8. Mining & Organization

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

Outline

8.1. Clustering

8.2. Faceted Search

8.3. Tracking Memes

8.4. Timelines

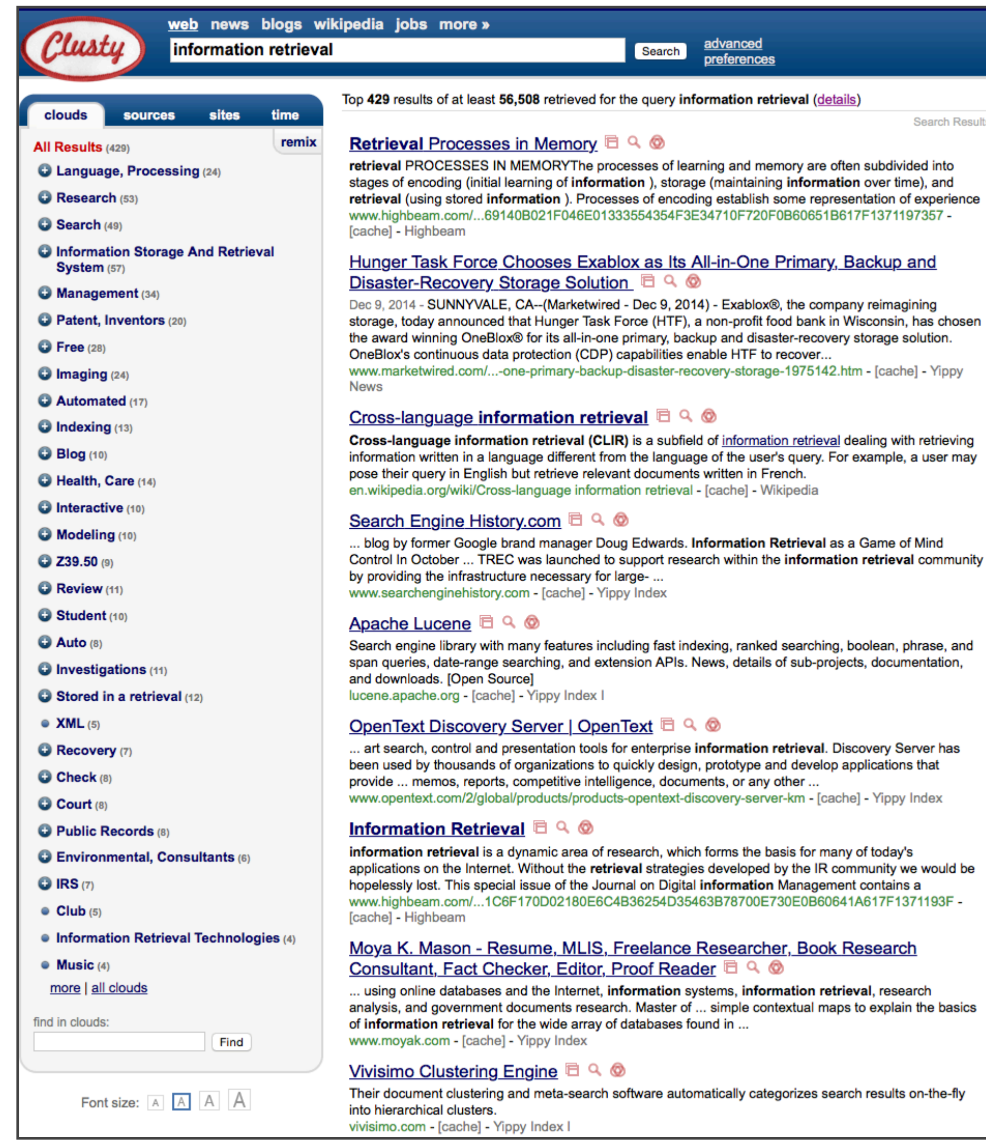
8.5. Interesting Phrases

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** sim(c,d) between document vectors c and d

- **Clusters** C_i represented by a cluster centroid document vector $\mathbf{c}_i$

- **k-Means** groups documents into **k** clusters, maximizing the average similarity between documents and their cluster centroid

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

- 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 $\mathbf{c}_1, \ldots, \mathbf{c}_k$ (e.g., as random documents)
  - **while** not converged (i.e., cluster assignments unchanged)
    - **for** every document **d**, determine **most similar** $\mathbf{c}_i$, and **assign** it to C_i
    - recompute **ci** 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-/ 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)

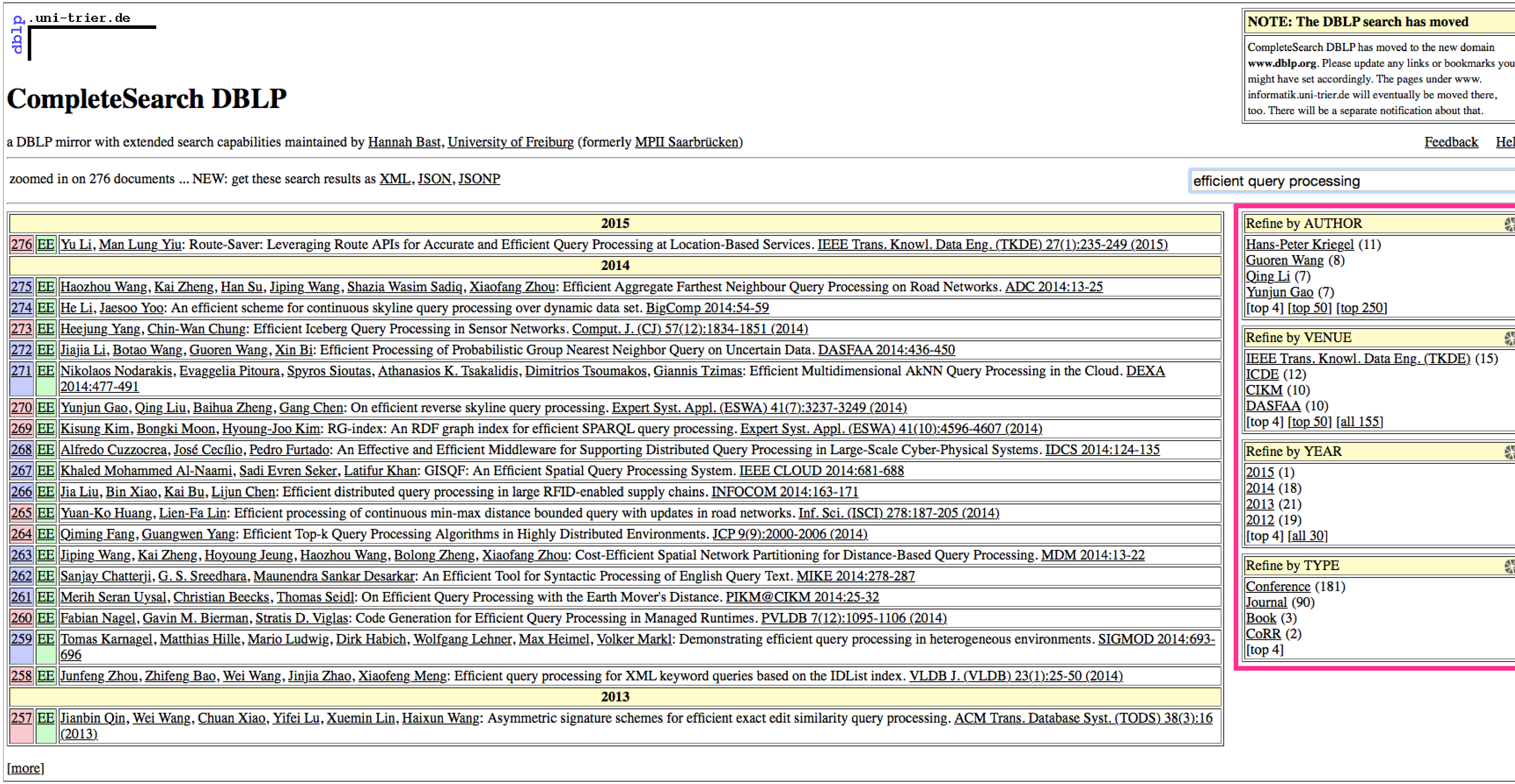
System	ℓ	Dataset 1 Similarity	Dataset 1 Time	Dataset 2 Similarity	Dataset 2 Time
`k-means`	—	0.7804	445.05	0.2856	705.21
`wand-k-means`	100	0.7810	83.54	0.2858	324.78
`wand-k-means`	10	0.7811	75.88	0.2856	243.9
`wand-k-means`	1	0.7813	61.17	0.2709	100.84

