Suggesting Topic-Based Query Terms as You Type

Ju Fan, Hao Wu, Guoliang Li, Lizhu Zhou

Tsinghua University, Beijing, China

Mar. 2010
Outline

1. Motivation

2. Model for Ranking Terms
   - Basic Idea
   - Modeling

3. Progressive Ranking Algorithm
   - Trie Index
   - top-\(k\) Ranking Algorithm

4. Experiments

5. Conclusion
Outline

1. Motivation

2. Model for Ranking Terms
   - Basic Idea
   - Modeling

3. Progressive Ranking Algorithm
   - Trie Index
   - top-\(k\) Ranking Algorithm

4. Experiments

5. Conclusion
Query Performance

Low-quality queries → Low performance of search.

Example 1 (Complete Search)

zoomed in on 94578 documents


Example 2 (Complete Search)

No hits

query prepare

zoomed in on 81 documents

query formulation
Query Term Suggestion

**Goal:**

formulate high-quality queries by presenting a list of candidate terms for users to choose

---

**Example:****

![Query Term Suggestion Example](image)

This technique improves retrieval effectiveness significantly.
How to suggest terms?

Existing Methods
- Document-based, e.g., Co-occurrence
- Term-based, e.g., thesauruses
- Concept-based, e.g., Wikipedia, query logs

Limitation: neglecting the topic information
- A document can be seen as a mixture of topics.
- A user, who issues a particular query, is generally interested in a single or a small number of topics.

Why not suggest topic-based query terms?
How to suggest terms?

Existing Methods
- Document-based, e.g., Co-occurrence
- Term-based, e.g., thesauruses
- Concept-based, e.g., Wikipedia, query logs

Limitation: neglecting the topic information
- A document can be seen as a mixture of topics.
- A user, who issues a particular query, is generally interested in a single or a small number of topics.

Why not suggest topic-based query terms?
### Our method

**DBLife data set**

<table>
<thead>
<tr>
<th>Queries</th>
<th>Our Approach</th>
<th>Complete Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>semantic o</td>
<td>ontology</td>
<td>org</td>
</tr>
<tr>
<td></td>
<td>ontologies</td>
<td>open</td>
</tr>
<tr>
<td></td>
<td>org</td>
<td>order</td>
</tr>
<tr>
<td></td>
<td>overview</td>
<td>object</td>
</tr>
<tr>
<td></td>
<td>object</td>
<td>oriented</td>
</tr>
</tbody>
</table>

---

**Advantages**

- Topic-based query terms
- Autocompletion
Our method

<table>
<thead>
<tr>
<th>DBLife data set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queries</td>
</tr>
<tr>
<td>--------:</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>semantic</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Advantages

- **Topic-based query terms**
- **Autocompletion**
Challenges and Solutions

- The mechanism of ranking terms
  - The ranking model with topic information is more complicated.
  - We make use of the *Latent Dirichlet Allocation* model to calculate the topical coherence of terms in absence of any external information.

- Suggesting terms in real time
  - Ranking query terms as well as their corresponding documents.
  - We use an index structure, *trie*, to efficiently access all possible terms with a particular prefix.
  - We design a progressive algorithm to find the top-$k$ terms based on our proposed model.
Outline

1. Motivation

2. Model for Ranking Terms
   - Basic Idea
   - Modeling

3. Progressive Ranking Algorithm
   - Trie Index
   - top-k Ranking Algorithm

4. Experiments

5. Conclusion
Intuition

If a term is more topically coherent with the context and is more likely to be sampled from the retrieved documents, it would obtain a higher score.

Two factors

1. **The topical coherence between two terms.** An LDA model is utilized to learn the term distribution over each topic from the underlying documents.

2. **The likelihood that a term is sampled from a document.** The term distribution over a set of documents is learnt by using a language model.
The LDA Model

LDA is one of the latest topic models that can learn hidden topics from text documents in an unsupervised manner, and represent each term as a distribution of topics.

