Information Retrieval with Verbose Queries

Manish Gupta — Microsoft

Michael Bendersky — Google

9th Aug 2015

Disclaimer

- This survey is entirely based on previously published research and publicly available datasets, rather than the internal practices of the respective employers of the authors.

- As such, it should prove useful for both practitioners and academic researchers interested in reproducing the reported results

Null Queries

- Null queries= Queries that lead to 0 matches
- Reasons
  - Verbosity
  - Mismatch between searcher and publisher vocabulary
  - Unavailability of documents (temporally, or general rarity)
  - Inability of naïve users to formulate appropriate queries
- We focus on the verbose sub-category of null queries

Long Verbose Queries are Frequent

- Community-based Question Answering
- Literature Search
- Search in Context
- Enterprise or Academic Search
- Voice-activated Search
  - Cortana, Siri, Google Now
- http://www.zdnet.com/blog/micro-markets/yahoo-searches-more-sophisticated-and-specific/27
  - Yahoo! claimed 17% queries contained 5+ words (2006)
- Convey sophisticated information needs
- Cut-and-paste: User finds some text on some topic and fires it as a query.

Search Performance for Verbose Queries

- Search engines perform poorly on verbose queries because
  - High degree of query specificity
  - Term redundancy or extraneous terms (lot of noise)
  - Rarity of verbose queries
  - Lack of sufficient natural language parsing
  - Difficulty in distinguishing between the key and complementary concepts

Scope of the Tutorial

- We will not talk about
  - Automatic Speech Recognition (ASR)
  - Processing null queries other than verbose queries
  - Query by document
  - Query processing tasks for short queries
- We will discuss about
  - Part 1: Query reduction, reformulation and segmentation techniques.
  - Part 2: Query concept weighting, expansion and learning-to-rank.

Datasets and Metrics

- Datasets
  - TREC datasets
  - NTCIR-4/5 English-English ad-hoc IR tasks
  - Real web query logs
  - Document paragraphs and passages

Collection	Content	#Docs	Topics
Robust04	Newswire	528155	250
W10g	Web	1692096	100
GOV2	Web	25205179	150
ClueWeb-09-Cat-B	Web	50220423	150
TREC123	TREC disks 1 and 2	742611	150
CERC	Enterprise Documents from *.csiro.au	370715	50

- Metrics
  - Mean Average Precision (MAP)
  - Mean Reciprocal Rank (MRR)
  - Precision@K
  - NDCG

```
<num> Number 829
<title> Spanish Civil War support
<desc> Provide information on all kinds of material
international support provided to either side in the
Spanish Civil War.
```

Tutorial Overview

- Properties of Verbose Queries [30 min]
- Techniques for Processing Verbose Web Queries
  - Query Reduction to a Single Sub-Query [50 min]
  - Query Reduction by Choosing Multiple Sub-Queries [10 min]
  - Break [30 min]
  - Weighting Query Words [90 min]
  - Lunch Break [90 min]
  - Query Expansion by Including Related Concepts [20 min]
  - Query Reformulation [30 min]
  - Query Segmentation [40 min]
  - Break [30 min]
  - Query-Dependent Learning to Rank [40 min]
- Applications of Verbose Query Processing [40 min]
- Summary and Future Research Directions [10 min]

Examples of Various Techniques

Technique	Original Query	Modified Query
Query Reduction to a Single Sub-query	ideas for breakfast menu for a morning staff meeting	breakfast meeting menu ideas
Query Reduction to a Multiple sub-queries	identify any efforts proposed or undertaken by world governments to seek reduction of iraqs foreign debt	reductions iraqs foreign debt, iraqs foreign debt
Query Weighting	civil war battle reenactments	civil:0.0889, war:0.2795, battle:0.1310, reenactments:0.5006
Query Expansion	staining a new deck	staining a new deck Shopping/Home and Garden/Home Improvement
Query Reformulation	how far is it from Boston to Seattle	distance from Boston to Seattle
Query Segmentation	new ac adapter and battery charger for hp pavilion notebook	new, ac adapter, and, battery charger, for, hp pavilion notebook

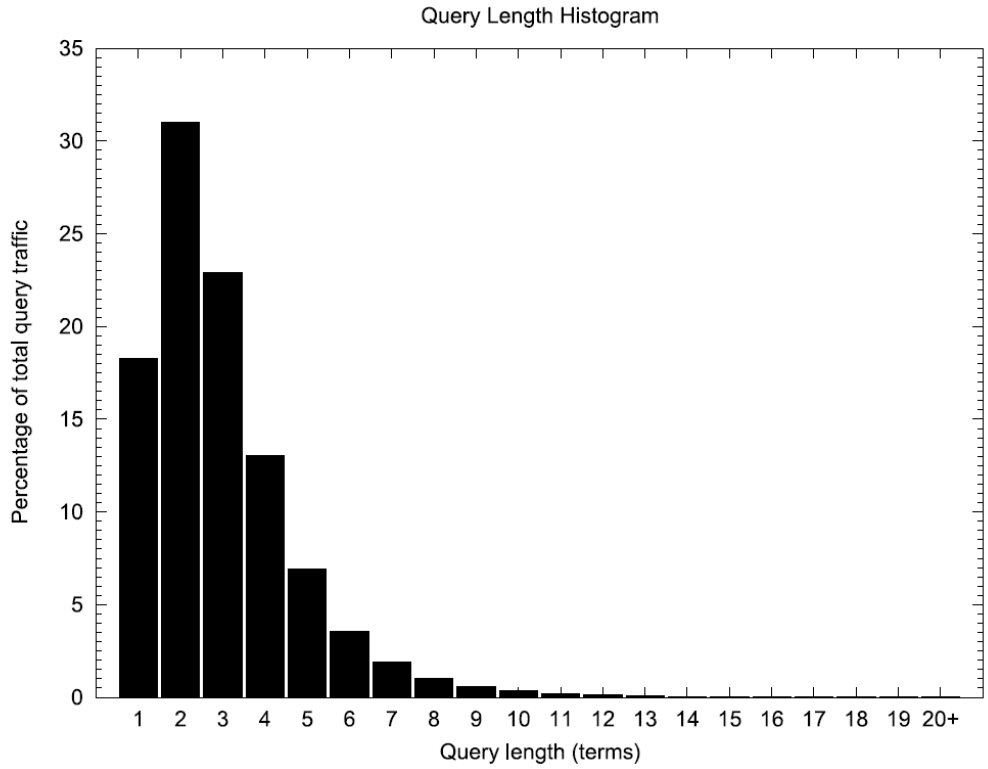
Tutorial Overview

- **Properties of Verbose Queries [30 min]**
- Techniques for Processing Verbose Web Queries
  - Query Reduction to a Single Sub-Query [50 min]
  - Query Reduction by Choosing Multiple Sub-Queries [10 min]
  - Break [30 min]
  - Weighting Query Words [90 min]
  - Lunch Break [90 min]
  - Query Expansion by Including Related Concepts [20 min]
  - Query Reformulation [30 min]
  - Query Segmentation [40 min]
  - Break [30 min]
  - Query-Dependent Learning to Rank [40 min]
- Applications of Verbose Query Processing [40 min]
- Summary and Future Research Directions [10 min]

