Indexing the Past

(1) Indexing and searching methods for versioned document collections

Why Search the Past?

- Historical information needs (an analyst working on the social reactions on the net after 9/11)
- To find relevant resources that do not exist anymore
- To discover trends, opinions, etc. for a certain time period (what people think about UFOs at the beginning of millenium?)

Time Travel?

- Time travel is easy (in IR)! You only need:
  - The data
  - The mechanisms to search
    - indexing + query processing
  - (And fasten your seat belts!)

Data: Preserving the Web

- Non-profit organizations:
  - Internet archive, European archive,...
- Several EU projects
  - Planets, PROTAGE, PrestoPRIME, Scape, ENSURE, BlogForever, LiWA, LAWA, ARCOMEM...

Data

- There are also other "versioned" data collections
  - Wikis (Wikipedia)
  - Repositories (code, document, organization intranets)
  - RSS feeds, news, blogs etc. with continuous updates
  - Personal desktops

Collections including multiple versions of each document with a different timestamp

Time-travel Queries

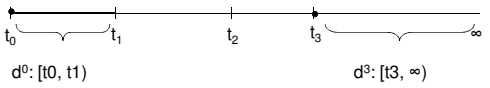
- "Queries that combine the content and temporal predicates" [Berberich et al., SIGIR 2007]
- Interesting query types
  - Point-in-time
    - "euro 2012 articles" @ 1/July/2012
  - Interval
    - "euro 2012 articles" between 01.06-01.07 2012
  - Durability in a time interval [Hou U et al., SIGMOD 2010]:
    - "search engine research papers" that are in top-10 results for 75% of the time between years 2000 and 2012

Indexing

- Various earlier approaches:
  - [Anick et al., SIGIR 1992], [Herscovici et al, ECIR 2007]
- Focus on recent work from two different perspectives
  - Indexing schemes that are more concentrated on the partitioning-related issues
    - [Berberich et al., SIGIR 2007; Anand et al. CIKM 2010; Anand et al. SIGIR 2011]
  - Indexing schemes that are more concentrated on the index size
    - [He et al., CIKM 2009; He et al., CIKM 2010; He et al., SIGIR 2011]

Time-point Queries: Indexing

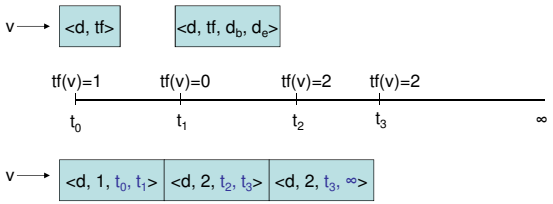
- Formal model:
  - For a document *d*, each version has begin/end timestamps (validity interval):
    - For the current version, $d_e = \infty$
    - For *d*, all validity times are disjoint



[Berberich et al., SIGIR 2007]

Indexing

- Key idea
  - Keep "validity intervals" within posting lists



- Problem: Index size explodes!
  - Even for Wikipedia, $9*10^9$ entries

From [Berberich et al., SIGIR 2007]

Optimization Problem

$$\underset{\mathcal{M}}{argmin} \sum_{[t_i, t_{i+1}) \in \mathcal{E}} P([t_i, t_{i+1})) \cdot PC([t_i, t_{i+1})) \quad \text{s.t.}$$

$$\sum |L_v : m| \leq \kappa |L_v| \,.$$

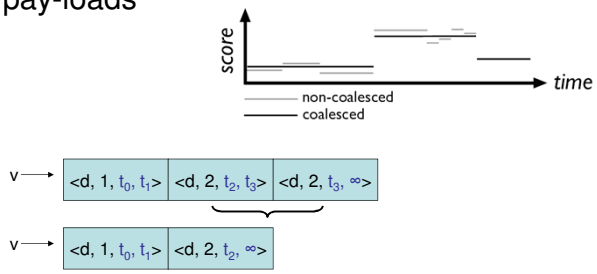
$$\underset{\mathcal{M}}{argmin}\, S(\mathcal{M}) \quad \text{s.t.}$$

$$\forall [t_i, t_{i+1}) \in \mathcal{E} : PC([t_i, t_{i+1})) \leq \gamma \cdot |L_v : [t_i, t_{i+1})|$$

[Berberich et al., SIGIR 2007]

Remedy: Coalescing

- Combine adjacent postings with similar pay-loads



[Berberich et al., SIGIR 2007]

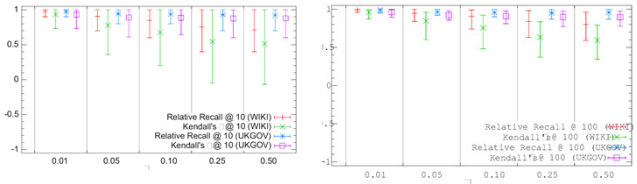
ATC

- Linear-time approximate algorithm

[Berberich et al., SIGIR 2007]

It works!

