table: role
personid=53205, movieid=42715, character=Cameo appearance (steerage dancer),
•Table: person
id=53205, name=Cameron James, sex=M,
•Table: movie
id=42715, title=Titanic (1997), year=1997, rating=7,
```

▶ 問題

▶ 与えられるもの

- ▶ 検索フォーム集合 $F = \{f_1, f_2, \dots, f_n\}$
 - F_i のパラメータ(列)集合は P_i
- ▶ キーワード集合 $K = \{k_1, k_2, \dots, k_m\}$

▶ 出力

- ▶ (f_j, p_j) のようなペア (p_j はパラメータの値)

Keyword Search on Form Results

▶ 課題

- ▶ 与えられたキーワードが属性か属性値かが不明
 - ▶ 「田島敬史 授業」→「田島敬史先生が担当の授業を知りたい」
 - ▶ 「田島敬史 DB」→「田島敬史先生が担当のDBの授業を知りたい」

▶ 第一段階: キーワード独立なインバースクエリの生成

- ▶ k個のパラメータが与えられた時のクエリ

- ▶ $Q = \Pi_{A_1, \dots, A_n} (\sigma_{B_1=\$B_1 \wedge B_2=\$B_2 \wedge \dots \wedge B_k=\$B_k} (r_1 \times r_2 \times \dots \times r_n))$

- ▶ キーワード独立なインバースクエリ(KIIQ)

- ▶ $KIIQ = \Pi_{B_1, B_2, \dots, B_k, A_1, \dots, A_n} (\sigma_p (r_1 \times r_2 \times \dots \times r_n))$

▶ 第二段階: KIIQにキーワードを組み込む

- ▶ キーワードが1つの場合

- ▶ $IQ = \Pi_{B_1, B_2, \dots, B_k, A_1, \dots, A_n} (\sigma_{Contains}((B_1, B_2, \dots, B_k, A_1, \dots, A_n), K_1) (KIIQ))$

- ▶ キーワードが複数の場合

- ▶ 各キーワードのIQのインターセクション

Keyword Search on Form Results

▶ 例

▶ 普通のクエリ

$$\text{▶ } Q = \Pi_{name}(\sigma_{Id=\$Id}(prof))$$

▶ キーワード独立なインバースクエリ

$$\text{▶ } KIIQ = \Pi_{name,Id}(prof)$$

▶ キーワードが「John」の場合のインバースクエリ

$$\text{▶ } IQ = \Pi_{Id}(\sigma_{contains((name,Id),"John")}(prof))$$

▶ キーワードが「John」と「Smith」の場合のインバースクエリ

$$\text{▶ } IQ = \Pi_{Id}(\sigma_{contains((name,Id),"John")}(prof) \\ \cap \Pi_{Id}(\sigma_{contains((name,Id),"Smith")}(prof)))$$

▶ 複数のRelationにまたがる場合

$$\text{▶ } \Pi_{Id}(\sigma_{contains((Id,name),"John") \wedge contains((teaches,ctitle),"John")}(prof \bowtie teaches))$$

$$\text{▶ } \Pi_{Id}(\sigma_{contains((name,Id),"John")}(prof \bowtie teaches) \\ \cup \Pi_{Id}(\sigma_{contains((teaches,cities),"Smith")}(prof \bowtie teaches)))$$