Top-k Interesting Phrase Mining in Ad-hoc Collections using Sequence Pattern Indexing

Chuancong Gao, Sebastian Michel

Max-Planck Institute for Informatics & Saarland University
Saarbrücken, Germany

EDBT, March 28, 2012
Motivation

Lost overview on current presidential campaign topics?
Get an update on key phrases of Mitt Romney!

- Want concise summary.
- Solution: Mining top-k phrases w.r.t. subset of all documents.
- Sub-set of documents is determined in ad-hoc fashion.
- E.g., using keyword queries, restriction to categorical attributes.
## Motivation (Example)

<table>
<thead>
<tr>
<th>Year</th>
<th>Candidate</th>
<th>Slogan</th>
</tr>
</thead>
<tbody>
<tr>
<td>1988</td>
<td>George Bush</td>
<td>Kinder, Gentler Nation</td>
</tr>
<tr>
<td>1992</td>
<td>Bill Clinton</td>
<td>Don't stop thinking about tomorrow</td>
</tr>
<tr>
<td>1992</td>
<td>Bill Clinton</td>
<td>Putting People First</td>
</tr>
<tr>
<td>1992</td>
<td>Ross Perot</td>
<td>Ross for Boss</td>
</tr>
<tr>
<td>1996</td>
<td>Bill Clinton</td>
<td>Building a bridge to the 21st century</td>
</tr>
<tr>
<td>1996</td>
<td>Bob Dole</td>
<td>The Better Man for a Better America</td>
</tr>
<tr>
<td>2000</td>
<td>Al Gore</td>
<td>Prosperity and progress</td>
</tr>
<tr>
<td>2000</td>
<td>George W. Bush</td>
<td>Compassionate conservatism</td>
</tr>
<tr>
<td>2000</td>
<td>George W. Bush</td>
<td>Leave no child behind</td>
</tr>
<tr>
<td>2000</td>
<td>Ralph Nader</td>
<td>Government of, by, and for the people...</td>
</tr>
<tr>
<td>2004</td>
<td>John Kerry</td>
<td>Let America be America Again</td>
</tr>
<tr>
<td>2004</td>
<td>George W. Bush</td>
<td>Yes, America Can!</td>
</tr>
<tr>
<td>2008</td>
<td>John McCain</td>
<td>Country First</td>
</tr>
<tr>
<td>2008</td>
<td>Barack Obama</td>
<td>Change We Can Believe In</td>
</tr>
<tr>
<td>2008</td>
<td>Barack Obama</td>
<td>Yes We Can!</td>
</tr>
</tbody>
</table>

[http://www.presidentsusa.net/campaignslogans.html](http://www.presidentsusa.net/campaignslogans.html)
Problem Statement

Given: a document corpus

Task: Create an Index

- that organizes documents and phrases
- in a compact way
- to query for top-k phrases of arbitrary ad-hoc collections.
- Devise appropriate algorithms to use the index efficiently.
Problem Definition

Given a document corpus $D$ and a subset $D'$.

Find the top-$k$ frequent phrases with the highest interestingness in $D'$.

Ad-hoc Document Subset
$D'$ is not known upfront

Phrase
A phrase $p$ is contained in document $d$ iff $p$ is a continuous sub-sequence of $d$ (notation $p \sqsubseteq d$).

- $p$'s length is restricted by $\text{min-len}$ and $\text{max-len}$.
Problem Definition (Cont.)

Frequency
The frequency of a phrase $p$ in $D'$ is called the support value of $p$ w.r.t. $D'$, denoted by $sup_{p}^{D'}$.

- $sup_{p}^{D'} = |\{d | d \in D' \land p \subseteq d\}|$
- A phrase is called frequent in a document collection $D'$ iff $p$'s support is larger than the user-specified threshold $min-sup_{D'}^{D'}$.

Interestingness
The interestingness measure we adopted is the confidence value (or relevance value), defined as

$$conf_{p}^{D'} = \frac{sup_{p}^{D'}}{sup_{D}}$$
Example

Find the top-\(k\) \((k = 3)\) interesting phrases for 4 documents.

\[(a)\] Documents
\((min-sup^{D'} = 2, k = 3, min-len = 2, max-len = 4)\)

\[(b)\] Top-\(k\) Phrases
Outline

Existing Approaches

Sequence Pattern Indexing

Experiments

Conclusion
Existing Approaches - Phrase Inverted Indexing

(A. Simitsis et al. - Multidimensional content eXploration. VLDB’08)

Indexing Steps
For each phrase $p$, store an inverted list that contains the IDs of all documents containing $p$.

Example index:

<table>
<thead>
<tr>
<th>ID</th>
<th>Document IDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>${d_1, d_2}$</td>
</tr>
<tr>
<td>$p_2$</td>
<td>${d_1, d_3}$</td>
</tr>
<tr>
<td>$p_3$</td>
<td>${d_1, d_3, d_4}$</td>
</tr>
<tr>
<td>$p_4$</td>
<td>${d_1, d_2, d_3, d_4}$</td>
</tr>
</tbody>
</table>

Given an ad-hoc collection $D'$: Alg. requires a full scan of all the inverted lists.
Existing Approaches - Forward Indexing

(S. Bedathur et al. - Interesting-Phrase Mining for Ad-Hoc Text Analytics. VLDB’10)

Indexing Steps

Builds a forward list for each document $d \in D$, containing all the IDs of phrases contained in $d$.