The Conditional Probabilities of A Terms under Topics

![Graph showing conditional probabilities of terms under topics.](image)
Model of Ranking Terms

**Ranking Model**

\[ P(c \mid s) = \lambda \sum_{t_i \in T} P(c \mid t_i)P(t_i \mid s) + (1 - \lambda) \sum_{d_j \in D} P(c \mid d_j)P(d_j \mid s) \]

1. \(c\) is a term to be suggested.
2. \(s\) is the previously issued keywords, context.
3. \(\sum_{t_i \in T} P(c \mid t_i)P(t_i \mid s)\), The relationship between \(c\) and \(s\) in terms of topics
4. \(\sum_{d_j \in D} P(c \mid d_j)P(d_j \mid s)\), the likelihood of sampling \(c\) from each document \(d_j\) retrieved by \(s\)
5. \(\lambda\) is a tuning parameter.
<table>
<thead>
<tr>
<th>ID</th>
<th>Sample Documents in DBLP dataset (only paper titles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Benchmarking Decision Models for Database Management Systems</td>
</tr>
<tr>
<td>1</td>
<td>A Dynamic Data Model for a Video Database Management System</td>
</tr>
<tr>
<td>2</td>
<td>Using Economic Models to Allocate Resources in Database Management Systems</td>
</tr>
<tr>
<td>3</td>
<td>Some Models of a Distributed Database Management System with Data Replication</td>
</tr>
<tr>
<td>4</td>
<td>A General Model for Event Specification in Active Database Management Systems</td>
</tr>
<tr>
<td>5</td>
<td>Mining Protein Database using Machine Learning Techniques</td>
</tr>
<tr>
<td>6</td>
<td>Mining Sequential Patterns across Multiple Sequence Databases</td>
</tr>
<tr>
<td>7</td>
<td>Data Mining: Machine Learning, Statistics, and Databases</td>
</tr>
<tr>
<td>8</td>
<td>Declarative Database Management in SQLServer</td>
</tr>
<tr>
<td>9</td>
<td>Machine Learning and Data Mining</td>
</tr>
</tbody>
</table>
### A Running Example (2/2)

<table>
<thead>
<tr>
<th>Term</th>
<th>Topic</th>
<th>P(term)</th>
<th>DF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>t₀</td>
<td>t₁</td>
<td>t₂</td>
</tr>
<tr>
<td>database</td>
<td>0.050</td>
<td>0.010</td>
<td>0.008</td>
</tr>
<tr>
<td>data</td>
<td>0.012</td>
<td>0.013</td>
<td>0.047</td>
</tr>
<tr>
<td>decision</td>
<td>0.004</td>
<td>0.015</td>
<td>0.038</td>
</tr>
<tr>
<td>model</td>
<td>0.060</td>
<td>0.020</td>
<td>0.001</td>
</tr>
<tr>
<td>management</td>
<td>0.003</td>
<td>0.004</td>
<td>0.040</td>
</tr>
<tr>
<td>mining</td>
<td>0.020</td>
<td>0.030</td>
<td>0.010</td>
</tr>
<tr>
<td>machine</td>
<td>0.010</td>
<td>0.040</td>
<td>0.005</td>
</tr>
<tr>
<td>system</td>
<td>0.040</td>
<td>0.020</td>
<td>0.005</td>
</tr>
<tr>
<td>statistic</td>
<td>0.002</td>
<td>0.036</td>
<td>0.015</td>
</tr>
<tr>
<td>sequence</td>
<td>0.001</td>
<td>0.045</td>
<td>0.018</td>
</tr>
</tbody>
</table>
### Computation of $P(model|database)$

| $P(c|t_0)$ | $P(c|d_0)P(d_0|s) = 1/54$ |
|------------|-----------------------------|
| 0.06       |                             |