System	p	ℓ	Dataset 1 Similarity	Dataset 1 Time	ℓ	Dataset 2 Similarity	Dataset 2 Time
`k-means`	—	—	0.7804	445.05	—	0.2858	705.21
`wand-k-means`	—	1	0.7813	61.17	10	0.2856	243.91
`wand-k-means`	500	1	0.7817	8.83	10	0.2704	4.00
`wand-k-means`	200	1	0.7814	6.18	10	0.2855	2.97
`wand-k-means`	100	1	0.7814	4.72	10	0.2853	1.94
`wand-k-means`	50	1	0.7803	3.90	10	0.2844	1.39

- **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



dblp.uni-trier.de

NOTE: The DBLP search has moved

CompleteSearch DBLP has moved to the new domain **www.dblp.org**. Please update any links or bookmarks you might have set accordingly. The pages under www. informatik.uni-trier.de will eventually be moved there, too. There will be a separate notification about that.

CompleteSearch DBLP

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

Feedback Help

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

efficient query processing

2015

276 EE Yu Li, Man Lung Yiu: Route-Saver: Leveraging Route APIs for Accurate and Efficient Query Processing at Location-Based Services. IEEE Trans. Knowl. Data Eng. (TKDE) 27(1):235-249 (2015)

2014

275 EE Haozhou Wang, Kai Zheng, Han Su, Jiping Wang, Shazia Wasim Sadiq, Xiaofang Zhou: Efficient Aggregate Farthest Neighbour Query Processing on Road Networks. ADC 2014:13-25

274 EE He Li, Jaesoo Yoo: An efficient scheme for continuous skyline query processing over dynamic data set. BigComp 2014:54-59

273 EE Heejung Yang, Chin-Wan Chung: Efficient Iceberg Query Processing in Sensor Networks. Comput. J. (CJ) 57(12):1834-1851 (2014)

272 EE Jiajia Li, Botao Wang, Guoren Wang, Xin Bi: Efficient Processing of Probabilistic Group Nearest Neighbor Query on Uncertain Data. DASFAA 2014:436-450

271 EE Nikolaos Nodarakis, Evaggelia Pitoura, Spyros Sioutas, Athanasios K. Tsakalidis, Dimitrios Tsoumakos, Giannis Tzimas: Efficient Multidimensional AkNN Query Processing in the Cloud. DEXA 2014:477-491

270 EE Yunjun Gao, Qing Liu, Baihua Zheng, Gang Chen: On efficient reverse skyline query processing. Expert Syst. Appl. (ESWA) 41(7):3237-3249 (2014)

269 EE Kisung Kim, Bongki Moon, Hyoung-Joo Kim: RG-index: An RDF graph index for efficient SPARQL query processing. Expert Syst. Appl. (ESWA) 41(10):4596-4607 (2014)

268 EE Alfredo Cuzzocrea, José Cecílio, Pedro Furtado: An Effective and Efficient Middleware for Supporting Distributed Query Processing in Large-Scale Cyber-Physical Systems. IDCS 2014:124-135

267 EE Khaled Mohammed Al-Naami, Sadi Evren Seker, Latifur Khan: GISQF: An Efficient Spatial Query Processing System. IEEE CLOUD 2014:681-688

266 EE Jia Liu, Bin Xiao, Kai Bu, Lijun Chen: Efficient distributed query processing in large RFID-enabled supply chains. INFOCOM 2014:163-171

265 EE Yuan-Ko Huang, Lien-Fa Lin: Efficient processing of continuous min-max distance bounded query with updates in road networks. Inf. Sci. (ISCI) 278:187-205 (2014)

264 EE Qiming Fang, Guangwen Yang: Efficient Top-k Query Processing Algorithms in Highly Distributed Environments. JCP 9(9):2000-2006 (2014)

263 EE Jiping Wang, Kai Zheng, Hoyoung Jeung, Haozhou Wang, Bolong Zheng, Xiaofang Zhou: Cost-Efficient Spatial Network Partitioning for Distance-Based Query Processing. MDM 2014:13-22

262 EE Sanjay Chatterji, G. S. Sreedhara, Maunendra Sankar Desarkar: An Efficient Tool for Syntactic Processing of English Query Text. MIKE 2014:278-287

261 EE Merih Seran Uysal, Christian Beecks, Thomas Seidl: On Efficient Query Processing with the Earth Mover's Distance. PIKM@CIKM 2014:25-32

260 EE Fabian Nagel, Gavin M. Bierman, Stratis D. Viglas: Code Generation for Efficient Query Processing in Managed Runtimes. PVLDB 7(12):1095-1106 (2014)

259 EE Tomas Karnagel, Matthias Hille, Mario Ludwig, Dirk Habich, Wolfgang Lehner, Max Heimel, Volker Markl: Demonstrating efficient query processing in heterogeneous environments. SIGMOD 2014:693-696

258 EE Junfeng Zhou, Zhifeng Bao, Wei Wang, Jinjia Zhao, Xiaofeng Meng: Efficient query processing for XML keyword queries based on the IDList index. VLDB J. (VLDB) 23(1):25-50 (2014)

2013

257 EE Jianbin Qin, Wei Wang, Chuan Xiao, Yifei Lu, Xuemin Lin, Haixun Wang: Asymmetric signature schemes for efficient exact edit similarity query processing. ACM Trans. Database Syst. (TODS) 38(3):16 (2013)

[more]

