8. Mining & Organization
Retrieving a list of relevant documents (10 blue links) insufficient for vague or exploratory information needs (e.g., “find out about brazil”) when there are more documents than users can possibly inspect.

Organizing and visualizing collections of documents can help users to explore and digest the contained information, e.g.:

- **Clustering** groups content-wise similar documents
- **Faceted search** provides users with means of exploration
- **Timelines** visualize contents of timestamped document collections
8.1. Clustering
8.2. Faceted Search
8.3. Tracking Memes
8.4. Timelines
8.5. Interesting Phrases
8.1. Clustering

- Clustering groups content-wise similar documents

- Clustering can be used to structure a document collection (e.g., entire corpus or query results)

- Clustering methods: DBScan, \(k\)-Means, \(k\)-Medoids, hierarchical agglomerative clustering

- Example of search result clustering: clusty.com
**k-Means**

- **Cosine similarity** $\text{sim}(c,d)$ between document vectors $c$ and $d$
- **Clusters** $C_i$ represented by a cluster centroid document vector $c_i$
- **k-Means** groups documents into $k$ clusters, maximizing the average similarity between documents and their cluster centroid

$$
\frac{1}{|D|} \sum_{d \in D} \max_{c \in C} \text{sim}(c, d)
$$

- Document $d$ is assigned to cluster $C$ having most similar centroid
Documents-to-Centroids

- k-Means is typically implemented iteratively with every iteration reading all documents and assigning them to most similar cluster
  - initialize cluster centroids $c_1, \ldots, c_k$ (e.g., as random documents)
  - while not converged (i.e., cluster assignments unchanged)
    - for every document $d$, determine most similar $c_i$, and assign it to $C_i$
    - recompute $c_i$ as mean of documents assigned to cluster $C_i$

- Problem: Iterations need to read the entire document collection, which has cost in $O(nkd)$ with $n$ as number of documents, $k$ as number of clusters and, and $d$ as number of dimensions
Centroids-to-Documents

Broder et al. [1] devise an alternative method to implement k-Means, which makes use of established IR methods

Key Ideas:

- build an **inverted index** of the document collection
- treat **centroids as queries** and identify the top-$l$ most similar documents in every iteration using **WAND**
- documents showing up in **multiple top-$l$ results** are assigned to the most similar centroid
- **recompute centroids** based on assigned documents
- finally, assign **outliers** to cluster with most similar centroid
Sparsification

- While documents are typically sparse (i.e., contain only relatively few features with non-zero weight), cluster centroids are dense.

- Identification of top-\(l\) most similar documents to a cluster centroid can further be speeded up by sparsifying, i.e., considering only the \(p\) features having highest weight.
Experiments

- **Datasets**: Two datasets each with about 1M documents but different numbers of dimensions: ~26M for (1), ~7M for (2)