AK08, BWLK10

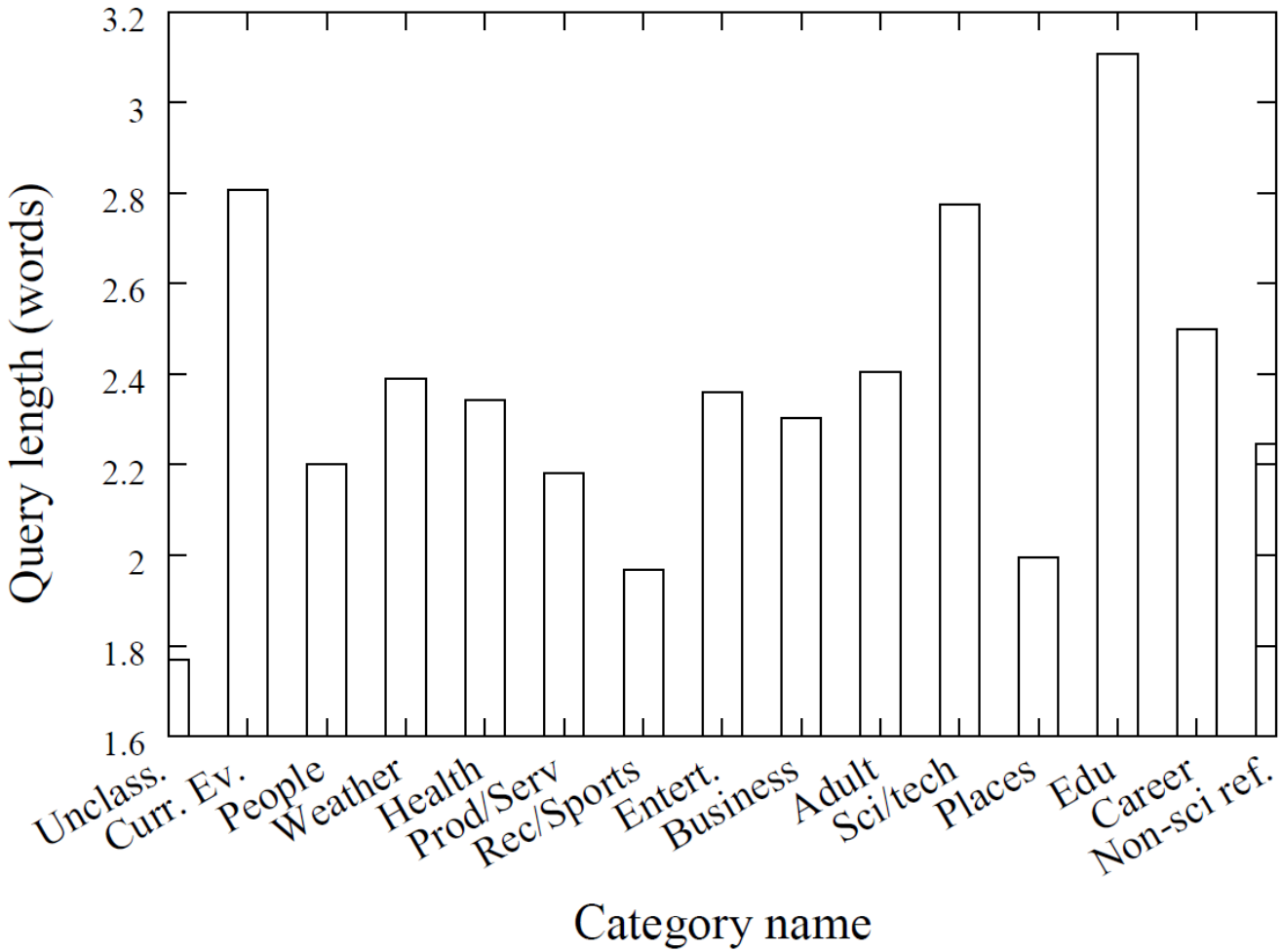
Length Distribution of TREC/Web Queries

- Power law distribution: $p(k) = Ck^{-s}$ for $k \geq k_o$
  - C is normalizing constant, s is slope, k_0 is lower bound from which power law holds



LH99

Query Length wrt Categories



BC09

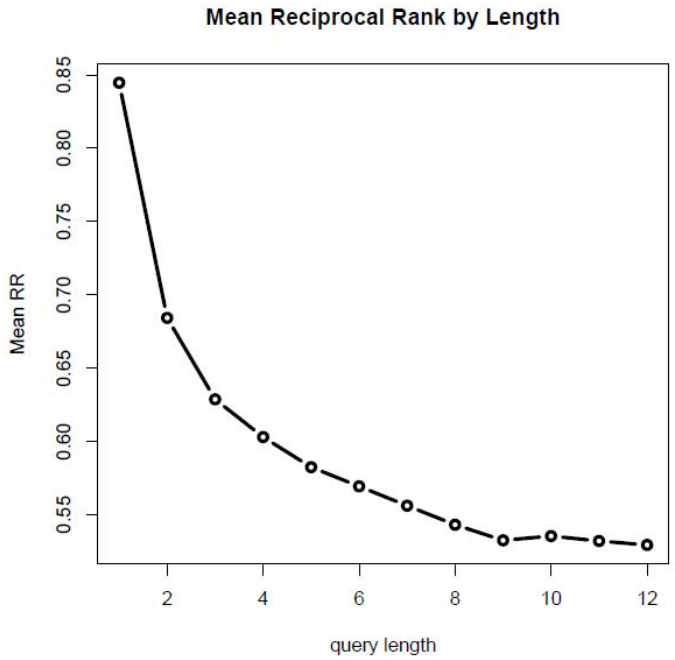
Types of Verbose Queries

Total Queries: $14,921,286$		
Long Queries $(5 \leq l(q) \leq 12)$ **:** $1,423,664$		
Type	Count	% of Long
Questions (QE)	106,587	7.49
Operators (OP)	78,331	5.50
Composite (CO)	910,103	63.93
Noun Phrases (NC_NO)	209,906	14.74
Verb Phrases (NC_VE)	118,736	8.34

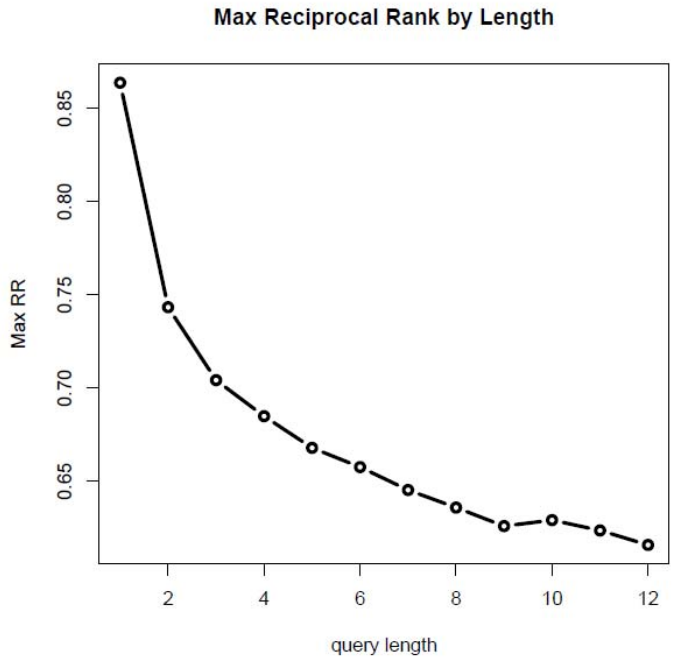
- QE: Queries that begin with {what, who, where, when, …}
- OP: Queries that contain at least one Boolean operator (AND, OR, NOT), one phrase operator (+, “), one special web search operator (contains:, filetype:, intitle:, …)
- CO: Queries which are a composition of short queries (segments)
- NC_NO: Non-composite queries that contain a noun phrase but not a verb phrase
- NC_VE: Non-composite queries that contain a verb phrase

Click Data Analysis

- There is a 29% decrease in the expected reciprocal rank of the first click between the shortest (length = 1) and the longest (length = 12) queries in MSN query log



(a) $meanRR(q)$



(b) $maxRR(q)$

BC09
SPS12

User Abandonment and Impact

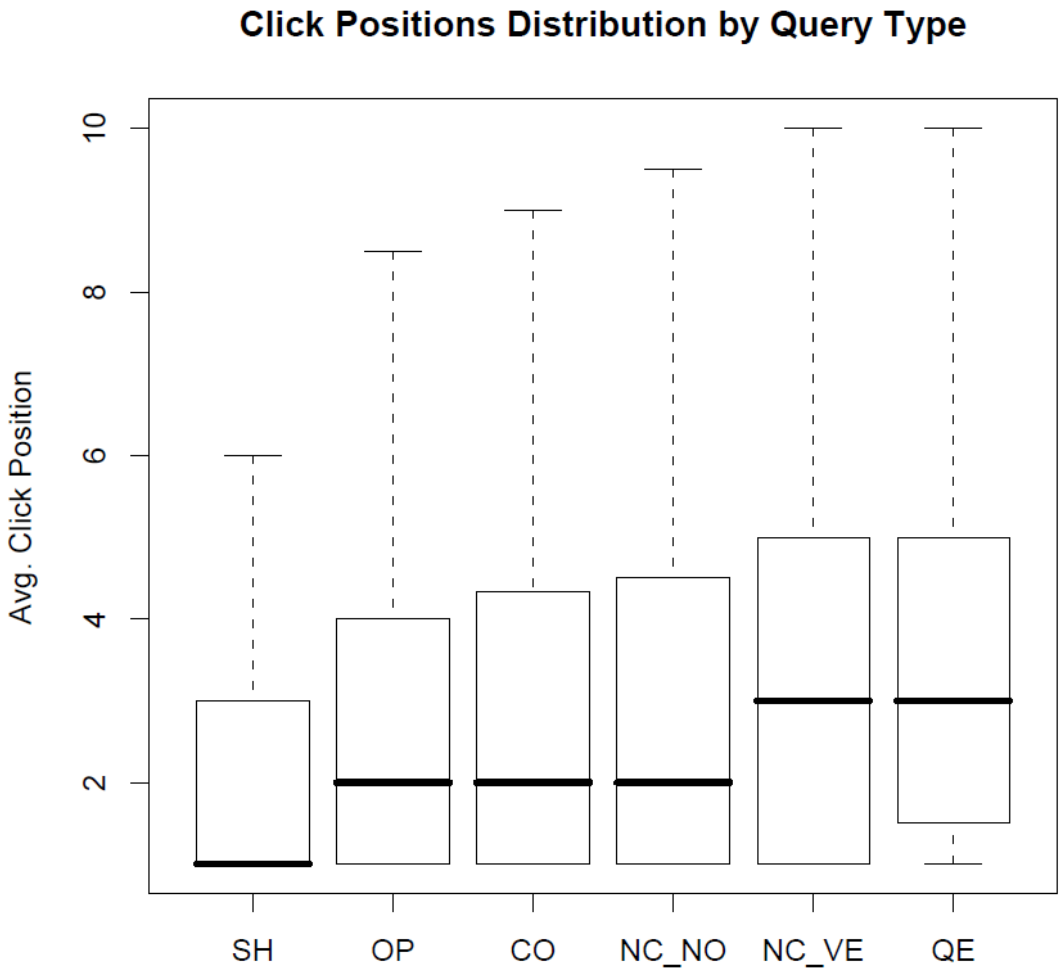
- For a user segment the table compares degradation of purchase rate for zero recall (null) search trails from non-zero recall (non-null) search trails. Purchase rate for all classes of users is lowered when null recall situations are encountered on their trails.