Example index structure:

<table>
<thead>
<tr>
<th>ID</th>
<th>Forward List</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$</td>
<td>${p_1:2, p_2:2, p_3:3, p_4:4}$</td>
</tr>
<tr>
<td>$d_2$</td>
<td>${p_1:2, p_4:4}$</td>
</tr>
<tr>
<td>$d_3$</td>
<td>${p_2:2, p_3:3, p_4:4}$</td>
</tr>
<tr>
<td>$d_4$</td>
<td>${p_3:3, p_4:4}$</td>
</tr>
</tbody>
</table>

Numbers after “:" denote global support values.
Existing Approaches - Forward Indexing (Cont.)

Query Processing

Compute top-\(k\) phrases with \(|D'|\)-way merge join.

- Global support value of \(p\) stored explicitly in list.
- Local support value + confidence is computed during the join process.
- Nice: only the forward lists of documents in \(D'\) need be scanned.
- Even better: can apply early termination:
  - phrase IDs in each forward list stored in ascending order of global support values
  - then, \(\text{conf}_{p}^{D'}\) has an upper bound \(\min\left\{1, \frac{|D'|}{\text{sup}_{p}^{D}}\right\}\).
  - Can stop if the top-k results we found can not be topped anymore.
Prefix-Maximal Indexing

(S. Bedathur et al. - Interesting-Phrase Mining for Ad-Hoc Text Analytics. VLDB’10)

Key Idea

Store only prefix-maximal phrases (small subset of all phrases) w.r.t. each document in the corpus.

- A phrase \( p \) is called **maximal** w.r.t. a document \( d \in D \) iff \( \not\exists p' \) that \( p' \) is frequent and \( p \sqsubseteq p' \).

- A phrase \( p \) is called **prefix-maximal** w.r.t. a document \( d \in D \) iff \( p \) is maximal and \( \not\exists p' \) that \( p' \) is maximal and \( p \) is a prefix of \( p' \).

Significantly reduces the index size comparing to Forward Indexing.
Phrase-Maximal Index. Comparison

Comparison: all phrases, maximal phrases, and prefix-maximal phrases:

<table>
<thead>
<tr>
<th>ID</th>
<th>Phrases</th>
</tr>
</thead>
<tbody>
<tr>
<td>d₁</td>
<td>{ad, be, bef, befc, ca, cad, ea, ef, efca, fc, fca, fcad}</td>
</tr>
<tr>
<td>d₂</td>
<td>{be, bea, cb, cbe, cbea, dc, dcb, dcbe, ea, fe}</td>
</tr>
<tr>
<td>d₃</td>
<td>{ac, acd, cd, ce, cea, ceac, ea, eac, eacd, fc, fce, fcea}</td>
</tr>
<tr>
<td>d₄</td>
<td>{ac, acd, acf, acfd, be, bea, cd, cf, cfd, ea, eac, fd}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Prefix-Maximal Phrases</th>
</tr>
</thead>
<tbody>
<tr>
<td>d₁</td>
<td>{ad, befc, cad, ea, efca, fcad}</td>
</tr>
<tr>
<td>d₂</td>
<td>{bea, cbea, dcbe, ea, fe}</td>
</tr>
<tr>
<td>d₃</td>
<td>{acd, cd, ceac, eacd, fce}</td>
</tr>
<tr>
<td>d₄</td>
<td>{acd, acfd, bea, cfd, eac, fd}</td>
</tr>
</tbody>
</table>

Maximal Phrases

Max. phrases keep index small, but top-k phrases computation is difficult.
The top-$k$ phrases are computed by $|D'|$-way merge join.

- Support values computed when merging all the forward lists in $D'$.

- The confidence values are calculated by using a separate global support value dictionary.

- Issues:
  - A full scan of the forward lists of $D'$ is required, since no early termination can be applied.
Objectives

Devise an approach that

1. is as fast as Forward Indexing (i.e., we must devise early stopping)

2. and has an index as small (or smaller than) Phrase-Maximal Indexing
Outline

Existing Approaches

Sequence Patter Indexing

Experiments

Conclusion
Our Approach - Key Ideas

We also store prefix-maximal phrases for each document \( d \in D \), but ...

Lexicographical order vs. matching position order:

<table>
<thead>
<tr>
<th>ID</th>
<th>Prefix-Maximal Phrases</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d_1 )</td>
<td>{ad, befc, cad, ea, efca, fcad}</td>
</tr>
<tr>
<td>( d_2 )</td>
<td>{bea, cbea, dcbe, ea, fe}</td>
</tr>
<tr>
<td>( d_3 )</td>
<td>{acd, cd, ceac, eacd, fcea}</td>
</tr>
<tr>
<td>( d_4 )</td>
<td>{acd, acfd, bea, cfd, eac, fd}</td>
</tr>
</tbody>
</table>

... ... ...