<table>
<thead>
<tr>
<th>System</th>
<th>ℓ</th>
<th>Dataset 1 Similarity</th>
<th>Dataset 1 Time</th>
<th>Dataset 2 Similarity</th>
<th>Dataset 2 Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-means</td>
<td>—</td>
<td>0.7804</td>
<td>445.05</td>
<td>0.2856</td>
<td>705.21</td>
</tr>
<tr>
<td>wand-k-means</td>
<td>100</td>
<td>0.7810</td>
<td>83.54</td>
<td>0.2858</td>
<td>324.78</td>
</tr>
<tr>
<td>wand-k-means</td>
<td>10</td>
<td>0.7811</td>
<td>75.88</td>
<td>0.2856</td>
<td>243.9</td>
</tr>
<tr>
<td>wand-k-means</td>
<td>1</td>
<td>0.7813</td>
<td>61.17</td>
<td>0.2709</td>
<td>100.84</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System</th>
<th>p</th>
<th>ℓ</th>
<th>Dataset 1 Similarity</th>
<th>Dataset 1 Time</th>
<th>ℓ</th>
<th>Dataset 2 Similarity</th>
<th>Dataset 2 Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-means</td>
<td>—</td>
<td>—</td>
<td>0.7804</td>
<td>445.05</td>
<td>—</td>
<td>0.2858</td>
<td>705.21</td>
</tr>
<tr>
<td>wand-k-means</td>
<td>—</td>
<td>1</td>
<td>0.7813</td>
<td>61.17</td>
<td>10</td>
<td>0.2856</td>
<td>243.9</td>
</tr>
<tr>
<td>wand-k-means</td>
<td>500</td>
<td>1</td>
<td>0.7817</td>
<td>8.83</td>
<td>10</td>
<td>0.2704</td>
<td>4.00</td>
</tr>
<tr>
<td>wand-k-means</td>
<td>200</td>
<td>1</td>
<td>0.7814</td>
<td>6.18</td>
<td>10</td>
<td>0.2855</td>
<td>2.97</td>
</tr>
<tr>
<td>wand-k-means</td>
<td>100</td>
<td>1</td>
<td>0.7814</td>
<td>4.72</td>
<td>10</td>
<td>0.2853</td>
<td>1.94</td>
</tr>
<tr>
<td>wand-k-means</td>
<td>50</td>
<td>1</td>
<td>0.7803</td>
<td>3.90</td>
<td>10</td>
<td>0.2844</td>
<td>1.39</td>
</tr>
</tbody>
</table>

- **Time per iteration** reduced from 445 minutes to 3.9 minutes on Dataset 1; 705 minutes to 1.39 minutes on Dataset 2
8.2. Faceted Search
8.2. Faceted Search

CompleteSearch DBLP

a DBLP mirror with extended search capabilities maintained by Hannah Bast, University of Freiburg (formerly MPI Saarbrücken)

zoomed in on 276 documents ... NEW: get these search results as XML, JSON, JSONP

<table>
<thead>
<tr>
<th>Year</th>
<th>ID</th>
<th>Title</th>
<th>Authors</th>
<th>Abstract</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014</td>
<td>274</td>
<td>He Li, Jaesoo Yoo: An efficient scheme for continuous skyline query processing over dynamic data sets. BigComp 2014: 54-59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2014</td>
<td>268</td>
<td>Alfredo Cuzzocrea, Jose Cecilio, Pedro Furtado: An Effective and Efficient Middleware for Supporting Distributed Query Processing in Large-Scale Cyber-Physical Systems. IDCs 2014: 124-135</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2014</td>
<td>261</td>
<td>Merit Seran Usay, Christian Beeske, Thomas Seidl: An Efficient Query Processing with the Earth Mover’s Distance. PKM’14 (2014)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2014</td>
<td>259</td>
<td>Thomas Kargies, Matthias Hille, Mario Ludwig, Dirk Habich, Wolfgang Lehner, Max Hemel, Volkmar Krueger: Demonstrating efficient query processing in heterogeneous environments. SIGMOD 2014: 693-698</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
8.2. Faceted Search

[Image of an Amazon search results page for digital cameras]
8.2. Faceted Search

<table>
<thead>
<tr>
<th>Facet</th>
<th>Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>Asia, Afghanistan, China, India, Japan, Turkey, Turkmenistan</td>
</tr>
<tr>
<td>Date</td>
<td>17th century, 18th century, 19th century, 20th century, date unknown</td>
</tr>
<tr>
<td>Theme</td>
<td>Music, writing, and sport, natural, religion</td>
</tr>
<tr>
<td>Object</td>
<td>Clothing, food, furnishings, ropes, vessels</td>
</tr>
<tr>
<td>Nature</td>
<td>Objects of water, fish, flowers, geological formations, human, invertebrates and arthropods, animal, marine, plant material, trees</td>
</tr>
<tr>
<td>Place and Space</td>
<td>Buildings, cities, landscapes</td>
</tr>
</tbody>
</table>

The opening page shows a text search box and the first level of metadata terms. Hovering over a facet name yields a tooltip (here shown below "Location") explaining the meaning of the facet.