Refine by AUTHOR

Hans-Peter Kriegel (11)
Guoren Wang (8)
Qing Li (7)
Yunjun Gao (7)
[top 4] [top 50] [top 250]

Refine by VENUE

IEEE Trans. Knowl. Data Eng. (TKDE) (15)
ICDE (12)
CIKM (10)
DASFAA (10)
[top 4] [top 50] [all 155]

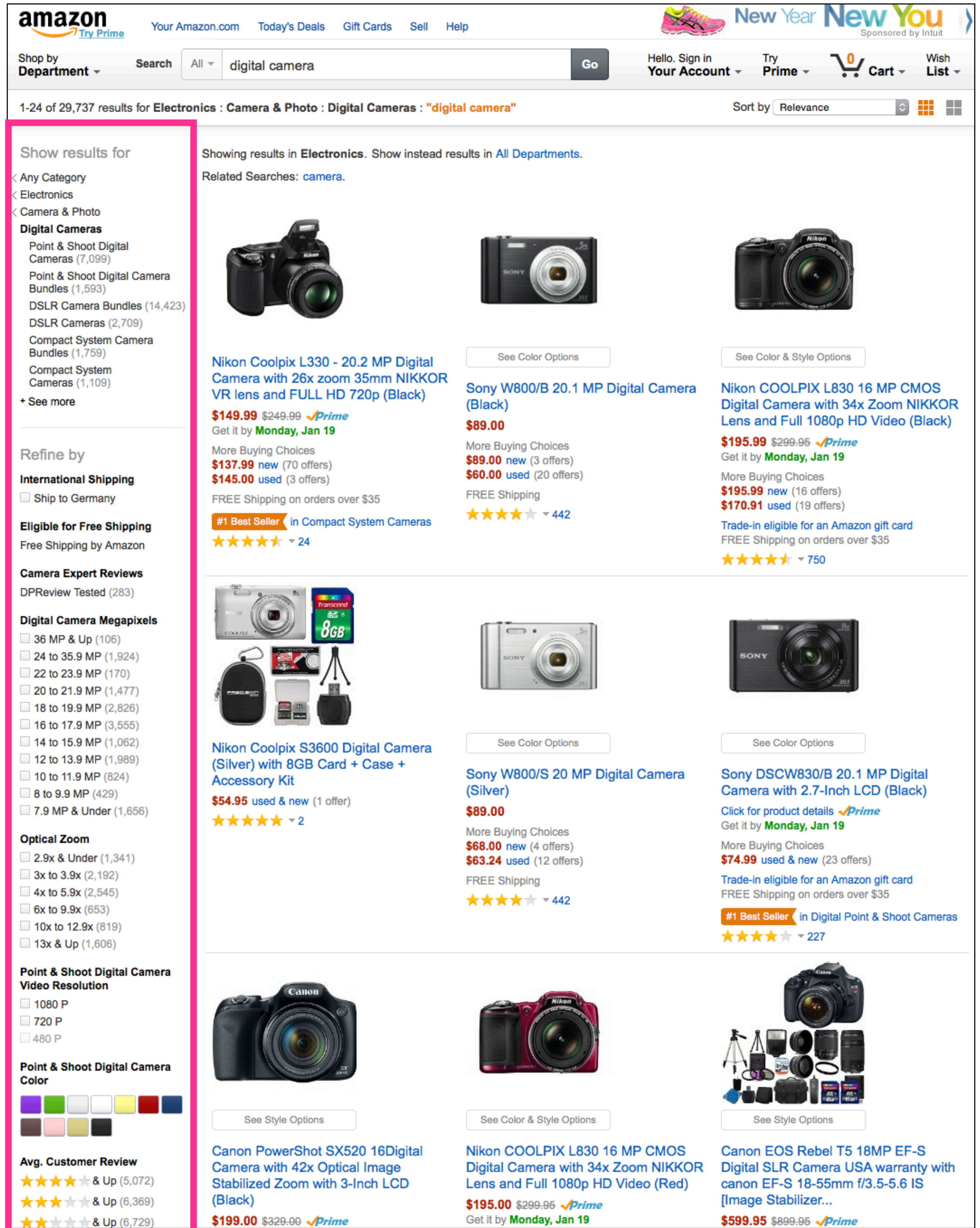
Refine by YEAR

2015 (1)
2014 (18)
2013 (21)
2012 (19)
[top 4] [all 30]

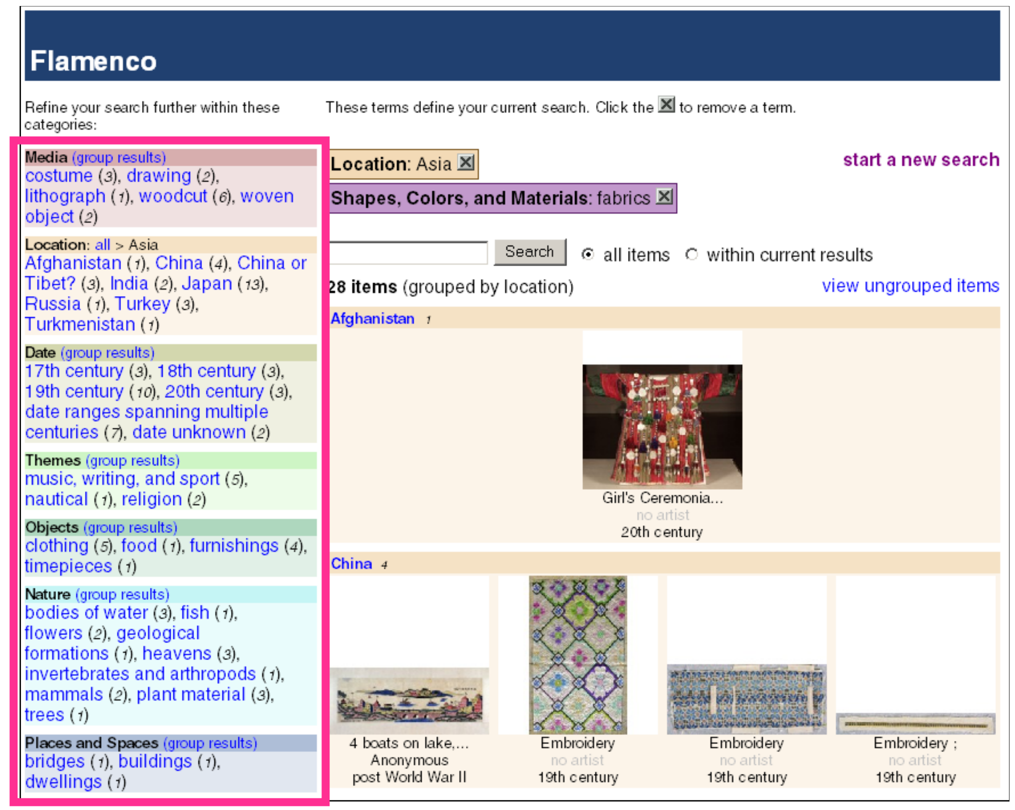
Refine by TYPE

Conference (181)
Journal (90)
Book (3)
CoRR (2)
[top 4]