ϵ	WIKI # Postings	WIKI Ratio	UKGOV # Postings	UKGOV Ratio
-	8,647,996,223	100.00%	7,888,560,482	100.00%
0.00	7,769,776,831	89.84%	2,926,731,708	37.10%
0.01	1,616,014,825	18.69%	744,438,831	9.44%
0.05	556,204,068	6.43%	259,947,199	3.30%
0.10	379,962,802	4.39%	187,387,342	2.38%
0.25	252,581,230	2.92%	158,107,198	2.00%
0.50	203,269,464	2.35%	155,434,617	1.97%



[Berberich et al., SIGIR 2007]

Trade-off

- Optimal approaches
  - Space-optimal
  - Performance-optimal
- Trading off space and performance:
  - Performance-Guarantee apparoach
  - Space-Bound apparoach

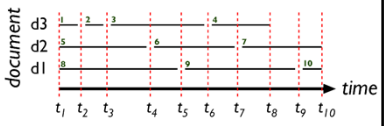
[Berberich et al., SIGIR 2007]

Temporal Partitioning (Slicing)

- Even after colescing, wasted I/O for redundant postings that does not overlap with query's time point

v, $[t_0, t_1)$ → <d, 1, t_0, t_1>

v, $[t_2, \infty)$ → <d, 2, t_2, ∞>



Remark: Validity intervals still reside in the postings!

[Berberich et al., SIGIR 2007]

Computational Complexity

- Performance-Guarantee Approach
  - DP: time and space complexity $O(n^2)$
- Space-Bound Approach
  - DP solution is prohibitively expensive
  - Approximate solution using simulated annealing
    - Time $O(n^2)$, Space $O(n)$
    - Notice *n* is the number of **all unique time interval boundaries** for a given term's posting list

[Berberich et al., SIGIR 2007]

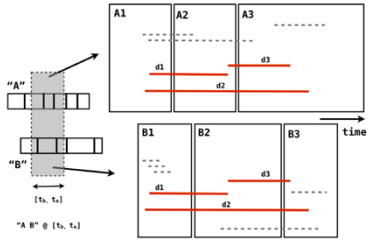
Time-point Queries: Result

- Given a space-bound of 3 (i.e., 3x of the optimal space), close-to-optimal performance is achievable!
  - (Reminder: Partitioning is not very cheap!)

[Berberich et al., SIGIR 2007]

Time-interval Queries?

- When more than one partitions should be accessed, wasted I/O due to repeated postings! (e.g., 3x more postings in SB!)
- Example



[Anand et al., CIKM 2010]

Time-point Queries: Result

- Time-point queries can be handled like this:
- Example

Solution Approaches

Partition selection [Anand et al., CIKM 2010]
Can we avoid accessing all partitions related to a given query?

Document partitioning (sharding) [Anand et al., SIGIR 2011]
Can we partition postings in a list in a different way
i.e., other than using time information?

Partition Selection

Optimization criterion: maximize the fraction of original query results (relative recall).

- Problem: Optimize result quality by accessing a subset of "affected partitions" without exceeding a given I/O upper-bound

Two types of constraints:
a) Size-based: allowed to read up to a fixed number of postings (focus on sequential access)

b) Equi-cost: allowed to read up to a fixed number of partitions (focus on random access)

[Anand et al., CIKM 2010]

Single-term queries

- Size-based partition selection

$$\max |\bigcup_{j,x_j \neq 0} \mathcal{P}_{q,j}| \quad \text{s.t.}$$

$$\sum_j x_j \cdot |\mathcal{P}_{q,j}| \leq \beta \cdot (\sum_j |\mathcal{P}_{q,j}|) \qquad x_j \in \{0,1\}$$

- Equi-cost partition selection $\sum_j x_j \leq \beta \cdot N$
- DP solutions exist, but expensive!
- Approximation
  - Reduce the problem to budgeted max coverage

[Anand et al., CIKM 2010]

Assumption

- Assume an oracle exists "providing us the cardinalities of the individual partitions as well as the result of their intersection/union"
- Later, KMV synopses will be employed as an approximation

[Anand et al., CIKM 2010]

GreedySelect Algorithm

- Cost(P):
  - Size-based: no. of postings in the partition
  - Equi-cost: 1
- Benefit(P):
  - no of **unread** postings in P
- At each step:
  - Select the partition with highest B(P) / C(P)
  - Update benefits of the unselected partitions

Recall: we assume an oracle providing these numbers!

[Anand et al., CIKM 2010]

Example

Simple Math does the job!

- Compute "union of intersections"
- **x**: a tuple of intersection from each term
- **x** is a tuple from the Cartesian product of partitions for each query term
  - $X = P_{term1} \times P_{term2} \times ..... \times P_{termk}$ and then,
  - $\mathbf{x} = \{(P_{term1,i}, P_{term2,j}, ..., \times P_{termk,l})\}$

Choose a partition from this term's partitions

[Anand et al., CIKM 2010]

Multi-term Queries

- Conjunctive semantics →intersect partitions of query terms
- Optimization objective: increase the coverage of postings that are in this intersection space of partitions

$$\max | \bigcap_{1 \le j \le m} \bigcup_{i, x_{ij} \neq 0} \mathcal{P}_{i,j} | \quad \text{s.t.}$$

$$\sum_i \sum_j x_{ij} \cdot |\mathcal{P}_{q,j}| \le \beta_e \cdot (\sum_i \sum_j |\mathcal{P}_{q,j}|)$$

Size-based

$$\sum_i \sum_i x_{ij} \le \beta \cdot N$$

Equi-cost

$$x_{ij} \in \{0, 1\}$$

[Anand et al., CIKM 2010]

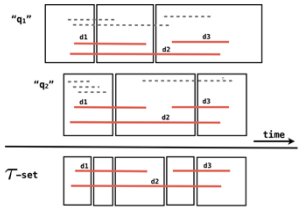
GreedySelect, again!