The leaf-level category labels were manually organized into hierarchical facets, using breadth and depth guidelines similar to those in [2].

**INTERFACE DESIGN**

**The Faceted Category Interface**

**Unifying Goals**

Our design goals are to support search usability guidelines [16], while avoiding negative consequences like empty result sets or feelings of being lost. Because searching and browsing are useful for different types of tasks, our design strives to seamlessly integrate both searching and browsing functionality throughout. Results can be selected by keyword search, by pre-assigned metadata terms, or by a combination of both. Each facet is associated with a particular hue throughout the interface. Categories, query terms, and item groups in each facet are shown in lightly shaded boxes, whose colors are computed by adjusting value and saturation but maintaining the appropriate hue.

In working with a large collection of items and a large number of metadata terms, it is essential to avoid overwhelming the user with complexity. We do this by keeping results organized, by sticking to simple point-and-click interactions instead of imposing any special query syntax on the user, and by not showing any links that would lead to zero results. Every hyperlink that selects a new result set is displayed with a query preview (an indicator of the number of results to expect).

The design can be thought of as having three stages, by rough analogy to a game of chess: the opening, middle game, and endgame. The most natural progression is to proceed through the stages in order, but users are not forced to do so.

**Opening**

The primary aims of the opening are to present a broad overview of the entire collection and to allow many starting paths for exploration. The opening page (Figure 1) displays each metadata facet along with its top-level categories. This provides many navigation possibilities, while immediately familiarizing the user with the high-level information structure of the collection. The opening also provides a text box for entering keyword searches, giving the user the freedom to choose between starting by searching or browsing.

Selecting a category or entering a keyword gathers an initial result set of matching items for further refinement, and brings the user into the middle game.

**Middle Game**

In the middle game (Figure 2) the result set is evaluated and manipulated, usually to narrow it down. There are three main parts of this display: the result set, which occupies most of the page; the category terms that apply to the items in the result set, which are listed along the left by facet (we refer to this category listing as The Matrix); and the current query, which is shown at the top. A search box remains available (for searching within the current result set or within the entire collection), and a link provides a way to return to the opening.

The key aim here is organization, so the design offers flexible methods of organizing the results. The items in the result set can be sorted on various fields, or they can be grouped into categories by any facet. Selecting a category both narrows the result set and organizes the result set in terms of the newly selected facet. For instance, suppose a user is currently looking at the results of selecting the category Bridges from the Places facet. If the user then selects Europe from the Locations facet, not only is the category Europe added to the query, but the results are organized by the subcategories of Europe, namely France, Italy, and so on. Generalizing or removing a category term broadens the result set. Selecting an individual item takes the user to the endgame.

---

**Flamenco**

Search further within these categories:

- Costume
- Drawing
- Photograph
- Woodcut
- Woven Object
- Location, Asia
  - Afghanistan (1), China (4), India (3), Japan (1), Turkey (2), Turkmenistan (1)
- Date
  - 17th century (3), 18th century (4), 19th century (1), 20th century (5), date unknown (4)
- Theme
  - Music, writing, and sport (1), natural (1), religion (2)
- Object
  - Clothing (1), food (1), furnishings (4), ropes, vessels (1)
- Nature
  - Objects of water (2), fish (1), flowers (1), geological formations (1), human (1), invertebrates and arthropods (1), animal (1), marine (1), plant material (6), trees (1)
- Place and Space
  - Buildings (1), cities (1), landscapes (3)

This term defines your current search. Click the X to remove a term.

**Start a new search**

**Shapes, Colors, and Materials, Fabrics**

28 items (grouped by location)
Faceted Search

- **Faceted search** [3,7] supports the user in exploring/navigating a collection of documents (e.g., query results).

- **Facets** are orthogonal sets of categories that can be flat or hierarchical, e.g.:
  - **topic**: arts & photography, biographies & memoirs, etc.
  - **origin**: Europe > France > Provence, Asia > China > Beijing, etc.
  - **price**: 1–10$, 11–50$, 51–100$, etc.