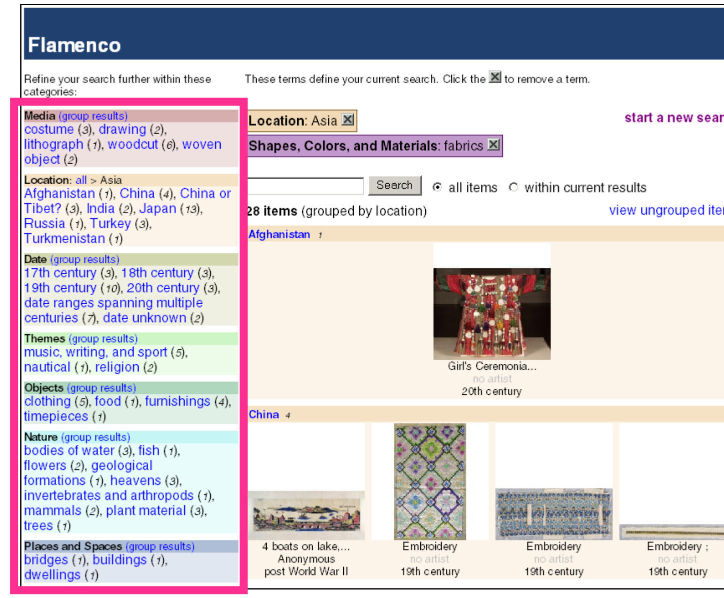
8.2. Faceted Search



8.2. Faceted Search



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.:
    - <u>topic</u>: arts & photography, biographies & memoirs, etc.
    - <u>origin</u>: Europe > France > Provence, Asia > China > Beijing, etc.
    - <u>price</u>: 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

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

- <u>Idea</u>: 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-alphanumeric characters (e.g., brackets), converting them to lower case, and removing items longer than 20 terms

List Weighting

- 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 **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 Dia_{max}
  - save cluster it total weight is larger than W_{min}

Dimension and Item Ranking

- 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, audemars piguet, seiko, accutron, 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, lucia ...
4. what they died for, across the sea, what kate does, the candidate, the last recruit, everybody loves hugo, 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, various, 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, travele, gaming, 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, ...

---

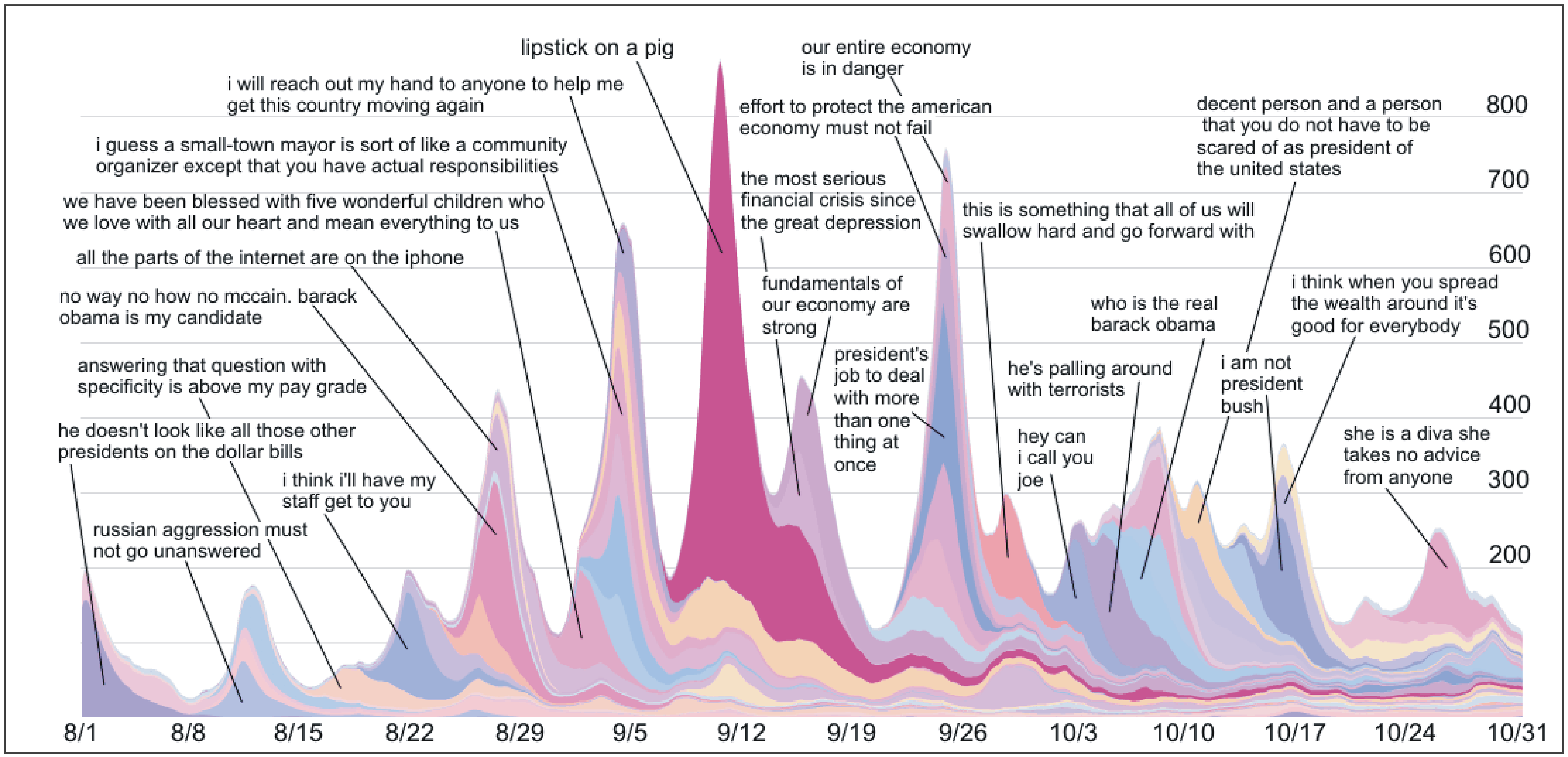
query: **visit beijing**
1. tiananmen square, forbidden city, summer palace, temple of heaven, great wall, beihai park, hutong
2. attractions, shopping, dining, nightlife, tours, travel tip, transportation, facts