- Apply GreedySelect over X to pick $\mathbf{x} \in X$
- C(**x**)
  - Sum of the sizes of unselected partitions in **x**, or, number of unselected partitions in **x**
- B(**x**)
  - No of new documents that appear in the intersection of the partitions in x
- Modify algorithm to update benefits and costs after each picking a tuple **x**.

[Anand et al., CIKM 2010]

Cartesian Product or Join?

- Problem: Set *X* might be very large!
- Remedy: Observe that partitions of different terms that have no temporal overlap cannot have any intersecting docs!



Apply the algorithm over the t-join set!

[Anand et al., CIKM 2010]

Performance of Partition Selection

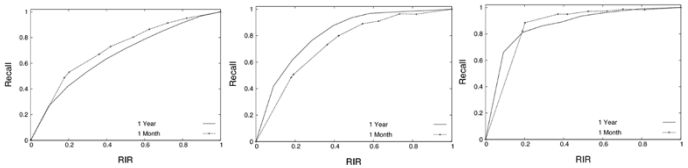
- Compare query run times for:
  - Unpartitioned index,
  - No partition selection,
  - Partition selection with I/O bound (all index files are compressed!)

Bound	WIKI- Unpartitioned time: 3,217 ms			
	Yearly no-sel: 1,021 ms		Monthly no-sel: 212 ms	
	Recall	Avg. Time (ms)	Recall	Avg. Time (ms)
0.1	0.27	207.6	0.01	5.7
0.2	0.42	367.7	0.49	88.5
0.3	0.53	436.8	0.53	94.3
0.4	0.63	521.0	0.67	134.3
0.5	0.71	594.8	0.73	135.9
0.6	0.78	646.7	0.80	165.2
0.7	0.85	736.3	0.87	165.9
0.8	0.91	798.5	0.91	186.4
0.9	0.97	877.4	0.95	192.0
1	1.00	1,020.8	1.00	212.0

[Anand et al., CIKM 2010]

Performance of Partition Selection

- Reletive recall: 50% of affected partitions might be enough!
  - 3 datasets, Wikipedia seems harder!



Wikipedia UKGOV NY Times

[Anand et al., CIKM 2010]

Real Life: KMV Synopses

- Instead of assuming an "oracle" for cardinality estimation, use KMV synopses

Effective sketches for sets supporting arbitrary multiset operations: ∪, ∩, /

- A KMV synopsis for a multiset *S*:
  - Fix a hash function *h*
  - Apply *h* to each distinct value in S
  - k-smallest of the hashed values form KMV

[Beyer et al., SIGMOD 2007]

[Anand et al., CIKM 2010]

Partition Selection with KMV

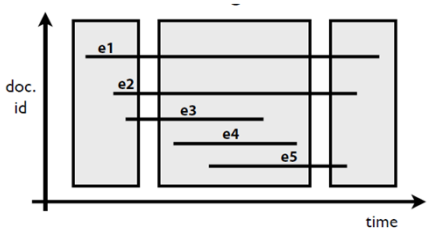
- For each partition of each term:
  - create a flat file storing KMV synopses (5% and 10% samples)

Index	UKGOV	NYT	WIKI
Fixed-7	11G	13G	13G
Syno sis Index - 5% sam le	146MB	134MB	146MB
Syno sis Index - 10% sam le	291MB	258MB	290MB
Fixed-30	4.4G	3.5G	6.3G
Syno sis Index - 5% sam le	61MB	39MB	75MB
Syno sis Index - 10% sam le	122MB	74MB	149MB

[Anand et al., CIKM 2010]

Document partitioning (Sharding)

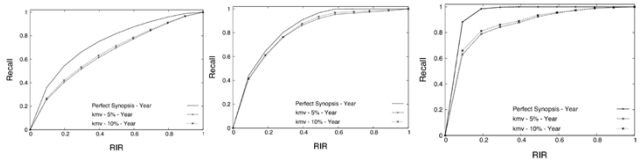
- Another solution to avoid reading and processing repeated postings with cross-cutting validity intervals in temporal partitioning
- Example



[Anand et al., SIGIR 2011]

Partition Selection with KMV

- KMV is promising
  - 5% is enough!