| $P(t_0|s)$ | $P(c|d_1) \cdot P(d_1|s) = 1/63$ |
|------------|----------------------------------|
| $P(s|t_0) \cdot P(t_0) / P(s)$ | 0.7246 |

| $P(c|t_1)$ | $P(c|d_2) \cdot P(d_2|s) = 1/72$ |
|------------|----------------------------------|
| 0.02       |                                  |

| $P(t_1|s)$ | $P(c|d_3) \cdot P(d_3|s) = 1/63$ |
|------------|----------------------------------|
| $P(s|t_1) \cdot P(t_1) / P(s)$ | 0.1449 |

| $P(c|t_2)$ | $P(c|d_4) \cdot P(d_4|s) = 1/72$ |
|------------|----------------------------------|
| 0.001      |                                  |

| $P(t_2|s)$ | $P(c|d_i) \cdot P(d_i|s) = 0 \ (i = 5, 6, 7, 8)$ |
|------------|--------------------------------------------------|
| $P(s|t_2) \cdot P(t_2) / P(s)$ | 0.1159 |

**Result:**

$$P(c|s) = 0.5 \cdot \sum_{i=0}^{2} P(c|t_i)P(t_i|s) + 0.5 \cdot \sum_{j=0}^{8} P(c|d_j)P(d_j|s) = 0.0623$$

Thus, given “database”, we rank the terms as follows: \{\textit{model}(0.0623), \textit{management}(0.0566), \textit{mining}(0.0365), \ldots\}.
Estimation of Some Probabilities

**Estimation of** $P(t_i|s)$

$$P(t_i|s) = \frac{P(s|t_i) \cdot P(t_i)}{P(s)} = \frac{P(t_i) \cdot \prod_{l=1}^{s} P(q_l|t_i)}{P(s') \cdot \sum_{t_j \in T} P(q_{s'|t_j}; s') \cdot P(t_j|s')}$$

where $s' = q_1, q_2, \ldots, q_{|s|-1}$. Obviously, both $P(s')$ and $P(q_{s'|t_j}; s') \cdot P(t_j|s')$ have been computed in the previous round.

**Estimation of** $P(c|d_j)$

$$P(c|d_j) = (1 - \gamma) \frac{\text{count}(c,d_j)}{|d_j|} + \gamma \frac{\text{count}(c,D)}{|D|}$$

where $\frac{\text{count}(c,d_j)}{|d_j|}$ is term frequency of $c$ in the document $d_j$ and $\frac{\text{count}(c,D)}{|D|}$ the frequency of $c$ in $D$. 
### Computation of $P(\text{model} \mid \text{database})$

<table>
<thead>
<tr>
<th>$P(c \mid t_0)$</th>
<th>$0.06$</th>
<th>$P(c \mid d_0)P(d_0 \mid s) = 1/54$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(t_0 \mid s)$</td>
<td>$\frac{P(s \mid t_0) \cdot P(t_0)}{P(s)} = 0.7246$</td>
<td>$P(c \mid d_1) \cdot P(d_1 \mid s) = 1/63$</td>
</tr>
<tr>
<td>$P(c \mid t_1)$</td>
<td>$0.02$</td>
<td>$P(c \mid d_2) \cdot P(d_2 \mid s) = 1/72$</td>
</tr>
<tr>
<td>$P(t_1 \mid s)$</td>
<td>$\frac{P(s \mid t_1) \cdot P(t_1)}{P(s)} = 0.1449$</td>
<td>$P(c \mid d_3) \cdot P(d_3 \mid s) = 1/63$</td>
</tr>
<tr>
<td>$P(c \mid t_2)$</td>
<td>$0.001$</td>
<td>$P(c \mid d_4) \cdot P(d_4 \mid s) = 1/72$</td>
</tr>
<tr>
<td>$P(t_2 \mid s)$</td>
<td>$\frac{P(s \mid t_2) \cdot P(t_2)}{P(s)} = 0.1159$</td>
<td>$P(c \mid d_i) \cdot P(d_i \mid s) = 0$ for $i = 5, 6, 7, 8$</td>
</tr>
</tbody>
</table>