- Facets are manually curated or automatically derived from meta-data.
Automatic Facet Generation

- Need to manually curate facets **prevents their application** for **large-scale** document collections with **sparse meta-data**

- Dou et al. [3] investigate how facets can be **automatically mined** in a **query-dependent manner** from pseudo-relevant documents

- **Observation**: **Categories** (e.g., brands, price ranges, colors, sizes, etc.) are typically **represented as lists** in web pages

- **Idea**: Extract lists from web pages, rank and cluster them, and use the **consolidated lists as facets**
List Extraction

- Lists are **extracted from web pages** using several patterns
  - **Enumerations** of items in text (e.g., we serve beef, lamb, and chicken) via: `item{, item}* (and|or) {other} item`
  - **HTML form elements** (<SELECT>) and **lists** (<UL><OL>) ignoring instructions such as “select” or “chose”
  - as rows and columns of **HTML tables** (<TABLE>) ignoring header and footer rows

- Items in extracted lists are **post-processed**, removing non-alphanumerical characters (e.g., brackets), converting them to lower case, and removing items longer than 20 terms
Some of the extracted lists are **spurious** (e.g., from HTML tables)

**Intuition**: Good lists consist of items that are **informative** to the query, i.e., are **mentioned in many** pseudo-relevant documents

**Lists weighted** taking into account a document matching weight $S_{DOC}$ and their average inverse document frequency $S_{IDF}$

$$S_l = S_{DOC} \cdot S_{IDF}$$

**Document matching weight $S_{DOC}$**

$$S_{DOC} = \sum_{d \in R} (s_d^m \cdot s_d^r)$$

with $s_d^m$ as fraction of list items mention in document $d$ and $s_d^r$ as importance of document $d$ (estimated as $\text{rank}(d)-1/2$)
List Weighting

- Average inverse document $S_{IDF}$ is defined as

$$S_{IDF} = \frac{1}{|l|} \sum_{i \in l} idf(i)$$

- **Problem**: Individual lists (extracted from a single document) may still contain **noise**, be **incomplete**, or **overlap** with other lists

- **Idea**: Cluster lists containing similar items to consolidate them and form dimensions that can be used as facets
List Clustering

- **Distance between two lists** is defined as

\[ d(l_1, l_2) = 1 - \frac{|l_1 \cap l_2|}{\min\{|l_1|, |l_2|\}} \]

- **Complete-linkage distance** between two clusters

\[ d(c_1, c_2) = \max_{l_1 \in c_1, l_2 \in c_2} d(l_1, l_2) \]

- **Greedy clustering algorithm**
  - pick most important not-yet-clustered list
  - add nearest lists while cluster diameter is smaller than \( \text{Dia}_{max} \)
  - save cluster if total weight is larger than \( W_{min} \)
Problem: In which order to present dimensions and items therein?

**Importance of a dimension** (cluster) is defined as

\[ S_c = \sum_{s \in Sites(c)} \max_{l \in c, l \in s} S_l \]

favoring dimensions grouping lists with high weight

**Importance of an item** within a dimension defined as

\[ S_{i|c} = \sum_{s \in Sites(c)} \frac{1}{\sqrt{AvgRank(c, i, s)}} \]

favoring items which are often ranked high within containing lists
Anecdotal Results

Dimensions mined from top-100 of commercial search engine

query: **watches**
1. cartier, breitling, omega, citizen, tag heuer, bulova, casio, rolex, auemars piguet, seiko, accurton, movado, fossil, gucci, ...
2. men’s, women’s, kids, unisex
3. analog, digital, chronograph, analog digital, quartz, mechanical, manual, automatic, electric, dive, ...
4. dress, casual, sport, fashion, luxury, bling, pocket, ...
5. black, blue, white, green, red, brown, pink, orange, yellow, ...