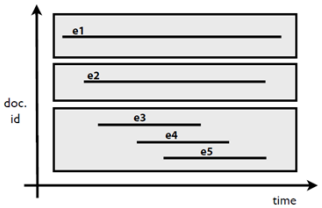


Wikipedia UKGOV NY Times

[Anand et al., CIKM 2010]

Sharding

- Key idea:
  - Instead of partitioning temporally, partition postings based on doc-ids
  - No repetition as in slicing
  - Example



[Anand et al., SIGIR 2011]

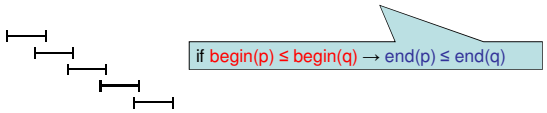
Sharding

- Entries in a shard are ordered according to their begin time
- For each shard, an axuliary data structure:
  - List of *pairs*: (query begin times, offset in shard)
  - Maintain for practical granularities of begin times (like, days) → can fit in memory!
  - For a query with a begin time
    - Seek to the *offset* position in the shard and then read sequentially

[Anand et al., SIGIR 2011]

Idealized Sharding

- For a given posting list L, find a minimal set of shards that satisfy staircase property (SP)

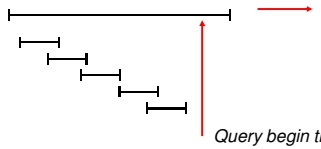


- Greedy solution:
  - For each posting *p* in L (in begin-time order)
    - Add *p* in an exiting shard *s* if SP is satisfied
    - Otherwise, start a new shard with *p*

[Anand et al., SIGIR 2011]

Why several shards?

- There can be still wasted disk I/O while reading a shard:



Earliest posting that **includes** the query begin time!
This causes reading 5 useless postings!

- Problem is the postings with long validity intervals (i.e., subsuming lot of other postings)

[Anand et al., SIGIR 2011]

Merging Shards

- Idealized sharding can yield several shards
  - random access per shard is expensive!

More on this later!

- A greedy merging algortihm
  - Penalty <= $Cost_{Ran}$ /$Cost_{Seq}$
  - Penalty (pairwise): wasted sequential reads for merging two shards
  - Merge in ascending choice order and then smallest size order

[Anand et al., SIGIR 2011]

Performance

- Sharding improves query processing times w.r.t. temporal partitioning or no partitioning et al.
- Merging shards results shorter QP times
- Time measurements are taken on **warm** caches!!!

[Anand et al., SIGIR 2011]

An Alternative Perspective

- Work from Polytechnique Ins. of NYU
- Focus on the index size

Approaches up to now consider each version of a document separately: no special attention on the overlap between versions

A Grand Summary

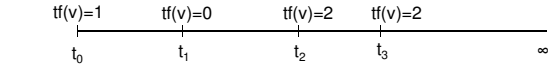
- A full posting list with validity intervals
  - High sequential access + CPU cost
  - 1 random access per query term
- Temporal partitioning (slicing)
  - Reduce seq access (but, repetitions)
  - Reduce CPU cost
  - ≥1 random accesses per query term (with partition selection)
- Document partitioning (sharding)
  - Reduce seq access (no repetitions + time map)
  - Reduce CPU cost (staircase property)
  - ≥1 random accesses per query term (read all shards?!)

TRADE-OFF: May increase random accesses for reducing sequential accesses and CPU cost

Index Compression

- Key ideas
  - Small integers can be represented with smaller codes
  - Doc ids are not so small: instead, compress the gaps between the ids
  - Term frequencies are already small
  - Example

Indexing Versions



v → | $<d_1, 1, t_0, t_1>$ | $<d_1, 2, t_2, t_3>$ | $<d_1, 2, t_3, \infty>$ |

- Assume validity-intervals are stored separately
  → Time-constraints @ post-processing
- Versions of d_i are represented as $d_{i,j}$

v → | $<d_{11}, 1>$ | $<d_{1,2}, 2>$ | $<d_{1,3}, 2>$ |

Question: How can we reduce the storage space for such a representation?

[He et al., CIKM 2009]

How to Exploit?

- Simplest idea:
  - Assign consecutive doc IDs to consecutive versions of the same document
  - Allows small d-gaps for overlapping terms among the versions

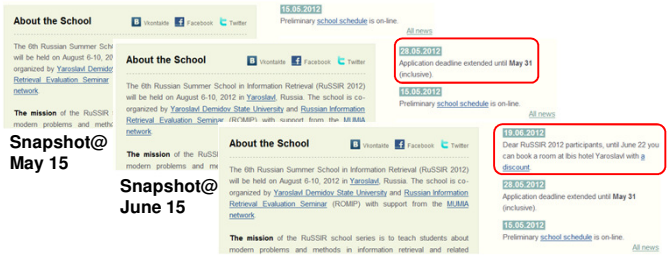
russir →	$<d_{1000}, 1>$	$<d_{3000}, 2>$	$<d_{6000}, 2>$	Random ids
russir →	$<d_1, 1>$	$<d_2, 2>$	$<d_3, 2>$	Consecutive ids

d_1: http://romip.ru/russir2012/, [May 15, June 15]
d_2: http://romip.ru/russir2012/, [June 15, July 15]
d_3: http://romip.ru/russir2012/, [July 15, Aug 15]

[He et al., CIKM 2009]

Versions Are Similar

- There is a high overlap between versions!



MSA Approach

- Multiple Segment Alignment [Herscovici et al., ECIR'07]
  - Given *d* with some versions, a virtual document Vi,j is all terms occuring in (only) versions i through j
  - Reduces the number of postings but increases the document space! (theoretically, up to N^2 virtual documents!)