Result:

$$P(c \mid s) = 0.5 \cdot \sum_{i=0}^{2} P(c \mid t_i)P(t_i \mid s) + 0.5 \cdot \sum_{j=0}^{8} P(c \mid d_j)P(d_j \mid s) = 0.0623$$

Thus, given “database”, we rank the terms as follows: \{\text{model}(0.0623), \text{management}(0.0566), \text{mining}(0.0365), \ldots\}.
Outline

1. Motivation
2. Model for Ranking Terms
   - Basic Idea
   - Modeling
3. Progressive Ranking Algorithm
   - Trie Index
   - top-\(k\) Ranking Algorithm
4. Experiments
5. Conclusion
How to efficiently access terms with a prefix?

A sample term set

<table>
<thead>
<tr>
<th>SID</th>
<th>Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>data</td>
</tr>
<tr>
<td>$s_2$</td>
<td>date</td>
</tr>
<tr>
<td>$s_3$</td>
<td>model</td>
</tr>
<tr>
<td>$s_4$</td>
<td>mining</td>
</tr>
<tr>
<td>$s_5$</td>
<td>management</td>
</tr>
<tr>
<td>$s_6$</td>
<td>andy</td>
</tr>
<tr>
<td>$s_7$</td>
<td>apweb</td>
</tr>
</tbody>
</table>

Branch for "management"
Threshold Algorithm (TA)

Monotonous Ranking Function

\[ P(c|s) = \lambda \sum_{t_i \in T} P(c|t_i)P(t_i|s) + (1-\lambda) \sum_{d_j \in D} P(c|d_j)P(d_j|s) \]

As either \( P(c|t_i) \) or \( P(c|d_j) \) increases, \( P(c|s) \) would increase accordingly.

Basic Idea of TA

- Inputs: multiple lists that present the different rankings of the same set of terms; a number, \( k \)
- Output: top-\( k \) ranked terms
- Procedure:
  - Scan multiple lists.
  - Maintain an upper bound \( T \) of all unscanned terms.
  - Compute the score of a term \( c \) when scan it.
  - If the score of \( c \geq T \), insert \( c \) into the result set, \( R \).
  - If \( |R| = k \), terminate the algorithm.
### An Example

**Ranking the top-1 term**
- The context, “database”
- The prefix that the user is typing in, “m”

#### Multiple Lists

| $t_i$ | $P(c|t_i)$ | $d_j$ | $P(c|d_j)$ |
|-------|------------|-------|------------|
| $t_0$ | model (0.06) | $d_0$ | model (0.167) |
|       | mining (0.02) | $d_1$ | mining (0.143) |
|       | management (0.003) | $d_2$ | management (0.143) |
| $t_1$ | model (0.03) | $d_3$ | model (0.125) |
|       | mining (0.01) | $d_4$ | management (0.167) |
|       | management (0.04) | $d_5$ | mining (0.143) |
| $t_2$ | management (0.01) | $d_6$ | mining (0.167) |
|       | model (0.01) | $d_7$ | mining (0.167) |
|       | management (0.004) | $d_8$ | management (0.25) |
The First Step

\[ P(c|t_i) \]

\begin{array}{ccc}
  t_0 & t_1 & t_2 \\
  \text{model (0.06)} & \text{mining (0.03)} & \text{management (0.04)} \\
  \text{mining (0.02)} & \text{model (0.01)} & \text{mining (0.01)} \\
  \text{management (0.003)} & \text{management (0.004)} & \text{model (0.001)} \\
\end{array}

\[ P(c|d_j) \]

\begin{array}{cccc}
  d_0 & d_1 & d_2 & d_3 & d_4 \\
  \text{model (0.167)} & \text{model (0.143)} & \text{model (0.125)} & \text{model (0.167)} & \text{model (0.125)} \\
  \text{management (0.167)} & \text{management (0.143)} & \text{management (0.125)} & \text{management (0.167)} & \text{management (0.125)} \\
  \text{mining (0.143)} & \text{mining (0.167)} & \text{mining (0.167)} & \text{management (0.25)} & \text{management (0.125)} \\
\end{array}