- Abandonment rate trends are generally similar to the reciprocal rank measures.

User segment	$\frac{Purchase\ rate\ for\ ZRST}{Purchase\ rate\ for\ NZRST}$
All users	0.63
Power users	0.68
Novices	0.61

Type	$l(q)$	$abRate(q)$
SH	1.99	0.40
CO	5.67	0.41
OP	6.05	0.58
NC_NO	5.77	0.61
NC_VE	6.35	0.59
QE	6.75	0.51

Average Click Positions for Different Query Types



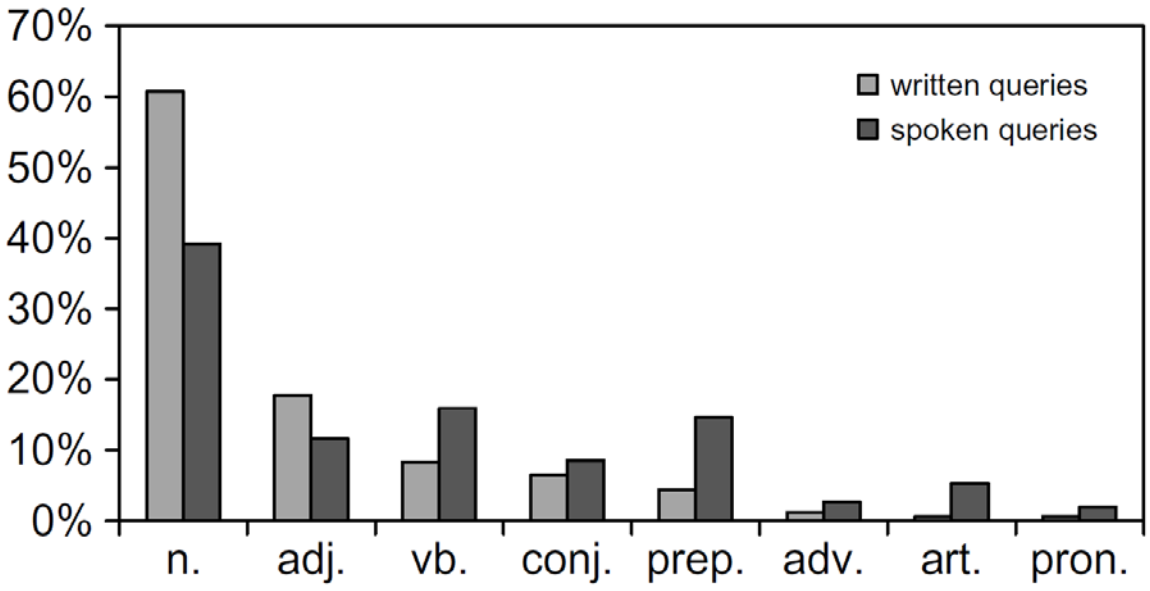
SH=short queries

CD06
YM11

Written vs Spoken Queries

Data set	Written queries	Spoken queries
Number of queries	120	120
Unique terms in queries	309	552
Average query length (with stopwords)	9.54	23.07
Average query length (without stopwords)	7.48	14.33
Median query length (without stopwords)	7	11

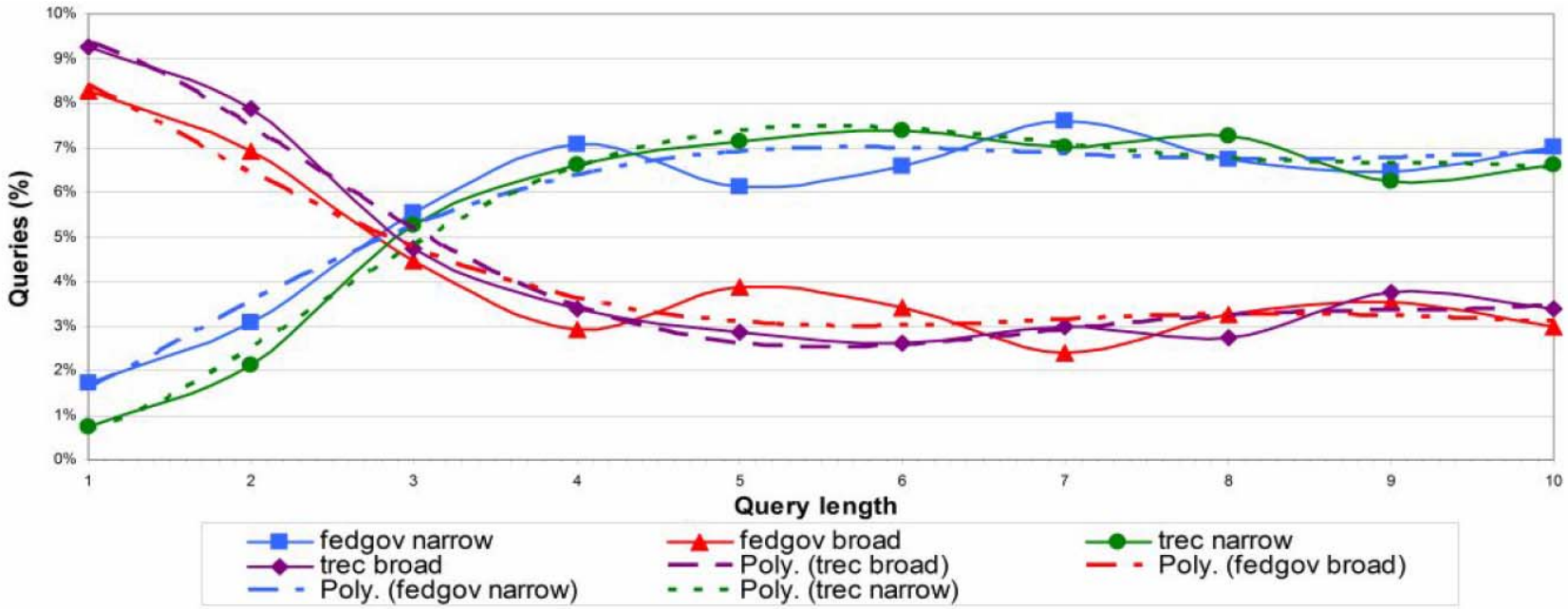
- Voice queries are considerably longer than typed mobile queries
- Some popular prepositions (like 'in' and 'at') appear twice as frequently in the voice sample data set as in the other samples



Percentages of part-of-speech in written and spoken queries.