---

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

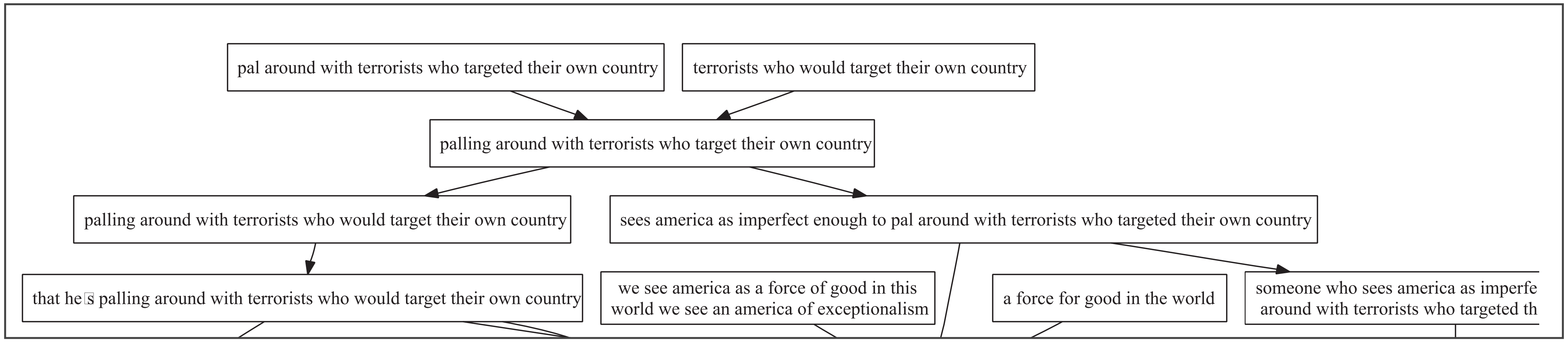


- Demo: http://www.memetracker.org

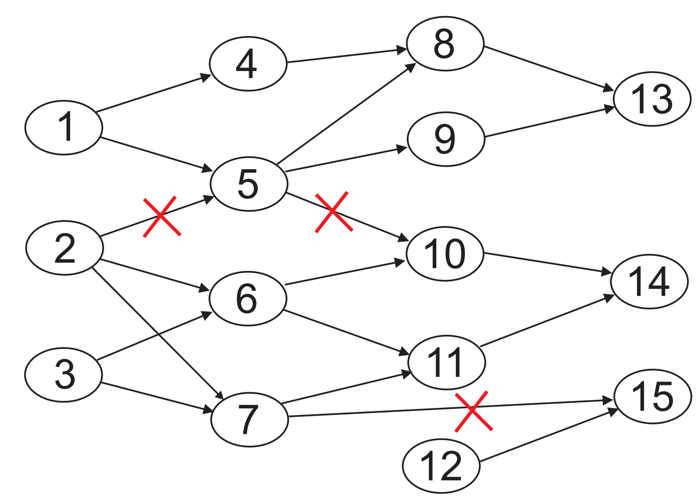
Phrase Graph Construction

- 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 **ε** 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



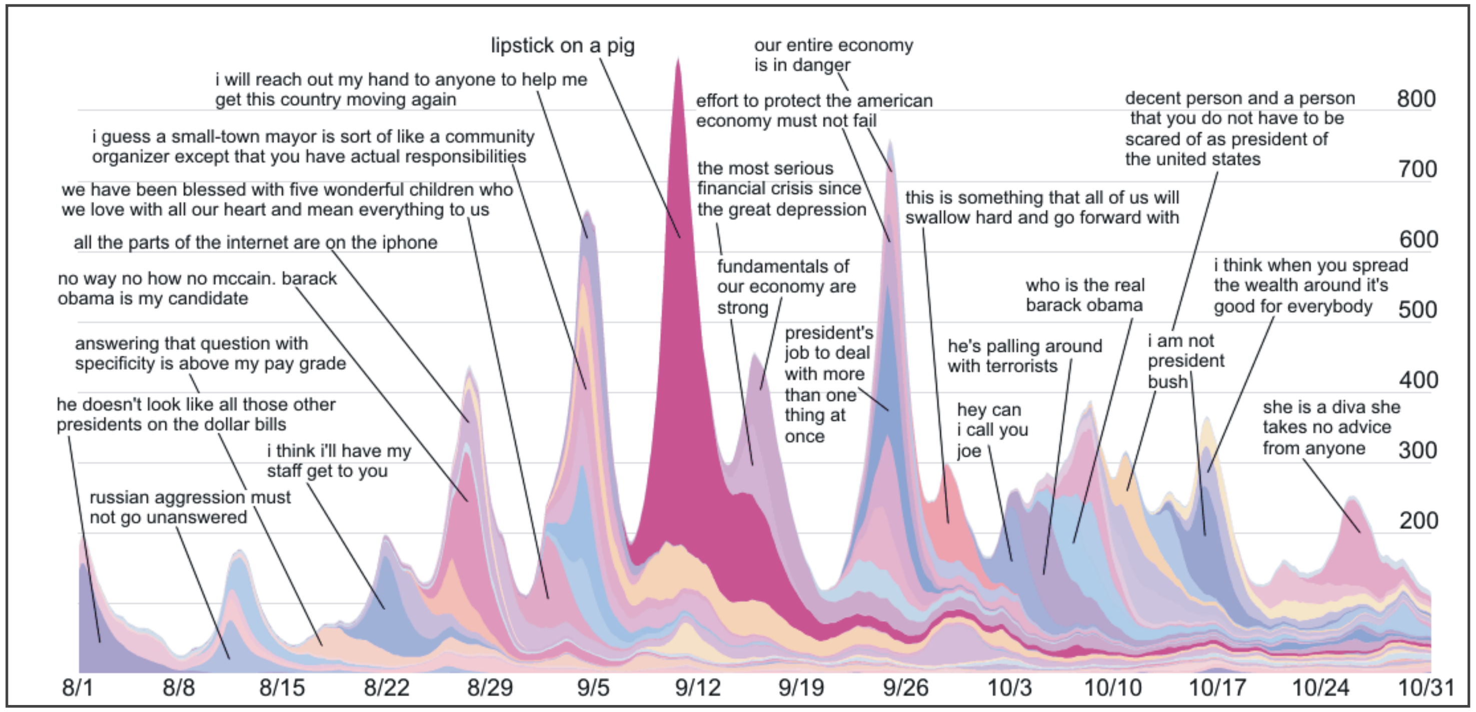
- Phrase graph is an **directed acyclic graph** (DAG) by construction
- Partition G(V, E) by **deleting a set of edges having minimum total weight**, so that each resulting **component is single-rooted**