query: **lost**
1. season 1, season 6, season 2, season 3, season 4, season 5
2. matthew fox, naveen andrews, evangeline lilly, josh holloway, jorge garcia, daniel dae kim, michael emerson, terry o’quinn, ...
3. jack, kate, locke, sawyer, claire, sayid, hurley, desmond, Boone, charlie, ben, juliet, sun, jin, ana, lara ...
4. what they died for, across the sea, what kate does, the candidate, the last recruit, everybody loves hag, the end, ...

query: **lost season 5**
1. because you left, the lie, follow the leader, jughead, 316, dead is dead, some like it hoth, whatever happened happened, the little prince, this place is death, the variable, ...
2. jack, kate, hurley, sawyer, sayid, ben, juliet, locke, miles, desmond, charlotte, variable, sun, none, richard, daniel
3. matthew fox, naveen andrews, evangeline lilly, jorge garcia, henry ian cusick, josh holloway, michael emerson, ...
4. season 1, season 3, season 2, season 6, season 4

query: **flowers**
1. birthday, anniversary, thanksgiving, get well, congratulations, christmas, thank you, new baby, sympathy, fall
2. roses, best sellers, plants, carnations, lilies, sunflowers, tulips, gerberas, orchids, iris
3. blue, orange, pink, red, purple, white, green, yellow

query: **what is the fastest animals in the world**
1. cheetah, pronghorn antelope, lion, thomson’s gazelle, wildebeest, cape hunting dog, elk, coyote, quarter horse
2. birds, fish, mammals, animals, reptiles
3. science, technology, entertainment, nature, sports, lifestyle, travel, game, world business

query: **the presidents of the united states**
1. john adams, thomas jefferson, george washington, john tyler, james madison, abraham lincoln, john quincy adams, william henry harrison, martin van buren, james monroe, ...
2. the presidents of the united states of america, the presidents of the united states ii, love everybody, pure frosting, these are the good times people, freaked out and small, ...
3. kitty, lump, peaches, dune buggy, feather pluckn, back porch, kick out the jams, stranger, boll weevil, ca plane pour moi, ...
4. federalist, democratic-republican, whig, democratic, republican, no party, national union, ...

query: **visit beijing**
1. tiananmen square, forbidden city, summer palace, temple of heaven, great wall, beihai park, hutong
2. attractions, shopping, dining, nightlife, tours, travel tip, transportation, facts

query: **cikm**
1. databases, information retrieval, knowledge management, industry research track
2. submission, important dates, topics, overview, scope, committee, organization, programme, registration, cfp, publication, programme committee, organisers, ...
3. acl, kdd, chi, sigir, www, icml, focs, ijcai, osdi, sigmod, sosp, stoc, uist, vldb, wsdm, ...
8.3. Tracking Memes

- Leskovec et al. [5] track memes (e.g., “lipstick on a pig”) and visualize their volume in traditional news and blogs.

- Demo: http://www.memetracker.org

---

Figure 4: Top 50 threads in the news cycle with highest volume for the period Aug. 1 – Oct. 31, 2008. Each thread consists of all news articles and blog posts containing a textual variant of a particular quoted phrases. (Phrase variants for the two largest threads in each week are shown as labels pointing to the corresponding thread.) The data is drawn as a stacked plot in which the thickness of the strand corresponding to each thread indicates its volume over time. Interactive visualization is available at http://memetracker.org.

---

Figure 5: Temporal dynamics of top threads as generated by our model. Only two ingredients, namely imitation and a preference to recent threads, are enough to qualitatively reproduce the observed dynamics of the news cycle.

---

3. GLOBAL ANALYSIS: TEMPORAL VARIATION AND A PROBABILISTIC MODEL

Having produced phrase clusters, we now construct the individual elements of the news cycle. We define a thread associated with an active phrase cluster to be these items (news article or blog post) containing some phrase from the cluster, and we then track all threads over time, considering both their individual temporal dynamics as well as their interactions with one another.