Information Need Specificity (Broad/Narrow)

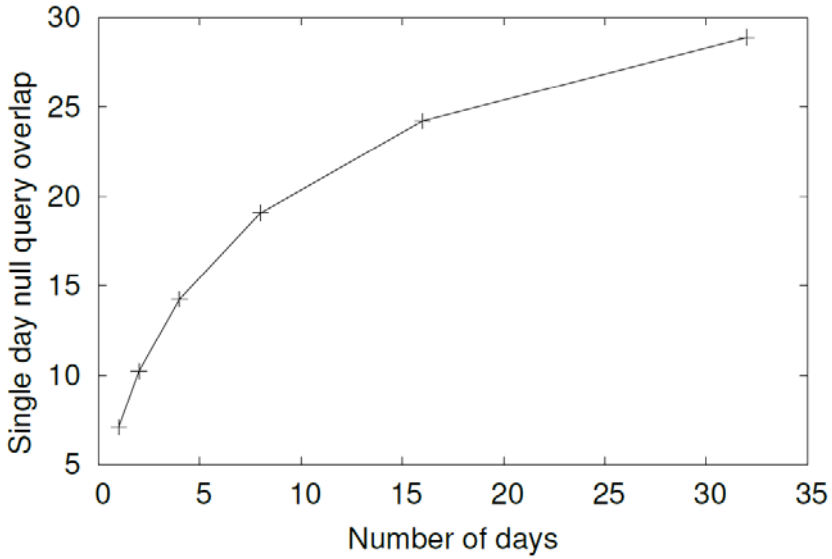
- Results show a strong correlation between decreasing query length and increasing broadness or generality of the IR request with a cross-over at 3 words/query
- Manually labeled as broad/narrow on 4-point scale



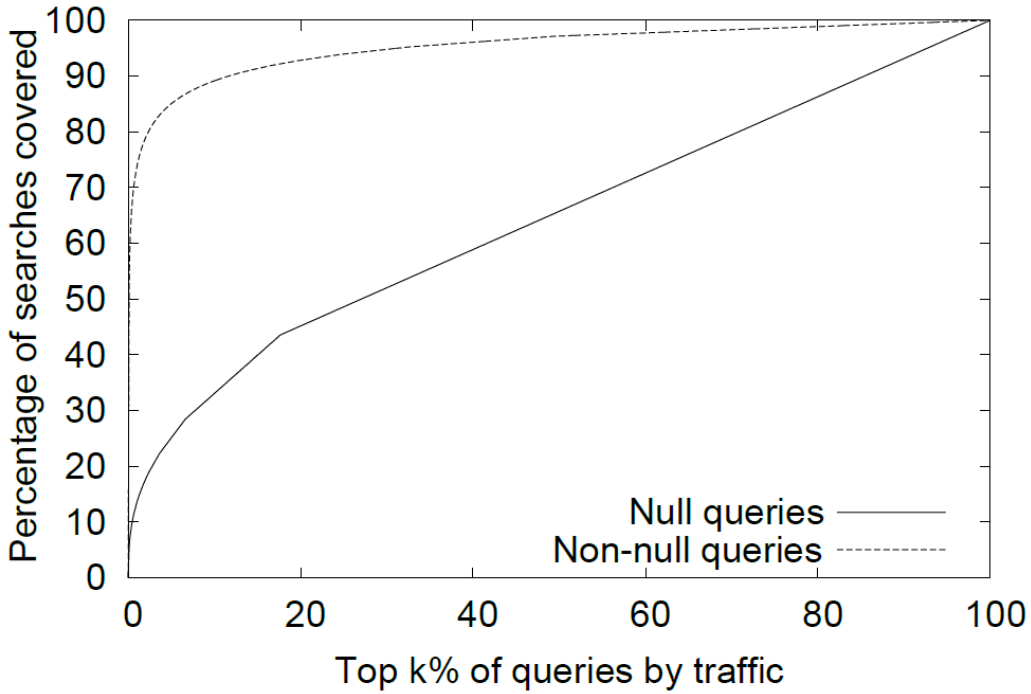
Repetition Factor

- Query log from ebay.com
- Even the most popular null queries do not repeat more than tens of thousands of times within a month. But the most popular non-null query repeats more than millions of times.

	Repetition factor
Null queries	1.45
Non-null queries	19.57



Percentage of Searches Covered by Top K% Queries



- Only 30% of null query traffic is covered by 10% most popular null queries. But 90% of non-null query traffic is generated by 10% most popular queries.

ZL01, ZL04

Smoothing for Verbose Queries

- Irrespective of the smoothing method, the performance of longer queries is much more sensitive to the choice of the smoothing parameters than that of title queries.
- Verbose queries generally require more aggressive smoothing compared to short keyword queries.
- Dirichlet prior method is good for short queries, but Jelinek-Mercer is better for verbose queries. [ZL04] Dirichlet prior method is best even for longer queries [CP15]
- Jelinek Mercer: $\mathrm{p_s(w|d)} = (1-\lambda)p_{ml}(w|d) + \lambda p(w|C)$
  - $p_{ml} = \frac{c(w,d)}{\sum_{w' \in V} c(w',d)}$; λ=0.7 usually works well.
- Dirichlet prior method: $p_s(w|d) = \frac{c(w,d)+\mu p(w|C)}{\sum_w c(w,d)+\mu}$
- 2-stage smoothing approach is better
  - First smooth the document language model using Dirichlet prior and then further smooth using Jelinek Mercer
  - $p_s(w|d) = (1-\lambda)\frac{c(w,d)+\mu p(w|C)}{|d|+\mu} + \lambda p(w|C)$

JZ07

Tokenization for Verbose Queries

- Performed study for biomedical queries.
- Main results
  - For queries with only gene symbols (short symbol queries), removing a set of special characters and replacing Greek letters with Latin letters are effective.
  - For queries with only full gene names (short name queries) and for verbose queries that also contain English words to describe the information needed, replacing special characters (like hyphens, slashes, brackets) with spaces produces best results.
    - Finding hidden break points (like alpha-numeric boundaries or camel-case boundaries) does not help much.
  - For verbose queries, stemming further improves the performance.

AGQ13, FK00

Effect of User Interfaces

- Does the user interface result in longer/shorter queries?
- Franzen et al. found that the apparent length of the query field has an effect on the length of the query users enter.
- Agapie et al. found that a halo with color change around the query field leads people to writing long queries.

Figure 1. An empty query box has a pink halo.

Figure 2. As the person starts to type, red hue starts to fade.

Figure 3. As the query gets longer, the halo becomes progressively bluer.

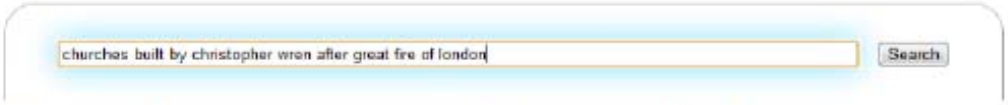


Figure 4. Long queries are displayed with a blue halo.

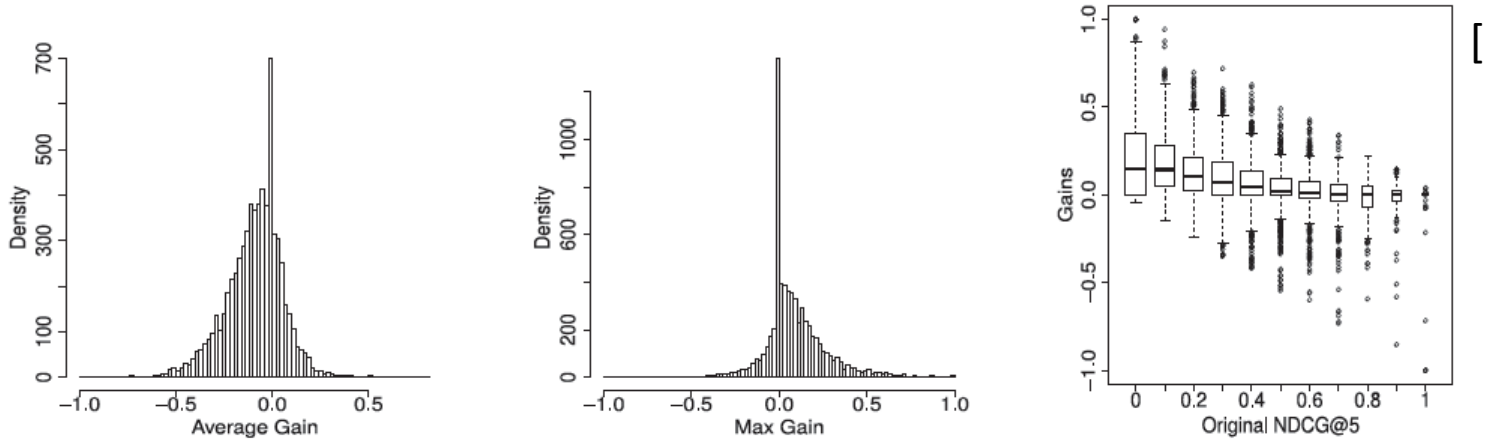
Tutorial Overview

- Properties of Verbose Queries [30 min]
- **Techniques for Processing Verbose Web Queries**
  - **Query Reduction to a Single Sub-Query [50 min]**
  - Query Reduction by Choosing Multiple Sub-Queries [10 min]
  - Break [30 min]
  - Weighting Query Words [90 min]
  - Lunch Break [90 min]
  - Query Expansion by Including Related Concepts [20 min]
  - Query Reformulation [30 min]
  - Query Segmentation [40 min]
  - Break [30 min]
  - Query-Dependent Learning to Rank [40 min]
- Applications of Verbose Query Processing [40 min]
- Summary and Future Research Directions [10 min]

Query Reduction to a Single Sub-Query

- Rather than firing the verbose query to the search engine, fire a shorter version!
- Formal Definition: Given an arbitrary query $Q = \{q_1, \ldots, q_n\}$, let P^Q denote power set of Q. Let $T_f(P)$ denote a target measure of effectiveness of ranking function f for query P.
  - Query reduction problem aims at finding a sub-query $P^* = argmax_{P \in P^Q} T_f(P)$
- Why? Observation: Perfectly reducing long TREC description queries can lead to an average improvement of 30% in mean average precision [KC09]

[BKC10]



(a) Average Gains (b) Max Gains (c) Original versus Gains

Figure 1: Distribution of Potential Gains in NDCG@5.