[He et al., CIKM 2009]

MSA: Example

Herscovici et al., ECIR 2007]

DIFF: Example

- Example:

- QP

Indexing the Differences

- DIFF approach [Anick et al., SIGIR'92]
  - For every pair of consecutive versions d_i and d_{i+1}; create a virtual document that is the symmetric difference between these versions.

[He et al., CIKM 2009]

Performance

- Two index compression schemes
  - Interpolative coding (IPC)
  - PForDelta with optimizations (PFD)
- Two datasets: Wiki, Ireland

Index sizes for doc ids

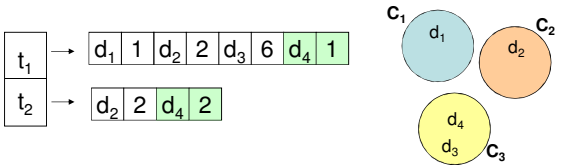
	Wikipedia	**Ireland**
Random-IPC	4957	2908
Random-PFD	5499	3289
Sorted-IPC	570	1086
Sorted-PFD	583	1137
DIFF-IPC	269	751
DIFF-PFD	323	927
MSA-IPC	237	682
MSA-PFD	287	799

Query processing times (in-memory): Sorted is the best!

[He et al., CIKM 2009]

Two-level Indexing: Roots

- Assume we have clusters of documents



- We have queries restricted to clusters:
  - {t1, t2} in C_3 → intersect lists → d2, d4 → d4

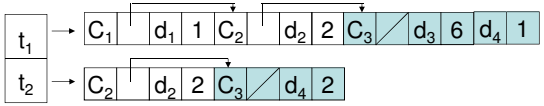
Two-level indexing

- Apply the same idea for versions:
  - **First level index**: document ids
  - **Second level index**: a bitvector for versions

No need for indexing seperate ids for each version → implicit from the bicvector.

[He et al., CIKM 2009]

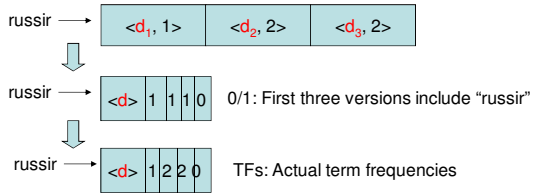
Cluster-skipping index



- Skip redundant postings during QP
  - Reduce decompression cost [Altingovde et al., TOIS 2008]
- Skipping is widely used for QP in practice

[Altingovde et al., TOIS 2008]

Two-level indexing

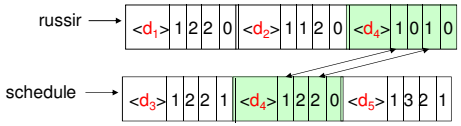


- In actual implementation, group blocks of doc ids and bitvectors of versions
- Bitvectors best compressed by hierarchical **Huffman** coding!

[He et al., CIKM 2009]

Two-level indexing: HUFF

- Query Processing: Decompress bitvectors of documents that are in the **intersection** of query terms



Final result: $\mathbf{d_{4,1}}$ and $\mathbf{d_{4,3}}$

[He et al., CIKM 2009]

Two level indexing: QP

- Queries without temporal constaints

[He et al., SIGIR 2011]

Two level indexing

- Same idea can be applied to MSA and DIFF
  - For virtual documents, create bitvectors as before
  - Compress first level IPC, second level PFD
- Even better compression performance

Index sizes for doc ids

	One Level		Two Level	
	Wiki	Ireland	Wiki	Ireland
DIFF-IPC	269	751	253	652
DIFF-PFD	323	927	269	757
MSA-IPC	237	682	233	663
MSA-PFD	287	799	243	672
HUFF			213	577

[He et al., CIKM 2010]

Two level indexing: QP

- Queries with temporal constaints
  - Post-processing

[He et al., SIGIR 2011]

Two level indexing: QP

- Queries with temporal constaints
  - Partitioning, again!

[He et al., SIGIR 2011]

Grand Summary

Partition-focused	**Size-focused**
– Time information kept in postings	– Time information kept separately
– Redundancy solved by partitioning • Process only relevant partitions	– Redundancy solved by 2-level compression • Process compact blocks • Hierarchical partitions
– Lossy compression of versions (coalescing)	– Lossless compression of versions

QP Performance

[He et al., SIGIR 2011]