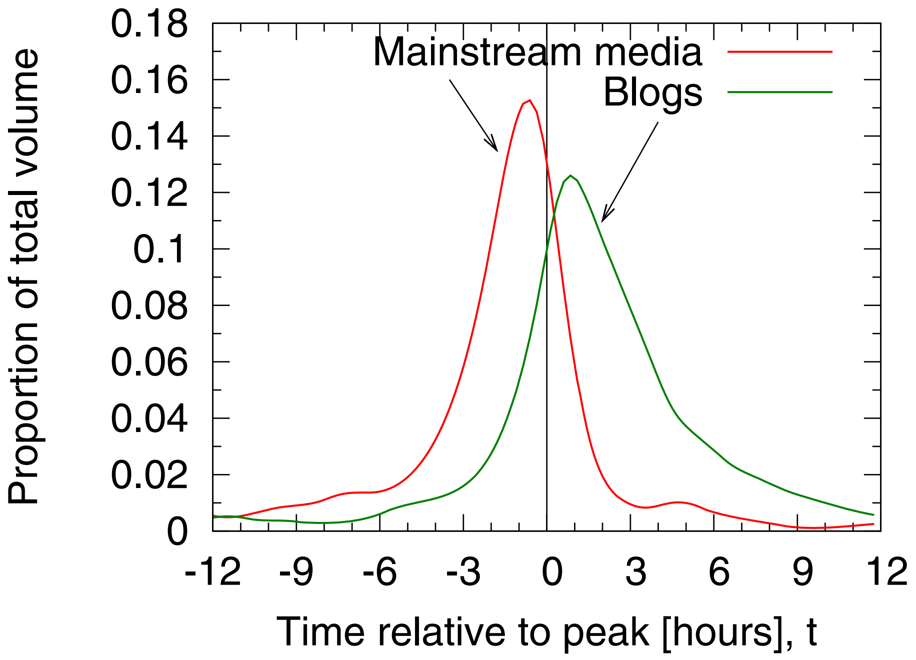


- Phrase graph partitioning is *NP*-hard, hence addressed by **greedy heuristic algorithm**

Applications

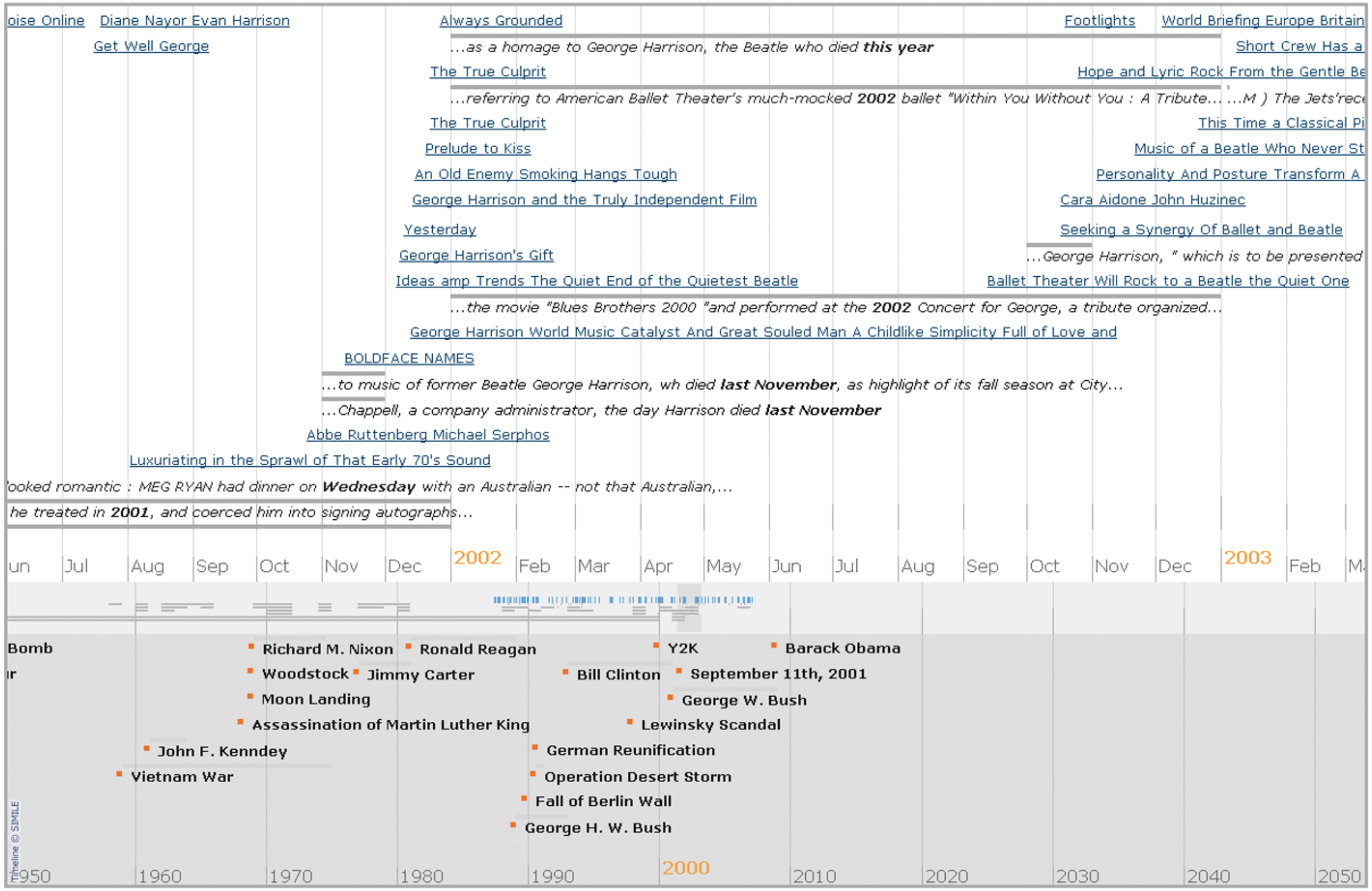
- **Clustering of meme mentions** allows for insightful analyses, e.g.:
  - **volume of meme** per time interval
  - **peek time** of meme in traditional news and social media
  - **time lag** between peek times in traditional news and social media





8.4. Timelines

- **Timelines** visualize, e.g., **major events and topics** and their **occurrence/importance** as they occur in a **collection of timestamped documents**



Timelines

- 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**

Timelines

- Problem: How to identify significantly time-varying features?
- Assume that the following **statistics** have been computed
  - **Nd** as the number of documents in the partition for day **d**
  - **N** as the number of documents in the document collection
  - **fd** 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