Query Reduction to a Single Sub-Query Examples

- "ideas for breakfast menu for a morning staff meeting" → "breakfast meeting menu ideas" [KC09]
- "Provide information on all kinds of material international support provided to either side in the Spanish Civil War" → "Spanish Civil War" [BC08]
- "Would the meteor shower that hits this weekend be better to watch tonight or tomorrow night?" → "meteor shower" [HC10]
- "*Define Argentina and Britain international relations*." → "Britain argentina" [KA07]
- "*Find articles containing contents from reports on the decline of the unemployment rate as South Korea overcame the foreign exchange crisis.*" → "unemployment rate South Korea" [LCKC09]

Considerations when performing Query Reduction

- Best sub-query: one that best satisfies the user's information need.
- What could be the candidates for a sub-query?
- What could be the features identifying an appropriate sub-query?
- Does asking for user input help?
- What are the ways to combine these features in order to find the best sub-query?
- How can we make the search for the best query efficient?

Types of Sub-query Candidates

- Individual words [ACC+96, LLCC09, PC10, PCS11]
- All two word combinations [LCKC09]
- All word subsets [KA07, KC09, DV11, CLOJ11, KA08]
- Word subsets with one word deleted [BKC10, JF03, YPSS14]
- Matching queries from personal query log [CZ09]
- POS blocks of a fixed length[LO08]
- One to three word queries without stop words [MC13]
- Right part of the query [HC10]
- Noun phrases [BC08]
- Named entities [KA07, KC09]
- All matching queries from personal query log [CZ09]
- Most frequent POS blocks [LO08]

Features to Extract Single Sub-query Statistical Features

- TF, IDF [BC08, CLOJ11, KC09, HC10, PCS11]
  - TF=Term Frequency in corpus [BC08]
  - IDF=Inverted Document Frequency in corpus [BC08]
  - Average IDF of query terms [CLOJ11, KC09]
  - Maximum IDF of query terms [CLOJ11, KC09]
  - Sum, standard deviation, max/min, arithmetic mean, geometric mean, harmonic mean, coefficient of variation of IDF of query terms [KC09]
  - RIDF=Residual IDF in corpus [BC08]
    - Difference between observed IDF and the value predicted by a Poisson model
    - $RIDF(P) = IDF(P) - \log\frac{1}{1-e^{\theta_i}}$ where $\theta_i = \frac{tf(P)}{N}$
  - TF extracted from Google n-grams counts [BC08]
  - TF in matching Wiki titles [XHC10]
  - TF in MSN query logs [XHC10]

Features to Extract Single Sub-query
Statistical Features

- Simplified clarity score (SCS) [KC09, CLOJ11]
  - KL divergence between query model and collection model
  - Pre-retrieval equivalent of clarity score
  - $SCS(q_i) = \sum_{q_i \in Q} \frac{tf_Q(q_i)}{|Q|} \times \log_2 \frac{\frac{tf_Q(q_i)}{|Q|}}{\frac{tf_C(q_i)}{|C|}}$
- Similarity Collection/Query-based (SCQ) score [KC09, CLOJ11]
  - Queries that have higher similarity to the collection as a whole are of higher quality
  - $SCQ(q_i) = \left(1 + \log \frac{tf(q_i)}{N}\right) \times \log\left(1 + \frac{N}{df(q_i)}\right)$
  - Max contributing term to SCQ score
- Dictionary based features [YPSS14]
  - isBrandName, probability of the word occurring in product titles

Features to Extract Single Sub-query
Statistical Features

- Mutual information (MI) between words [KA07, KA08, KC09, YPSS14]
  - For a subquery q, form a graph with words as vertices, weight=mutual information between terms.
  - $I(q_i, q_j) = \log \frac{\frac{n(q_i,q_j)}{|C|}}{\frac{tf(q_i)}{|C|} \times \frac{tf(q_j)}{|C|}}$
    - Average mutual information between words
    - Weight of the maximum spanning tree
    - Versions of the above two with only those subqueries that contain entities.
- MI between a word and category [YPSS14]
  - Is word w most relevant to query category among other terms in q?
- Count of passages containing sub-query [XHC10]

Features to Extract Single Sub-query
Linguistic Features

- POS [PCS11, LCKC09, LLCC09, LO08, YPSS14, XHC10]
  - Noun, verb, adjective, adverb
  - Ratio of nouns, adjectives and verbs in a query per query length
  - Conjunction, adjective, numeric
- Named entities [KA07, LCKC09, LLCC09,BKC10, XHC10]
  - Is the query word a person names, locations, organizations, time
  - Does the query contain a location?
- Combination of POS and Named entities for a pair of words as candidate [LCKC09]
  - E.g., if both q_i and q_j are nouns (or person names), pos_nn=1 (or ne_pp=1)
- Acronyms [LCKC09, LLCC09]
- Syntactic features [PCS11]
  - Number of noun phrases in query
  - Average depth of key-concept (noun phrases) terms in parse tree of query
  - Height of parse tree for query [XHC10]
- Lexical forms of neighboring words [YPSS14, LAC09]

Features to Extract Single Sub-query Query Features

- Query length [KC09, PCS11, BKC10]
- Similarity Original Query [KC09]
  - Cosine similarity between TFIDF vectors representing each sub-query and the original long query
- Presence of stop words [PCS11, BKC10]
  - Ratio of stop words
- Presence of URL [BKC10]
- isRightMost [YPSS14]
  - Users tend to delete the rightmost word in the query
- isLeftMost [YPSS14]
  - Users tend to put optional adjectives on the left and key noun phrases on the right.
- Is the query a question? [PCS11]
- Is the query a wh-question? [PCS11]
- Category of query [YPSS14]
- Location of word in query [LAC09]
- Is the word trailed by a comma [LAC09]

Features to Extract Single Sub-query
Word Dependency Features

- Binary dependencies [PC10]

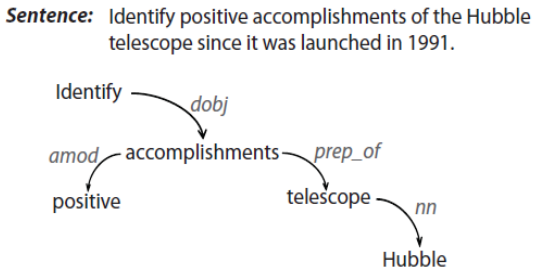


Figure 1: An example of dependency parsing trees. Labels attached to arcs are types of dependencies.

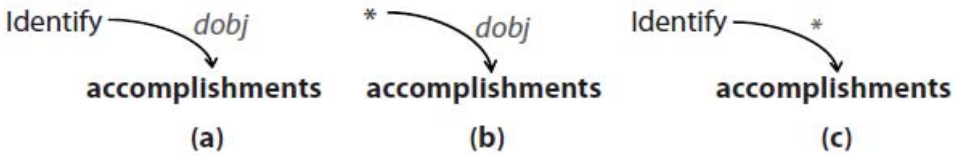


Figure 2: Three types of syntactic features for the term *"accomplishments"*. (a) An original syntactic feature (b) The word is generalized to a * (c) The type of the dependency is generalized to a *

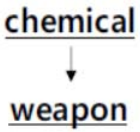
[XHC10]
Dep-obj
Dep-subj
Dep-nn

Features to Extract Single Sub-query
Word Dependency Features

- Quasi-synchronous dependencies [PCS11]
  - Number of dependent clauses in query
  - Ratio of dependent term pairs which have parent-child, ancestor-dependent, siblings and c-commanding relations in query.