References

- **[Altingovde et al., TOIS 2008]** Ismail Sengör Altingövde, Engin Demir, Fazli Can, Özgür Ulusoy: Incremental cluster-based retrieval using compressed cluster-skipping inverted files. ACM Trans. Inf. Syst. 26(3): (2008)
- **[Anand et al., CIKM 2010]** Avishek Anand, Srikanta J. Bedathur, Klaus Berberich, Ralf Schenkel: Efficient temporal keyword search over versioned text. CIKM 2010: 699-708
- **[Anand et al., SIGIR2011]** Avishek Anand, Srikanta J. Bedathur, Klaus Berberich, Ralf Schenkel: Temporal index sharding for space-time efficiency in archive search. SIGIR 2011: 545-554
- **[Anick et al., SIGIR 1992]** Peter G. Anick, Rex A. Flynn: Versioning a Full-text Information Retrieval System. SIGIR 1992: 98-111
- **[Berberich et al., SIGIR 2007]** Klaus Berberich, Srikanta J. Bedathur, Thomas Neumann, Gerhard Weikum: A time machine for text search. SIGIR 2007: 519-526
- **[Beyer et al., SIGMOD 2007]** Kevin S. Beyer, Peter J. Haas, Berthold Reinwald, Yannis Sismanis, Rainer Gemulla: On synopses for distinct-value estimation under multiset operations. SIGMOD Conference 2007: 199-210
- **[He et al., CIKM 2009]** Jinru He, Hao Yan, Torsten Suel: Compact full-text indexing of versioned document collections. CIKM 2009: 415-424
- **[He et al., CIKM 2010]** Jinru He, Junyuan Zeng, Torsten Suel: Improved index compression techniques for versioned document collections. CIKM 2010: 1239-1248
- **[He et al., SIGIR 2011]** Jinru He, Torsten Suel: Faster temporal range queries over versioned text. SIGIR 2011: 565-574
- **[Herscovici et al, ECIR 2007]** Michael Herscovici, Ronny Lempel, Sivan Yogev: Efficient Indexing of Versioned Document Sequences. ECIR 2007: 76-87
- **[Hou U et al., SIGMOD 2010]** Leong Hou U, Nikos Mamoulis, Klaus Berberich, Srikanta J. Bedathur: Durable top-k search in document archives. SIGMOD Conference 2010: 555-566

Thank you!

Questions???

Searching the past

- Problem statements
  - Time must be **explicitly modeled** in order to increase the effectiveness
  - **Time uncertainty** should be taken into account
    - Two temporal expressions can refer to *the same* time period even though they are *not equally written*
- Example
  - Given the query **"Independence Day 2011"**, a retrieval model relying on term-matching will fail to retrieve documents mentioning "July 4, 2011"

Retrieval and Ranking Models

(1) Searching the past
(2) Searching the future

Current approach

- Previous time-aware ranking methods follow two main approaches
  1. Mixture model: linearly combining *textual* and *temporal* similarity
  2. Probabilistic model: generating a query from the *textual part* and *temporal part* of a document independently

Models of document, query, time

- A **temporal query** *q* is composed of keywords *q_text* and temporal expressions *q_time*.
- A **document** *d* consists of the textual part *d_text*, i.e., a bag of words, and the temporal part *d_time* composed of:
  - Publication time *PubTime(d)* of *d*
  - Temporal expressions *ContentTime(d)* mentioned in *d*
- A model for **time** (content or publication time) is represented as a quadruple: (tb_l, tb_u, te_l, te_u)
  - tb_l and tb_u are the lower and upper bound for the begin boundary
  - te_l and te_u are the lower and upper bound for the end boundary

[Berberich et al., ECIR 2010]

Ranking model (cont')

- Temporal expression $t_q \in q\_time$ is generated independently from each other
  - Using a two-step generative model

$$S''(q_{time}, d_{time}) = \prod_{t_q \in q_{time}} P(t_q|d_{time}) = \prod_{t_q \in q_{time}} \left(\frac{1}{|d_{time}|} \sum_{t_d \in d_{time}} P(t_q|t_d) \right)$$

  - Linear interpolation smoothing is applied to $P(t_q|t_d)$ for an unseen temporal expression t_q in d
  - $P(t_q|t_d)$ will be determined wrt. a ranking method

[Kanhabua et al., SIGIR 2011]

Ranking model

- **Mixture model**: linearly combine textual and temporal similarity

$$S(q, d) = (1 - \alpha) \cdot S'(q_{text}, d_{text}) + \alpha \cdot S''(q_{time}, d_{time})$$

  - α indicates the importance of *S'(q_text, d_text)* and *S''(q_time, d_time)*
    - Scores are normalized before combining
  - *S'(q_text, d_text)* is keyword-based ranking
    - E.g., TF-IDF or language modeling

[Kanhabua et al., SIGIR 2011]

Comparison of time-aware ranking

- Five time-aware ranking methods
  - LMT [Berberich et al., ECIR 2010]
  - LMTU [Berberich et al., ECIR 2010]
  - TS [Kanhabua et al., ECLD 2010]
  - TSU [Kanhabua et al., ECLD 2010]
  - FuzzySet [Kalczynski et al., Inf. Process. 2005]

Method	Time		Uncertainty	
	Publication	*Content*	*Ignore*	*Concern*
LMT	x	√	√	x
LMTU	x	√	x	√
TS	√	x	√	x
TSU	x	√	x	√
FuzzySet	√	x	x	√

[Kanhabua et al., SIGIR 2011]

Discussion

- Experiment
  - NYT Corpus and 40 temporal queries
- Result
  - **TSU** outperforms other methods significantly for most metrics
- Conclusions
  - Although TSU gains the best performance, it is **limited** for a document collection with *no time metadata*
  - LMT, LMTU can be applied to any collection without time metadata, but *extraction* of temporal expressions is **needed**