Using our approach we completely automatically created and also automatically labeled the plot in Figure 4, which depicts the 50 largest threads for the three-month period Aug. 1 – Oct. 31. It is drawn as a stacked plot, a style of visualization (see e.g. [16]) in which the thickness of each strand corresponds to the volume of the corresponding thread over time, with the total area equal to the total volume. We see that the rising and falling pattern does in fact tell us about the patterns by which blogs and the media successively focus and defocus on common story lines.

An important point to note at the outset is that the total number of articles and posts, as well as the total number of quotes, is approximately constant over all weekdays in our dataset. (Refer to [1] for the plots.) As a result, the temporal variation exhibited in Figure 4 is not the result of variations in the overall amount of global news and blogging activity from one day to the next. Rather, the periods when the upper envelope of the curve are high correspond to times when there is a greater degree of convergence on key stories, while the low periods indicate that attention is more diffuse, spread out over many stories. There is a clear weekly pattern in this (again, despite the relatively constant overall volume), with the five large peaks between late August and late September corresponding, respectively, to the Democratic and Republican National Conventions, the overwhelming volume of the “lipstick on a pig” thread, the beginning of peak public attention to the financial crisis, and the negotiations over the financial bailout plan. Notice how the plot captures the dynamics of the presidential campaign coverage at a very fine resolution. Spikes and the phrases pinpoint the exact events and moments that triggered large amounts of attention.

Moreover, we have evaluated competing baselines in which we produce topic clusters using standard methods based on probabilistic term mixtures (e.g. [7, 8]).

The clusters produced for this time period correspond to much coarser divisions of the content (politics, technology, movies, and a number of essentially unrecognizable clusters). This is consistent with our initial observation in Section 1 that topical clusters are working at a level of granularity different from what is needed to talk about the news cycle. Similarly, producing clusters from the most linked-to documents [23] in the 2nd As these do not scale to the size of the data we have here, we could only use a subset of 10,000 most highly linked-to articles.
Phrase Graph Construction

Problem: Memes are often modified as they spread, so that first all mentions of the same meme need to be identified

Construction of a phrase graph $G(V, E)$:

- vertices $V$ correspond to mentions of a meme that are reasonably long and occur often enough
- edge $(u, v)$ exists if meme mentions $u$ and $v$
  - $u$ is strictly shorter than $v$
  - either: have small directed token-level edit distance (i.e., $u$ can be transformed into $v$ by adding at most $\varepsilon$ tokens)
  - or: have a common word sequence of length at least $k$
- edge weights based on edit distance between $u$ and $v$ and how often $v$ occurs in the document collection
Phrase Graph Partitioning

- Phrase graph is an **directed acyclic graph** (DAG) by construction.

- Partition G(V, E) by **deleting a set of edges** having minimum total weight, so that each resulting **component is single-rooted**.

- Phrase graph partitioning is **NP-hard**, hence addressed by **greedy heuristic algorithm**.
Clustering of meme mentions allows for insightful analyses, e.g.:

- **volume of meme** per time interval
- **peek time** of meme in traditional news and social media
- **time lag** between peek times in traditional news and social media
8.4. Timelines

- **Timelines** visualize, e.g., major events and topics and their occurrence/importance as they occur in a collection of timestamped documents.
Swan and Allan [6] devise an approach based on statistical tests to automatically generate a timeline from a collection of timestamped documents (e.g., entire corpus or query result).

- consider only named entities (e.g., persons, organizations, locations) and noun phrases (e.g., nuclear power plant, debt crisis, car insurance)

- partition document collection at day granularity
Problem: How to identify significantly time-varying features?