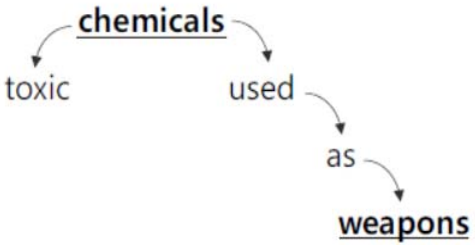
(a) *parent-child*

The inspectorate searched **chemical weapons.**



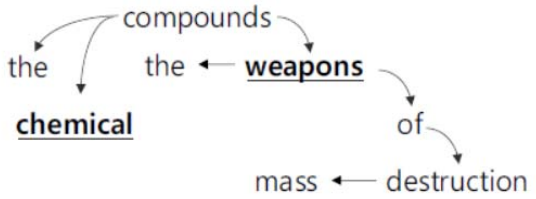
(b) *ancester-descendent*

The inspectorate searched toxic **chemicals** which is used as **weapons**.



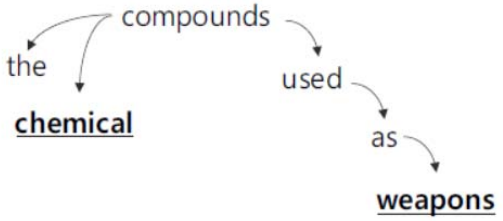
(c) *siblings*

The inspectorate searched the **chemical** compounds, the **weapons** of mass destruction.



(d) *c-commanding*

The inspectorate searched the **chemical** compounds which is used as **weapons**.



Features to Extract Single Sub-query
Query Log based Features

- Query log frequency [BC08]
  - #times q_i was used as part of a query
  - #times q_i was used as an exact query
- Similarity with old queries [CZ09]
  - Count of common terms between the long query Q and short query P
    - $R(Q,P) = \sum_{i=1}^{n} \frac{|S_i \cap P|}{|P|}$ where S_i is i[th] sentence in the long query
  - Number of common noun phrases between long query and noun phrases from sentences around short query terms in clicked result page
    - R(LQF, STF)=$\frac{2|LQF \cap STF|}{|LQF|+|STF|}$ where LQF and STF are features from long query and short query context resp.
- Deletion history [YPSS14,JF03]
  - #times word q_i was deleted
  - #times word q_i was deleted/#times q_i was seen in query log for category c
  - Conditional deletion: P(deleting q_i for similar long query earlier)
- Rareness [YPSS14]
  - Ratio of number of queries in category c to the number of queries in category c containing q_i

Features to Extract Single Sub-query
Post Retrieval Features

- Expensive to Compute!
- Query-document Relevance Scores [BKC10], [CZ09]
  - LambdaRank and BM25 scores of top-K documents (position-wise as well as aggregated as min/max/avgstd dev/variance)
  - Click through counts of top-K documents [BKC10]
  - Page-rank like scores of top-K documents [BKC10]
- Term-term co-occurrence, term-topic co-occurrence, term-term context, term-topic context [LCKC09, LLCC09]
  - Term-topic: Co-occurrence of word q_i and $\{Q - q_i\}$
  - Term-term: Co-occurrence of word q_i and $\mathrm{q_j} \in \{Q - q_i\}$ where $\mathrm{i} \neq j$
  - Term-term context: cosine similarity between context vectors of q_i and $\mathrm{q_j} \in \{Q - q_i\}$ where $\mathrm{i} \neq j$
  - Term-topic context: cosine similarity between context vectors of q_i and $\{Q - q_i\}$.
  - Context vector for q_i: <docID, relevance score> list for word q_i.
  - Defined appropriately for 2-term combinations.
  - Co-occurrence is measured as follows (N=total #docs, a=docs with q_i and q_j, b=docs(q_i,!q_j), c=docs(!q_i,q_j), d=docs(! q_i,! q_j)
    - Point-wise mutual information PMI (q_i, q_j)=log(aN/(a+b)(a+c)) [also in YPSS14]
    - Chi square statistic $X^2(q_i, q_j) = \frac{[N \times (ad-bc)^2]}{[(a+b)(a+c)(b+d)(c+d)]}$
    - Log likelihood ratio: $-2 \log LLR(q_i, q_j) = a \log \frac{aN}{(a+b)(a+c)} + b \log \frac{bN}{(a+b)(b+d)} + c\ log \frac{cN}{(c+d)(a+c)} + d\ log \frac{dN}{(c+d)(b+d)}$
    - Use average, min and max to compute feature values across various terms.
- Word co-occurrence in pseudo-relevant documents [MC13]

Features to Extract Single Sub-query Post Retrieval Features

- Query scope [KC09]
  - $QS(P) = -\log\frac{N_P}{N}$ where N_P is the number of documents containing at least one query term
- WIG(w)=Weighted Information Gain [BC08]
  - Change in information about the quality of the retrieval (in response to w) from a state where only average document is retrieved to a state where the actual results are observed
    - $wig(P) = \frac{\frac{1}{M}\sum_{d\in T_M(P)}\log(p(P|d)) - \log(p(P|C))}{-\log(p(P|C))}$ where $T_M(P)$ is a set of top M documents retrieved in response to P from collection C, and $p(P|.)$ is calculated using MLE.
- Query clarity [KC09, CLOJ11]
  - KL divergence of query model from collection model
- $QC(P) = \sum_{q_i\in P}\frac{tf_{T_M(P)}(q_i)}{|P|}\times\log_2\frac{\frac{tf_{T_M(P)}(q_i)}{|P|}}{\frac{tf_C(q_i)}{|C|}}$ where $tf_{T_M(P)}(q_i)$ and $tf_C(q_i)$ are the number of occurrences of the word q_i in the top M documents retrieved in response to the query P and the number of occurrences of the word q_i in the collection C respectively.
  - Better than IDF and MI features.
- Query drift among results
  - Standard deviation at 100 documents [CLOJ11]
  - A normalized version of standard deviation at 100 documents (i.e., the ncq predictor) [CLOJ11]
    - Std deviation normalized by relevance score of corpus (treat corpus as a big document) for the query.
  - The maximum standard deviation in the ranked-list [CLOJ11]
  - Standard deviation using a variable cut-off point [CLOJ11]
    - Cutoff = doc whose score is at least 50% of the score of the top ranking document
  - Query length normalized standard deviation using a variable cut-off point [CLOJ11]

Methods to Combine these Features/Signals

- Classification/Regression [BKC10, KC09, PC10, LCKC09, YPSS14, LLCC09]
- Core Term Identification using Heuristic Rules [ACC+96]
- Clustering and rules based approach [CZ09]
- Key Concept Discovery [BC08]
- Using Stop structures [HC10]
- Using Random walk [MC13]
- Different Formulations to Optimize [BKC10]

Classification/Regression

- RankSVM [BKC10, KC09, PC10]
- Regression using Random Forests [BKC10]
- Multi-level regression (for individual words and also for pairs of words) [LCKC09]
- Logistic Regression [YPSS14]
- Generative and Reduction approaches based on classification and regression [LLCC09]
  - Generation: Greedily select "effective" terms until k terms are chosen.
  - Reduction: Greedily remove "in-effective" terms until k terms have been removed.

Settings	Method	Indri	TFIDF	Okapi	Avg.
NTCIR-4	UB	0.2233	0.3052	0.3234	0.2839
<desc>	Gen-C	0.1949**	0.2823**	**0.2946****	0.2572(+8.38%,+10.2%)
Queries	Gen-R	0.1954**	**0.2861****	0.2875*	0.2563(+8.00%,+9.90%)
	Red-C	0.1911**	0.2755**	0.2854**	0.2506(+5.60%,+7.46%)
	Red-R	**0.1974****	0.2773**	0.2797	0.2514(+5.94%,+7.80%)
NTCIR-5	UB	0.1883	0.2245	0.2420	0.2182
<desc>	Gen-C	0.1699**	0.2117*	0.2213*	0.2009(+9.42%,+8.65%)
Queries	Gen-R	0.1712**	**0.2221***	**0.2232***	0.2055(+11.9%,+11.1%)
	Red-C	0.1645**	0.2194*	0.2084	0.1974(+7.51%,+6.76%)
	Red-R	**0.1749****	0.2034**	0.2160*	0.1981(+7.89%,+7.13%)