[Kanhabua et al., SIGIR 2011]

Recorded Future

Drawback: a user must specify a query in advance using "predefined" entities

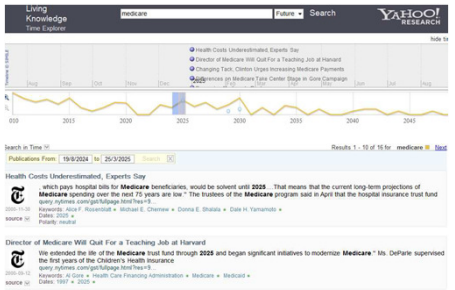
[http://www.recordedfuture.com/]

Searching the future

- Searching the future
  - Extract temporal expressions from news articles
  - Retrieve future information using a probabilistic model, i.e., multiplying term similarity and a time confidence
- Supporting analysis of future-related information in news and Web
  - Extract future mentions from news snippets obtained from search engines
  - Summarize and aggregate results using clustering methods, but no ranking

[Baeza-Yates SIGIR Forum 2005; Jatowt et al., JCDL 2009]

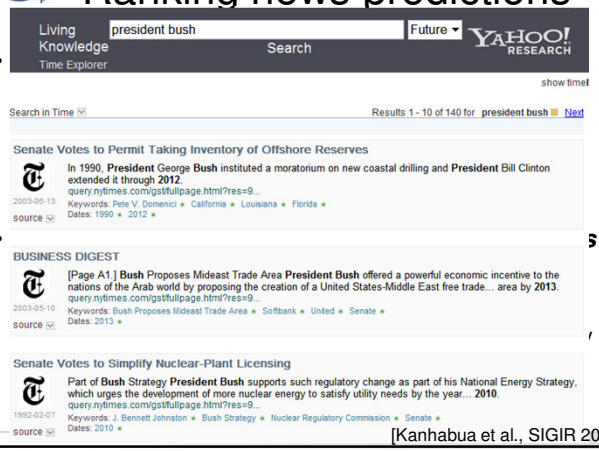
Yahoo! Time Explorer



Drawback: No ranking or performance evaluation is done.

[Matthews et al., HCIR 2010]

Ranking news predictions



[Kanhabua et al., SIGIR 2011a]

Approach

- Four classes of features
  - Term similarity, entity-based similarity, topic similarity and temporal similarity
- Rank predictions using a learning-to-rank technique
- Evaluate using dataset with over 6000 judgments from the NYT Annotated Corpus

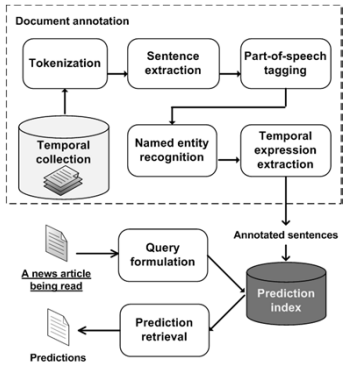
[Kanhabua et al., SIGIR 2011a]

Related news predictions

[Kanhabua et al., SIGIR 2011a]

System architecture

- **Step 1: Document annotation.**
  - Extract temporal expressions using time and event recognition.
  - Normalize them to dates so they can be anchored on a timeline.
  - Output: sentences annotated with named entities and dates, i.e., *predictions.*
- **Step 2: Retrieving predictions.**
  - Automatically generate a query from a news article *being read.*
  - Retrieve predictions that match the query.
  - Rank predictions by relevance. A prediction is "relevant" if it is about the *topics of the article.*



[Kanhabua et al., SIGIR 2011a]

Term similarity

- Capture the term-similarity between *q and p*
  1. *retScore(q,p) Lucene's TF-IDF scoring function*
     - Problem: keyword matching, short texts
     - Predictions not match with query terms
  2. *bm25f(q,p) field-aware ranking function*
     - Extend a sentence by surrounding sentences
     - Search CONTEXT in addition to TEXT

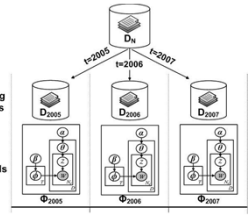
[Kanhabua et al., SIGIR 2011a]

Topic similarity

- Compute the similarity between *q and p on topic*
  - Latent Dirichlet allocation [Blei 2003] for modeling topics
    1. Train a topic model
    2. Infer topics
    3. Compute topic similarity

Step 2: Infer topics.

- Determine topics for *q* and *p* using their contents, called *topic inference*.
- Both *q* and *p* are represented by a probability distribution of topics.
- $p_{\phi} = p(z_1), \ldots, p(z_n)$, where $p(z)$ is a probability of a topic z.