<table>
<thead>
<tr>
<th>ID</th>
<th>Prefix-Maximal Phrases (Ordered by Position)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d_1 )</td>
<td>{ea@1, befc@3, efca@4, fcad@5, cad@6, ad@7}</td>
</tr>
<tr>
<td>( d_2 )</td>
<td>{dcbe@1, cbea@2, bea@3, ea@4, fe@6}</td>
</tr>
<tr>
<td>( d_3 )</td>
<td>{fcea@1, ceac@2, eacd@3, acd@4, cd@5}</td>
</tr>
<tr>
<td>( d_4 )</td>
<td>{acfd@1, cfd@2, fd@3, bea@5, eacd@6, acd@7}</td>
</tr>
</tbody>
</table>

... ... ...

☆ – Duplicate Prefix Sub-Phrase Comparing to Previous Phrase  □ – Duplicate Prefixes among Phrases
Our Approach - Indexing

ID  Prefix-Maximal Phrases (Ordered by Position)

\[ d_1 \{ ea@1, befc@3, efca@4, fcad@5, cad@6, ad@7 \} \]

- Sub-phrases are organized in a forest structure; each node contains one item + length information.
- Links from outside point to start items of indexed prefix-maximal phrases.
- Possible length information is stored in bit array.
Memory Layout

(Number below cells indicate a relative memory address)

Each node is assigned with 4 bytes (32 bits); the first byte storing flags, the last 3 bytes store the item.
Still some duplications like $e \rightarrow f \rightarrow c \rightarrow a$ and $a \rightarrow d$.

Further improvement: utilize specific matching positions (instead of just matching orders).

Idea: Use graph structure to eliminate more duplicates.
Our Approach - Top-\(k\) Phrase Computing

Computation is done in two steps (interleaved): \(|D'|\)-way merge join and growing patterns:

Merge join approach is used on phrase prefixes with length 1

- Index contains an address list for *different items* in document \(d\).
- Apply \(|D'|\)-way merge join on these address lists to retrieve all length-1 phrases.

Extend each length-1 phrase

In each step: the prefix is extended with one of the local frequent items.
Optimizations

Early Stopping

- Can tell when it is “OK” to stop retrieving more length 1 patterns

Search Space Pruning

- During expansion of a length-1 pattern to larger patterns, stop when unpromising.
Our Approach - Summary

Our Method’s Advantages:

- Index with little redundancy
- Algorithms fully utilize this new index structure.
- Visiting only frequent phrases:
  - With early termination in $|D'|$-merge join process.
  - And search space pruning in pattern-growth process.
Outline

Existing Approaches

Sequence Pattern Indexing

Experiments

Conclusion
Evaluation

Different versions of own approach, plus two competitors. Using an Intel Xeon W3520 (2.66 GHz) CPU and 24 GB memory workstation. Implementation in C#. Index in memory.

Algorithms under Comparison

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baselines</strong></td>
<td></td>
</tr>
<tr>
<td>ForwardIndex</td>
<td>Forward Indexing</td>
</tr>
<tr>
<td>PreMaxIndex</td>
<td>Prefix-Maximal Indexing</td>
</tr>
<tr>
<td><strong>Our Approaches</strong></td>
<td></td>
</tr>
<tr>
<td>SeqPattIndex</td>
<td>Our Indexing</td>
</tr>
<tr>
<td>SeqPattIndex_IM</td>
<td>Our Improved Indexing</td>
</tr>
<tr>
<td>SeqPattIndex_NP</td>
<td>SeqPattIndex With Early Termination and Search Space Pruning Turned off</td>
</tr>
</tbody>
</table>
Datasets

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PubMed Titles</td>
<td>17,826,927</td>
<td>2,028,673</td>
<td>167</td>
<td>11.59</td>
<td>10.99</td>
</tr>
<tr>
<td>PubMed Abstracts</td>
<td>2,500,000</td>
<td>2,075,526</td>
<td>1599</td>
<td>149.72</td>
<td>92.1</td>
</tr>
</tbody>
</table>

Extracted from PubMed – the largest free publication database on life sciences and biomedical topics.

- Titles are rather short, little redundancy per title.
- Abstracts contain much more information, more redundancy document.

Great to study pros and cons of the discussed approaches.

Queries: Generated

- to simulate different kind of queries (by size of $D'$)
- grouped phrases by global support according to several ranges
Evaluation - In-Memory Index Size (in MB)

- PubMed Titles: improved approach close to original one, as phrases in titles have little redundancy
Evaluation - Average Querying Time (in sec)

PubMed Titles

(a) Varying $|D'|$ ($k = 10$, min-sup = 5)

(b) Varying $k$ (min-sup = 5)
Evaluation - Average Querying Time (in sec)

PubMed Abstract

(a) Varying $|D'|$ ($k = 10$, $min-sup = 5$)

(b) Varying $k$ ($min-sup = 5$)
Outline

Existing Approaches

Sequence Pattern Indexing

Experiments

Conclusion
Conclusion

- Presented approach to ad-hoc frequent phrases mining
- Making use of redundancy of sequentially aligned phrases
- Index is very compact.
- High performance in query processing based on early stopping and search space pruning.