	f	**¬f**
d	fd	Nd - fd
¬d	F - fd	N - Nd - F + fd

	f	**¬f**
d	a	b
¬d	c	d

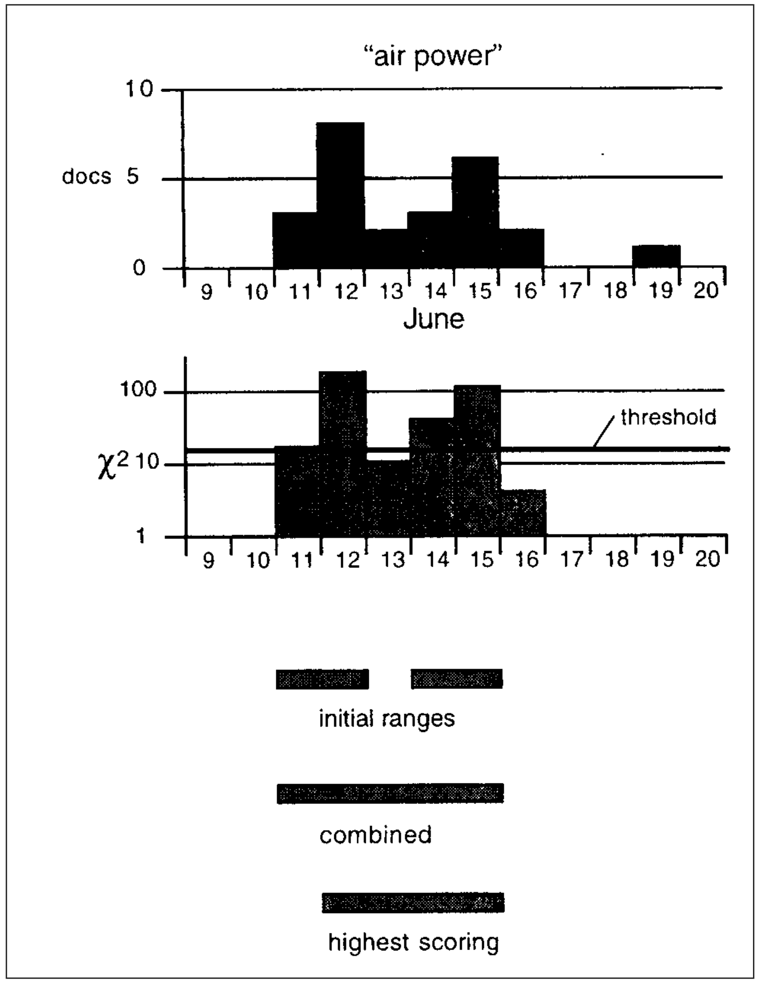
X^2 Statistic

- **X^2 statistic** identifies features **which occur significantly more often on day d** than at other times covered by the collection

$$\chi^2 = \frac{N(ad - bc)^2}{(a+b)(a+c)(b+c)(b+d)}$$

- Keep days with **X^2 score above threshold** and **coalesce ranges** of days allowing for a gap of at most one days in between

- Determine subrange with **highest X^2 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 σ times** in the document collection (e.g., set as 10)
  - **has length of at most λ** (e.g., set as 5)

How to Identify Interesting Phrases Efficiently?

- **Forward index** maintains a representation of every document

$d_{12} \longrightarrow$ representation of $\mathbf{d_{12}}$'s content

$d_{37} \longrightarrow$ representation of $\mathbf{d_{37}}$'s content

$d_{42} \longrightarrow$ representation of $\mathbf{d_{42}}$'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** ∈ **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} → < a x z b l k a q x >
d_{37} → < z x z d l e s q x >
d_{42} → < k x z d a k q a y >

- Benefit:
  - **space efficient**
- Drawbacks:
  - requires **enumeration of all phrases** in document including globally infrequent ones that occur less than σ 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} → < a > < a x > < a x z > < b > < b l > ...

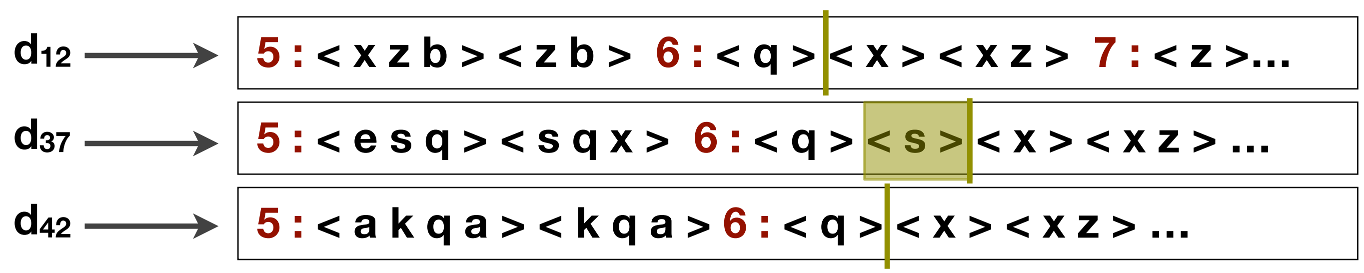
d_{37} → < d > < d l > < d l e > < e > < e s > ...

d_{42} → < a > < a k > < a y > < d > < d a > ...

- 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**



- 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**

d12 → 5 : <x z b> <z b> 6 : <q> <x> <x z> 7 : <z> ...

d37 → 5 : <e s q> <s q x> 6 : <q> <s> <x> <x z> ...

d42 → 5 : <a k q a> <k q a> 6 : <q> <x> <x z> ...

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

$$min(1, \frac{|D'|}{df(p, D)}) \qquad \frac{3}{6}$$

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**

d_{12} → 5 : < x z b > < z b > 6 : < q > < x > < x z > 7 : < z >...

d_{37} → 5 : < e s q > < s q x > 6 : < q > < s > < x > < x z > ...

d_{42} → 5 : < a k q a > < k q a > 6 : < q > < x > < x z > ...

- 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

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 > ...

- 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 $\mathbf{d_{12}}$) **represents all its prefixes** (i.e., **<a>** and **<a x>**); they're guaranteed to be globally frequent and contained in **d**

Prefix-Maximal Phrases

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

d_{12} → < a x z > < b l > ...

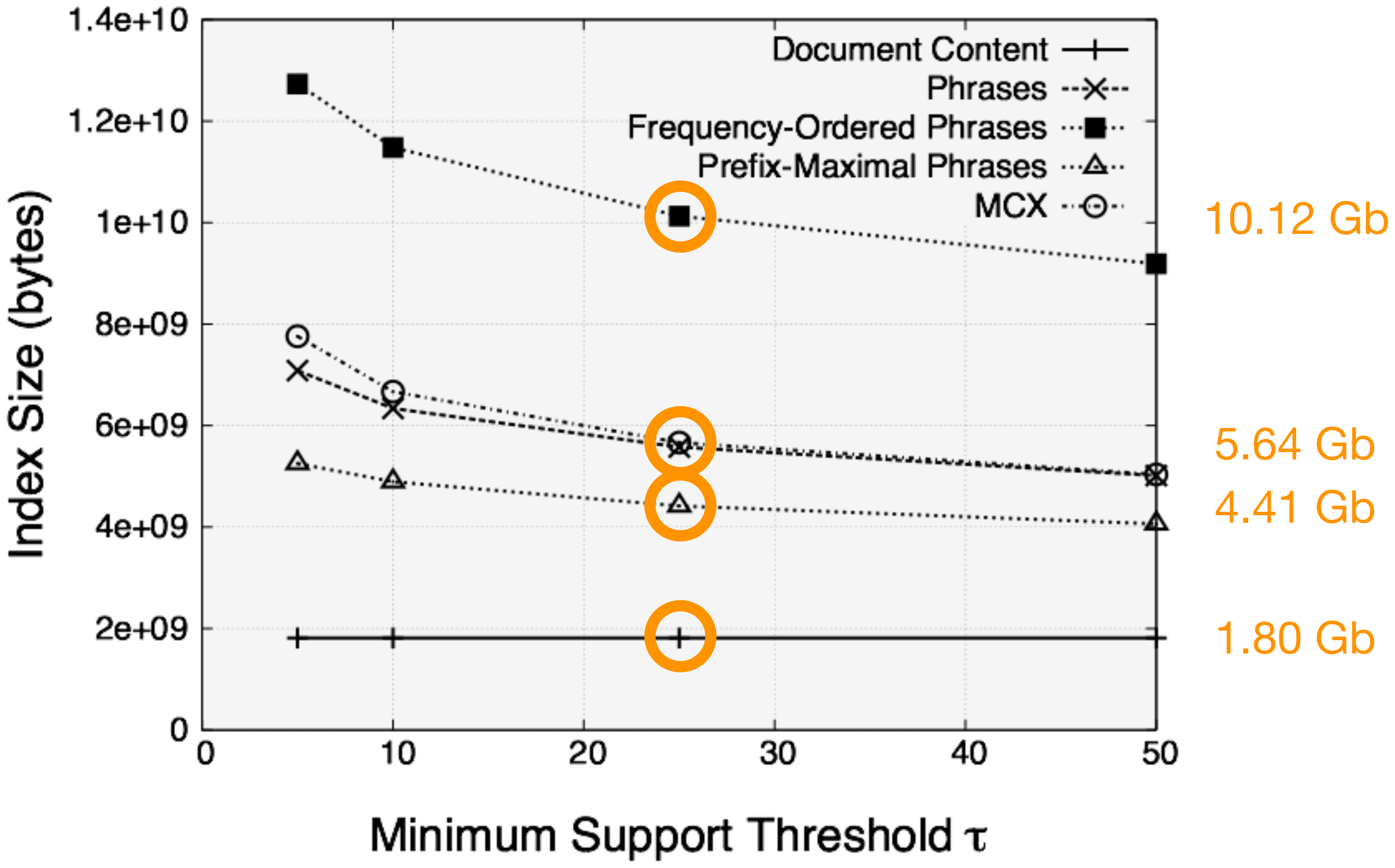
d_{37} → < d l e > < e s > ...