Assume that the following **statistics** have been computed:

- $N_d$ as the number of documents in the partition for day $d$
- $N$ as the number of documents in the document collection
- $f_d$ as the number of documents with feature $f$ in the partition for day $d$
- $F$ as the number of documents with feature $f$ in the document collection

Derive a **contingency table** from these statistics:

<table>
<thead>
<tr>
<th></th>
<th>$f$</th>
<th>$\neg f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d$</td>
<td>$f_d$</td>
<td>$N_d - f_d$</td>
</tr>
<tr>
<td>$\neg d$</td>
<td>$F - f_d$</td>
<td>$N - N_d - F + f_d$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$f$</th>
<th>$\neg f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d$</td>
<td>$a$</td>
<td>$b$</td>
</tr>
<tr>
<td>$\neg d$</td>
<td>$c$</td>
<td>$d$</td>
</tr>
</tbody>
</table>
X² Statistic

- **X² statistic** identifies features which occur significantly more often on day d than at other times covered by the collection

\[
X^2 = \frac{N(ad - bc)^2}{(a + b)(a + c)(b + c)(b + d)}
\]

- Keep days with X² score above threshold and **coalesce ranges** of days allowing for a gap of at most one days in between

- Determine subrange with highest X² score
8.5. Interesting Phrases

- Bedathur et al. [2] consider the problem of identifying interesting phrases that are descriptive for a given query result D’.

- Phrase $p$ is considered interesting if it occurs more often in documents from $D’$ than in the general document collection $D$.

  $$I(p, D’) = \frac{df(p, D’)}{df(p, D)}$$

- Phrase $p$ is only considered if it
  - occurs at least $\sigma$ times in the document collection (e.g., set as 10)
  - has length of at most $\lambda$ (e.g., set as 5)
How to Identify Interesting Phrases Efficiently?

- **Forward index** maintains a representation of every document
  
  - $d_{12}$ ➔ representation of $d_{12}$'s content
  - $d_{37}$ ➔ representation of $d_{37}$'s content
  - $d_{42}$ ➔ representation of $d_{42}$'s content

- **Phrase dictionary** keeps frequency $df(p, D)$ for every phrase $p$

- **High-level algorithm** for identifying top-$k$ interesting phrases
  - **access** the forward index for each $d \in D'$
  - **merge** the $|D'|$ document representations
  - **output** the $k$ most interesting phrases

- **Different document representations** differ in terms of efficiency
Document Content

- **Idea:** Represent document content explicitly as a **sequence of terms** (or compressed term identifiers)

  - $d_{12} \rightarrow \langle axzblkaqx \rangle$
  - $d_{37} \rightarrow \langle zxzdelseqx \rangle$
  - $d_{42} \rightarrow \langle kxzdaqay \rangle$

- **Benefit:**
  - space efficient

- **Drawbacks:**
  - requires **enumeration of all phrases** in document including globally infrequent ones that occur less than $\sigma$ times in $D$
  - requires **phrase dictionary**
Phrases

- **Idea**: Keep all **globally frequent phrases** contained in document $d$ in a consistent (e.g., lexicographic) order

  $d_{12} \rightarrow \langle a \rangle \langle a x \rangle \langle a x z \rangle \langle b \rangle \langle b l \rangle \ldots$
  $d_{37} \rightarrow \langle d \rangle \langle d l \rangle \langle d l e \rangle \langle e \rangle \langle e s \rangle \ldots$
  $d_{42} \rightarrow \langle a \rangle \langle a k \rangle \langle a y \rangle \langle d \rangle \langle d a \rangle \ldots$

- **Benefits**:
  - considers **only globally frequent phrases**
  - **consistent order** allows for **efficient merging**

- **Drawbacks**:
  - **space inefficient**
  - requires **phrase dictionary**
## Frequency-Ordered Phrases

**Idea**: Keep all **globally frequent phrases** contained in document $d$ in ascending order of their embedded global frequency.

<table>
<thead>
<tr>
<th>Document</th>
<th>Phrases</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_{12}$</td>
<td>$5: &lt;xzb&gt; &lt;zb&gt; 6: &lt;q&gt; &lt;x&gt; &lt;xz&gt; 7: &lt;z&gt;...$</td>
</tr>
<tr>
<td>$d_{37}$</td>
<td>$5: &lt;esq&gt; &lt;sqx&gt; 6: &lt;q&gt; &lt;s&gt; &lt;x&gt; &lt;xz&gt;...$</td>
</tr>
<tr>
<td>$d_{42}$</td>
<td>$5: &lt;akqa&gt; &lt;kqa&gt; 6: &lt;q&gt; &lt;x&gt; &lt;xz&gt;...$</td>
</tr>
</tbody>
</table>

**Interestingness of any unseen phrase is upper-bounded** by

$$\min(1, \frac{|D'|}{df(p, D)})$$

where $p$ is the **last phrase encountered**.