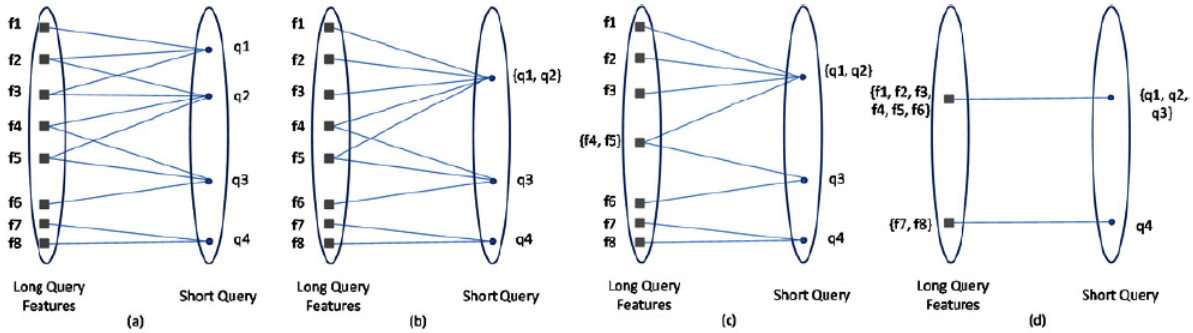
ACC+96

Core Term Identification using Heuristic Rules

- Discard query words from stopword list
- Weigh words (weight=w) based on rules like
  - Purpose clause is the most important part of query
  - For "What is the X of Y?" and "How X is Y?", Y is more important than X.
  - Etc.
- Rank words by $w \times avgTF^{0.7} \times idf$. Core term=top ranked term.
- If core term is part of some phrase, phrase becomes core term.
- Cluster query terms. If cluster has multiple terms, replace core term with a proximity operator (unordered window) containing clustered terms.
- If the term with highest w is not in core term, add it to core term.
- If proximity operator matches <10 documents, relax it by discarding terms with low weights until >=10 documents are matched.

Clustering and Rules based Approach

- Retrieve several short queries related to a long query from user's query history
  - Count of common terms between the long query Q and short query P
    - $R(Q,P) = \sum_{i=1}^{n} \frac{|S_i \cap P|}{|P|}$ where S_i is i[th] sentence in long query
- Filter non-relevant results by comparing contexts from search results with contexts from original long query
  - Number of common noun phrases between long query and noun phrases from sentences around short query terms in clicked result page
    - R(LQF, STF)=$\frac{2|LQF \cap STF|}{|LQF|+|STF|}$ where LQF and STF are features from long query and short query context resp.
- Construct short query clusters



- Select most representative query to substitute the long query
  - $S(P) = \frac{1}{n} \sum_{i=1}^{|P|} \left[\frac{num_j(P_i)}{\sum_k num_j(P_k)} \times \log \frac{|c|}{|\{c: P_i \in c\}|} \right]$
  - n=#terms in query P, $num_j(P_i)$ = #occurrences of term P_i in cluster c_j, $|c|$= #clusters, $|\{c: P_i \in c\}|$ is #clusters containing P_i

Key Concept Discovery

- Ranking principle
  - $rank(d) \propto \lambda p(Q|d) + (1-\lambda)\sum_i p(P_i|Q)p(P_i|d)$
    - P_i corresponds to a sub-query/concept (or a noun phrase) and can be a key concept or not.
    - Use AdaBoost algorithm with many features to estimate $h_k(c_i)$, i.e., confidence that $c_i \in KC$
    - $\hat{p}(P_i|Q) = \frac{h_k(P_i)}{\sum_{P_i \in Q} h_k(P_i)}$
    - For a query, choose 1 or 2 most key concepts.
- Datasets
  - TREC (ROBUST04, W10g, GOV2)
    - ROBUST04 is a newswire collection, while W10g and GOV2 are web collections
  - <title> as keyword query and <desc> as verbose query

```
<num> Number 829
<title> Spanish Civil War support
<desc> Provide information on all kinds of material
international support provided to either side in the
Spanish Civil War.
```

	ROBUST04		W10g		GOV2	
	prec@5	MAP	prec@5	MAP	prec@5	MAP
<title>	47.80	25.28	30.73_d	19.31	56.75	$\mathbf{29.67}_d$
<desc>	47.26	24.50	39.20^t	18.62	52.62	25.27^t
SeqDep<desc>	**49.11**	25.69_d	39.80^t	19.28	$\mathbf{56.88}_d$	27.53^t_d
KeyConcept[2]<desc>	48.54	$\mathbf{26.20}_d$	$\mathbf{40.40}^t$	$\mathbf{20.46}^t_d$	56.77_d	27.27^t_d

Using Stop Structures

- Stop structure = stop phrase that begins at the first word in a query.
  - "My husband would like to know more about cancer" → "cancer"
  - "if i am having a lipid test can i drink black coffee" → "lipid test can i drink black coffee"
- Stop structure identification using CRF++ and Yamcha (sequential tagger using SVMs)

	Test Set 1				Test Set 2			
	Precision	Recall	Accuracy	MSE	Precision	Recall	Accuracy	MSE
CRF++	0.891	0.825	0.816	5.1	0.930	0.867	0.866	2.68
Yamcha	0.915	0.921	0.894	2.64	0.938	0.926	0.905	1.86

	Query Test Set 1 Yahoo API		Query Test Set 1 Bing API	
	nDCG@5	nDCG@10	nDCG@5	nDCG@10
Original	0.1963	0.1718	0.2043	0.1803
Stopword List	0.2264	0.2011	0.2354	0.2162
Manual Stop Structure	0.3705^{+}	0.3338^{+}	0.3330^{+}	0.3059^{+}
Manual Stop Structure + Stopwords	0.3631^{+}	0.3177^{+}	0.3551^{+}	0.3267^{+}
Classified Stop Structure	0.3810^{+}	0.3412^{+}	0.3176^{+}	0.2893^{+}
Classified Stop Structure + Stopwords	0.3530^{+}	0.3177^{+}	0.3299^{+}	0.3038^{+}

Using Random Walk

- PhRank
  - Given query Q={$w_1, \ldots, w_n$}, let N be set of top-K relevant documents (pseudo relevant doc set). Build co-occurrence graph of word stems such that adjacent stems in N are linked.
  - Edge weights: $l_{ij} = r_{ij} \times \sum_{d_k \in T_M(Q)} p(d_k|Q)\left(\lambda c_{ijw_2} + (1-\lambda)c_{ijw_{10}}\right)$
    - c_{ijw_2} is #stem co-occurrences within window=2
    - $r_{ij} = \log_2 \frac{\sum_{ij \in T_M(Q)} c_{ijw_2}}{1+c_{ijw_2}}$
  - Compute word scores using random walk on this graph
  - Weigh each vertex score by exhaustiveness and global saliency in collection
    - $\pi_{q_i} = \pi_{q_i} \times f_{avg}(q_i) \times idf(q_i)$ where $f_{avg}(q_i)$ is freq of q_i averaged over all docs in $T_M(Q)$ and normalized by max avg freq of any term in $T_M(Q)$; $idf(q_i) = \log_2 \frac{|C|}{1+df(q_i)}$ where |C| is size of collection vocabulary
  - Candidate terms are all combinations of 1-3 words in a query that are not stop-words.
  - Term scores are computed using the average word score for words in a term, combined with global discrimination weights.
    - $f(x,Q) = z_x \times \sum_{q_i \in x} \frac{\pi_{q_i}}{n}$ where $z_x = f_x \times idf(x) \times l_x$; $l_x = |x|^{|x|}$
    - |x|=#words in term x, f_x is #times x appears in a window of size 4|x| in C. $idf(x) = \log_2 \frac{|C|}{1+df(x)}$

Using Random Walks

- PhRank achieves significant performance gains with a small number of compact terms while retaining the flexibility to select more and longer terms if required.
- R-Precision is precision at cut-off R where R is number of relevant docs for query.