d_{42} → < a k > < a y > < d a > ...

- Benefits:
  - **space efficient**
- Drawbacks:
  - extraction of prefixes entails **additional bookkeeping**
  - requires **phrase dictionary**

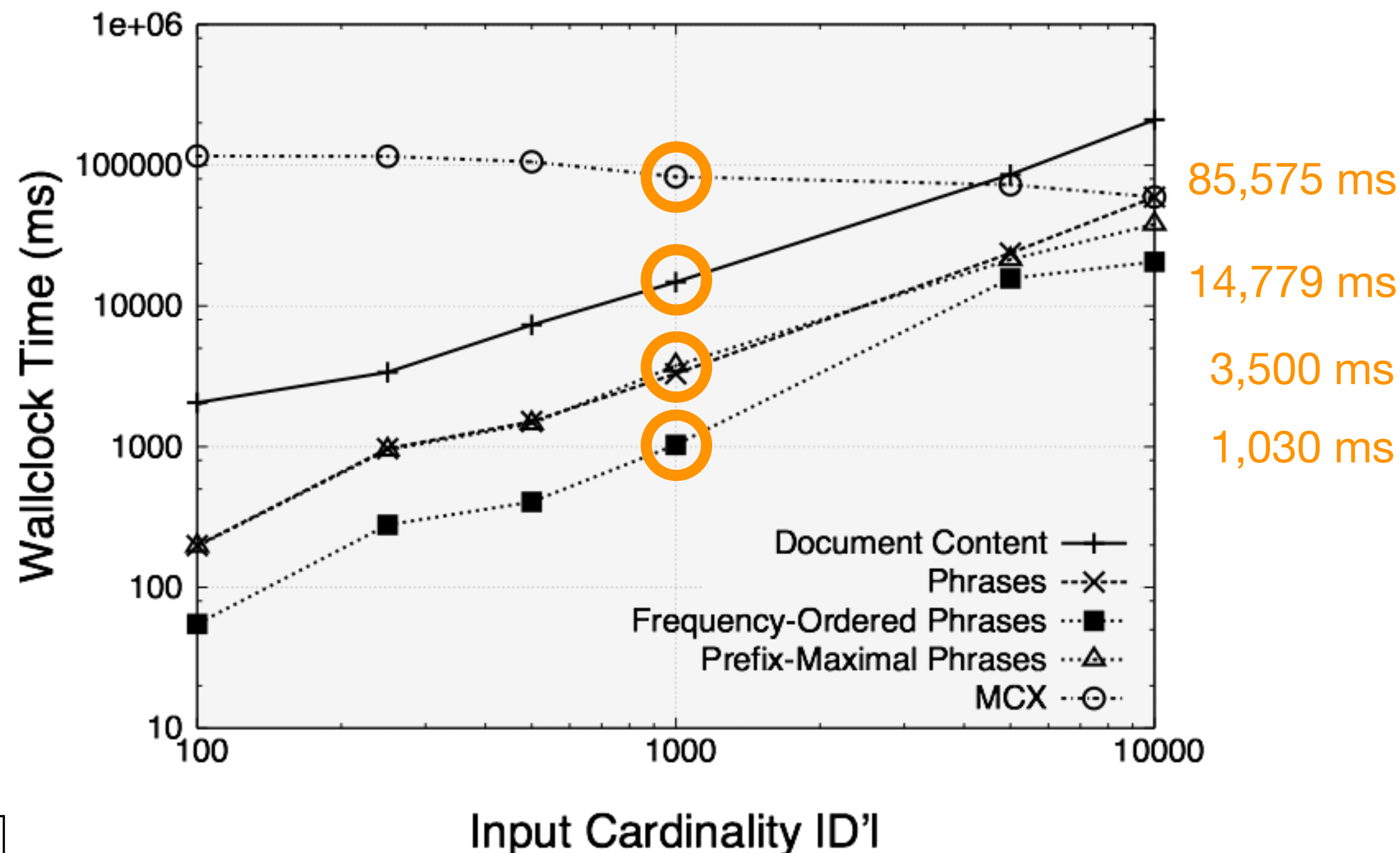
Experiments

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



Experiments

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



k = 100
τ = 10

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

[1] **A. Broder, L. Garcia-Pueyo, V. Josifovski, S. Vassilvitskii, S. Venkatesan:** *Scalable k-Means by Ranked Retrieval*, WSDM 2014

[2] **S. Bedathur, K. Berberich, J. Dittrich, N. Mamoulis, G.:** *Interesting-Phrase Mining for Ad-Hoc Text Analytics*, PVLDB 2010

[3] **Z. Dou, S. Hu, Y. Luo, R. Song, J.-R. Wen:** *Finding Dimensions for Queries*, CIKM 2011

[4] **M. Hearst:** *Clustering Versus Faceted Categories for Information Exploration*, CACM 49(4), 2006

[5] **J. Leskovec, L. Backstrom, J. Kleinberg:** *Meme-tracking and the Dynamics of the News Cycle*, KDD 2009

[6] **R. Swan and J. Allan:** *Automatic Generation of Timelines*, SIGIR 2000

[7] **K.-P. Yee, K. Swearingen, K. Li, M. Hearst:** *Faceted Metadata for Image Search and Browsing*, CHI 2003