---

*Advanced Topics in Information Retrieval / Mining & Organization*
Frequency-Ordered Phrases

- **Idea**: Keep all globally frequent phrases contained in document $d$ in ascending order of their embedded global frequency.

- Interestingness of any unseen phrase is **upper-bounded** by

\[ \min(1, \frac{|D'|}{df(p, D)}) \]

where $p$ is the last phrase encountered.
**Frequency-Ordered Phrases**

- **Idea**: Keep all globally frequent phrases contained in document \(d\) in ascending order of their embedded global frequency.

- **Benefits**:
  - **early termination** possible when no unseen phrase can make it into the top-\(k\) most interesting phrases
  - **self-contained** (i.e., no phrase dictionary needed)

- **Drawbacks**:
  - space inefficient
Prefix-Maximal Phrases

Observation: Globally frequent phrases are often redundant and we do not have to keep all of them.

Definition: A phrase $p$ is prefix-maximal in document $d$ if

- $p$ is globally frequent
- $d$ does not contain another globally frequent phrase $p'$ of which $p$ is a prefix

Prefix-maximal phrase $p$ (e.g., $<a x z>$ in $d_{12}$) represents all its prefixes (i.e., $<a>$ and $<a x>$); they’re guaranteed to be globally frequent and contained in $d$. 

```
d_{12}  →  <a> <a x> <a x z> <b> <b l> ...
d_{37}  →  <d> <d l> <d l e> <e> <e s> ...
d_{42}  →  <a> <a k> <a y> <d> <d a> ...
```
Prefix-Maximal Phrases

- **Idea**: Keep **only prefix-maximal phrases** contained in $d$ in lexicographic order and extract prefixes on-the-fly

  
  
  \[
  \begin{align*}
  d_{12} & \rightarrow \langle axz \rangle \langle bl \rangle \ldots \\
  d_{37} & \rightarrow \langle dle \rangle \langle es \rangle \ldots \\
  d_{42} & \rightarrow \langle ak \rangle \langle ay \rangle \langle da \rangle \ldots
  \end{align*}
  \]

- **Benefits**:
  - space efficient

- **Drawbacks**:
  - extraction of prefixes entails additional bookkeeping
  - requires phrase dictionary
**Experiments**

- **Dataset:** The New York Times Annotated Corpus consisting of 1.8 million newspaper articles published in 1987–2007

- $k = 100$
- $\tau = 10$

- Wallclock Time (ms):
  - 85,575 ms
  - 14,779 ms
  - 3,500 ms
  - 1,030 ms
Anecdotal Results

- **Query: john lennon**
  1) …since john lennon was assassinated…
  2) …lennon’s childhood…
  3) …post beatles work…

- **Query: bob marley**
  1) …music of bob marley…
  2) …marley the jamaican musician…
  3) …i shot the sheriff…

- **Query: john mccain**
  1) …to beat al gore like…
  2) …2000 campaign in arizona…
  3) …the senior senator from virginia…
Summary

- **Clustering** groups similar documents; *k*-Means can be implemented efficiently by leveraging established IR methods.

- **Faceted search** uses orthogonal sets of categories to allow users to explore/navigate a set of documents (e.g., query results).

- **Memes can be tracked** and allow for insightful analyses of media attention and time lag between traditional media and blogs.

- **Timelines** identify significant time-varying features in a set of documents (e.g., query results) and visualize them.

- **Interesting phrases** provide insights into query results; they can be determined efficiently by using a suitable index organization.
References