The estimate of \( T \)

\[ T = 0.107 \]

The computation of seen terms

\[ P(\text{model}|\text{database}) = 0.0623, \ P(\text{mining}|\text{database}) = 0.0566 \ \text{and} \ P(\text{management}|\text{database}) = 0.0365 \]
The Second Step

The estimate of $T$

$T = 0.049$

Ranking Result

Since $0.0623 > 0.049$, “model” can be safely inserted into $R$ and the algorithm terminates.
Datasets

- **DBLP**: a bibliography corpus in computer science, where each document contains title, year, authors and venue of a paper.

- **DBLife**: various kinds of web pages in the community of database research, e.g., the homepages of researchers, tour information of a city, personal blogs and photos, etc.

<table>
<thead>
<tr>
<th>Data Sets</th>
<th># of doc.</th>
<th>avg_len</th>
<th>(\Sigma)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBLP</td>
<td>1109727</td>
<td>13.5</td>
<td>406015</td>
</tr>
<tr>
<td>DBLife</td>
<td>8301</td>
<td>224</td>
<td>109716</td>
</tr>
</tbody>
</table>
Query Sets and Settings

Query Sets
- 100 frequently issued queries from the query logs of *iSearch* and *DBLPSearch* (average length = 2.58).
- 100 queries related to database research from the logs from *iSearch* and *DBLPSearch* (average length = 2.8).

Experiment Settings
- C++ compiled with GCC 4.2.3 with -O3 flag
- Ubuntu Linux machine with an Intel Core 2 Quad X5450 3.00GHz processor and 4 GB memory
Baseline and Evaluation Metrics

Baseline

“Word Refinement” in Complete Search: the number of documents which are shared by the context and each term (i.e., co-occurrence).

Evaluation Metric

- 10 volunteers labeled the relevance of each suggested query term in a *blind-test* manner.
- Standard measures of precision and recall
- Two aggregate metrics
  - Average precision at position $n$ ($P@n$)
  - Precision-recall curves
Effectiveness Comparison (DBLife Dataset)

Precision-Recall Curves

- **Our Approach**
- **Complete Search**

![Graph showing precision-recall curves for Our Approach and Complete Search.](image)
Effectiveness Comparison (DBLP Dataset)

Average Precision at Position $n$

![Bar chart showing Average Precision (%)](chart.png)

- **Our Approach**
- **Complete Search**

Position of Suggested Terms

<table>
<thead>
<tr>
<th>Position</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Precision (%)</td>
<td>90</td>
<td>80</td>
<td>70</td>
<td>60</td>
<td>50</td>
<td>40</td>
<td>30</td>
<td>20</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>
Efficiency Comparison on the DBLP Dataset

Average Suggestion Time

- Query Time (ms)
- # of previous query keywords (context)

- Our Approach
- Complete Search

Graph showing the efficiency comparison with a bar chart.
Scalability Comparison

Our Approach
Complete Search

Query Time (ms)

# of documents (* 100K)
Outline

1. Motivation

2. Model for Ranking Terms
   - Basic Idea
   - Modeling

3. Progressive Ranking Algorithm
   - Trie Index
   - top-k Ranking Algorithm

4. Experiments

5. Conclusion
Conclusion

Contributions

- We propose a topic-based query term suggestion model to help users formulate high-quality queries.
- We develop an efficient algorithm to suggest query terms as users type in queries letter by letter, which can save users’ typing efforts.
- Extensive experiments on two real data sets show the effectiveness of our topic-based suggestion model and the efficiency of our algorithm.
Thanks for Your Attention

Questions?