[Kanhabua et al., SIGIR 2011a]

Entity-based similarity

- Measure the similarity between *q and p* using *annotated entities in d_p, p, q*
  - Only applicable for Q_E, Q_C
  - Features commonly employed in *entity ranking*
  - *Time distance* captures the relatedness of term and time

ID	Feature
1	*entitySim*(*q*, *p*)
2	*title*(*e*, d^p)
3	*titleSim*(*e*, d^p)
4	*senPos*(*e*, d^p)
5	*senLen*(*e*, d^p)
6	*cntSenSubj*(*e*, d^p)
7	*cntEvent*(*e*, d^p)
8	*cntFuture*(*e*, d^p)
9	*cntEventSubj*(*e*, d^p)
10	*cntFutureSubj*(*e*, d^p)
11	*timeDistEvent*(*e*, d^p)
12	*timeDistFuture*(*e*, d^p)
13	*tagSim*(*e*, d^p)
14	*isSubj*(*e*, *p*)
15	*timeDist*(*e*, *p*)

[Kanhabua et al., SIGIR 2011a]

Temporal similarity

- **Hypothesis I.** Predictions that are more recent to the query are more relevant

- **Hypothesis II.** Predictions extracted from more recent documents are more relevant

[Kanhabua et al., SIGIR 2011a]

Ranking method

- **Learning-to-rank**: Given an unseen (*q, p*), *p* is ranked using a model trained over a set of labeled query/prediction

$$score(q,p) = \sum_{i=1}^{N} w_i \times f_i$$

  - SVM-*MAP* [Yue et al., SIGIR 2007]
  - RankSVM [Joachims, KDD 2002]
  - SGD-SVM [Zhang, ICML 2004]
  - PegasosSVM [Shalev-Shwartz et al., ICML 2007]
  - PA-Perceptron [Crammer et al., J. Mach. Learn. 2006]

[Kanhabua et al., SIGIR 2011a]

Experiments

- NYT Corpus
  - **More than 25%** contain *at least one prediction*
- Annotation process uses several language processing tools
  - OpenNLP for tokenizing, sentence splitting, part-of-speech tagging, shallow parsing
  - SuperSense tagger for named entity recognition
  - TARSQI for extracting temporal expressions
- Apache Lucene for indexing and retrieving.
  - 44,335,519 sentences and 548,491 predictions
  - 939,455 future dates (avg. future date/prediction is 1.7)

[Kanhabua et al., SIGIR 2011a]

Relevance judgments

- 42 future-related topics

POLITICS	ENVIRONMENT	SPACE
president election	global warming	Mars
Iraq war	energy efficiency	Moon
SCIENCE	**PHYSICS**	**HEALTH**
earthquake	particle Physics	bird flue
tsunami	Big Bang	influenza
BUSINESS	**SPORT**	**TECHNOLOGY**
subprime	Olympics	Internet
financial crisis	World cup	search engine

[Kanhabua et al., SIGIR 2011a]

Discussion

- Results
  - **Topic features** play an important role in ranking
  - Features in top-5 features with *lowest weights* are **entity-based features**
- Open issues
  - Extract predictions from other sources, e.g., Wikipedia, blogs, comments, etc.
  - Sentiment analysis for future-related information

[Kanhabua et al., SIGIR 2011a]

References

- [Baeza-Yates SIGIR Forum 2005] Ricardo A. Baeza-Yates: Searching the future. SIGIR workshop MF/IR 2005
- [Berberich et al., ECIR 2010] Klaus Berberich, Srikanta J. Bedathur, Omar Alonso, Gerhard Weikum: A Language Modeling Approach for Temporal Information Needs. ECIR 2010: 13-25
- [Crammer et al., J. Mach. Learn. 2006] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, Yoram Singer: Online Passive-Aggressive Algorithms. Journal of Machine Learning Research 7: 551-585 (2006)
- [Jatowt et al., JCDL 2009] Adam Jatowt, Kensuke Kanazawa, Satoshi Oyama, Katsumi Tanaka: Supporting analysis of future-related information in news archives and the web. JCDL 2009: 115-124
- [Joachims, KDD 2002] Thorsten Joachims: Optimizing search engines using clickthrough data. KDD 2002: 133-142
- [Kalczynski et al., Inf. Process. 2005] Pawel Jan Kalczynski, Amy Chou: Temporal Document Retrieval Model for business news archives. Inf. Process. Manage. 41(3): 635-650 (2005)
- [Kanhabua et al., SIGIR 2011] Nattiya Kanhabua, Kjetil Nørvåg: A comparison of time-aware ranking methods. SIGIR 2011: 1257-1258
- [Kanhabua et al., SIGIR 2011a] Nattiya Kanhabua, Roi Blanco, Michael Matthews: Ranking related news predictions. SIGIR 2011: 755-764
- [Matthews et al., HCIR 2010] Michael Matthews, Pancho Tolchinsky, Roi Blanco, Jordi Atserias, Peter Mika, Hugo Zaragoza: Searching through time in the new york times. HCIR workshop 2010
- [Shalev-Shwartz et al., ICML 2007] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, Andrew Cotter: Pegasos: primal estimated sub-gradient solver for SVM. Math. Program. 127(1): 3-30 (2011)
- [Yue et al., SIGIR 2007] Yisong Yue, Thomas Finley, Filip Radlinski, Thorsten Joachims: A support vector method for optimizing average precision. SIGIR 2007: 271-278
- [Zhang, ICML 2004] Tong Zhang: Solving large scale linear prediction problems using stochastic gradient descent algorithms. ICML 2004

Web Science – **Investigating the Future** of Information and Communication

Question?