	ROBUST04		WT10G		GOV2	
	MAP	R-Pr	MAP	R-Pr	MAP	R-Pr
QL	25.25	28.69	19.55	22.77	25.77	31.26
SD	26.57	30.02	20.63	24.31	28.00	33.30
sDist	–	–	21.14	24.93	27.64	33.50
PR-zF	**27.32**	**30.32**	**23.68‡**	**26.71‡**	**28.64†**	**34.13‡**
KC	25.62	28.89	20.15	22.58	26.88	32.73

Query: *Locations of volcanic activity which occurred within the present day boundaries of the U.S. and its territories.*

PhRank	Sequential Dependence	Key Concept	Subset Distribution
volcanic volcanic boundaries volcanic territories volcanic activity volcanic occurred	locations volcanic volcanic activity activity which which occurred occurred within within present present day day boundaries boundaries us us territories	present day boundaries volcanic activity	volcanic day boundaries day boundaries territories volcanic activity occurred day boundaries present day boundaries volcanic boundaries territories volcanic activity occurred activity occurred day boundaries volcanic activity occurred boundaries volcanic present day boundaries volcanic occurred boundaries + 20 bigrams (*if weights collapsed*)

Different Formulations to Optimize

- Independent Prediction
  - Given a long query and its reduced version, predict performance of each sub-query independently and select the sub-query with highest predicted performance.

$$h^* = \arg\min_h \sqrt{\sum_{\forall Q_t \in \bar{Q}_t, P \in P_1^{Q_t}} (h(P) - T(P))^2} \qquad P^* = \arg\max_{P \in P_1^Q} h^*(P)$$

- Difference Prediction
  - Predict the difference in performance between each reduced version and its original query, and then select the query that has the highest positive difference.

$$h_d^* = \arg\min_{h_d} \sqrt{\sum_{Q_t \in \bar{Q}_t} \sum_{P \in P_1^{Q_t} \wedge P \neq Q_t} (h_d(Q_t, P) - D(Q_t, P))^2} \qquad P^* = \arg\max_{P \in P_1^Q} h^*(Q, P)$$

- Ranking Queries
  - Goal is to rank the original long query and its reduced versions in order to select the top ranking query. Ranking model is learned by training on pairwise preferences between queries.

$$h_r^* = \arg\min_{h_r} \sum_{Q_t \in \bar{Q}_t} \sum_{P \in P_1^{Q_t}} I[sign(h(P) - h(Q_t)) \neq sign(T(P) - T(Q_t))] \qquad P^* = \arg\max_{P \in P_1^Q} h_r^*(P)$$

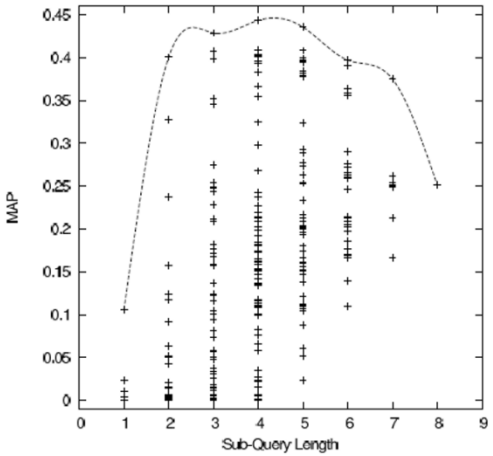
Different Formulations to Optimize

- Thresholding
  - In Independent, a reduced version is selected if and only if, there exists a reduced version whose predicted performance is greater than that of the original query by a specified threshold.
  - For Difference, the positive difference has to exceed a threshold in order to choose a reduced version.
  - For Ranking, the predicted performance of the top ranking reduced version must exceed the original query's predicted performance by the specified threshold.

	Overall NDCG@5	Affected Queries	Improved Queries	Hurt Queries	Subset NDCG Gain
No Thresholding					
Independent	35.18	4567 (70%)	1583	2346	- 4.26 (-12%)
Difference	38.63*	1761 (27%)	513	427	+ 1.61 (+4.2%)
Ranking	38.50*	612 (9%)	245	212	+ 4.64(+12.1%)
Thresholding					
Independent	**38.64***	457 (7%)	219	149	+ 6.33 (+16.5%)
Difference	38.63*	1761 (27%)	513	427	+ 1.61 (+4.2%)
Ranking	38.50*	612 (9%)	245	212	+ 4.64(+12.1%)

Efficiency Aspect

- Evaluating scores for all possible sub-queries is highly inefficient
- Ways to make candidate generation efficient
  - Consider sub-queries with a small fixed length only [KC09], say between 3 and 6 terms.
  - Consider only those subqueries that contain named entities [KC09]
  - Consider a fixed type of candidates only like named entities or noun phrases
  - Consider sub-sequences only with no gaps
  - One word deletion [BKC10, JF03, YPSS14]
    - Dropping just a single (and correct!) term from the original long query can result in a 26% improvement in NDCG@5 [BKC10]
  - Randomly pick up a few sub-queries [DV11]
    - Discard kth term with prob=$\frac{l_{opt}}{|Q|}$ where $|Q|$ is length of query and l_{opt} is optimal length of queries.
    - Better than [BKC10] with sample size as small as 3 times the query length.



Ask for User Input

- Applying automated ways to reduce long queries did not give much gain
- Solution: Rank sub-queries, seek user input
- Interface:
  - Display the description (long query) and narrative portion of TREC query
  - List of candidate sub-queries along with result snippets

	MAP	GMAP
Baseline	0.243	0.136
Average	0.172	0.025
MaxST	0.172	0.025
NE_Average	0.170	0.023
NE_MaxST	0.182	0.029

	MAP	GMAP
Baseline	0.243	0.136
AverageTop10	0.296	0.167
MaxSTTop10	0.293	0.150
NE_AverageTop10	0.278	0.156
NE_MaxSTTop10	0.286	0.159

Score of best sub-query in top 10

	Percentage of candidates better than baseline
Average	28.5%
MaxST	35.5%
NE_Average	31.1%
NE_MaxST	36.6%

% of candidates from top 10 that exceeded the baseline

Ask for User Input

- Two tricks to reduce #options shown to the user
  - Overlapping search results: Let X be union of sets of 10 docs retrieved across options. Find minimal set of options that cover X. NP hard. Greedy algo.
  - Identical snippets: Retain 1 option from the set that retrieves same snippet.

Corpus	System	P@5	P@10	NDCG@15	MAP	Avg. # options
Robust 05	Baseline	0.412	0.386	0.270	0.156	-
	IQR with 5 options	0.488	0.450	0.333	0.194	5.0
	IQE with 5 options	0.512	0.490	0.351	0.229	5.0
	SIRE with combined options	0.612	0.564	0.417	0.262	10.0
	SIRE with set cover-based pruning	0.604	0.556	0.415	0.257	7.2
	SIRE with snippet-based pruning	0.500	0.484	0.405	0.221	2.5
	IQR with 10 options	0.572	0.510	0.396	0.218	10.0
	IQE with 10 options	0.540	0.510	0.364	0.237	10.0
HARD 2003	Baseline	0.544	0.478	0.441	0.228	-
	IQR with 5 options	0.644	0.57	0.515	0.283	4.9
	IQE with 5 options	0.628	0.544	0.476	0.298	5.0
	SIRE with combined options	0.720	0.646	0.572	0.346	9.9
	SIRE with set cover-based pruning	0.720	0.636	0.570	0.343	6.8
	SIRE with snippet-based pruning	0.632	0.544	0.568	0.300	2.5
	IQR with 10 options	0.712	0.618	0.560	0.301	9.8
	IQE with 10 options	0.636	0.580	0.496	0.305	10.0
TREC 5	Baseline	0.384	0.322	0.305	0.163	-
	IQR with 5 options	0.372	0.308	0.327	0.154	4.9
	IQE with 5 options	0.352	0.302	0.312	0.166	5.0
	SIRE with combined options	0.444	0.370	0.389	0.202	9.9
	SIRE with set cover-based pruning	0.440	0.358	0.382	0.19	6.9
	SIRE with snippet-based pruning	0.348	0.298	0.378	0.161	2.3
	IQR with 10 options	0.468	0.374	0.386	0.171	9.7
	IQE with 10 options	0.368	0.312	0.320	0.168	10.0

Tutorial Overview

- Properties of Verbose Queries [30 min]
- Techniques for Processing Verbose Web Queries
  - Query Reduction to a Single Sub-Query [50 min]
  - **Query Reduction by Choosing Multiple Sub-Queries [10 min]**
  - Break [30 min]
  - Weighting Query Words [90 min]
  - Lunch Break [90 min]
  - Query Expansion by Including Related Concepts [20 min]
  - Query Reformulation [30 min]
  - Query Segmentation [40 min]
  - Break [30 min]
  - Query-Dependent Learning to Rank [40 min]
- Applications of Verbose Query Processing [40 min]
- Summary and Future Research Directions [10 min]

Why multiple sub-queries?

- IR with reduction to multiple sub-queries
  - Model the query reduction problem as a distribution over the space of all possible sub-queries based on their expected retrieval performance.
  - All these sub-queries can then be submitted to the search engine and the results obtained from all these queries are merged to obtain the final list of results for the long query.
- "give information on steps to manage control or protect squirrels."
  - "steps protect squirrels":0.621
  - "steps control squirrels":0.324
  - "steps control protect squirrels":0.048
  - "steps manage squirrels":0.002.

[XHC10]

Subset Distributions using CRF-perf

- Instead of selecting the best sub-query, it is more general to model a distribution over the space of all possible subqueries.
- A CRF optimizes labeling accuracy based on a training set of input sequences and their corresponding gold-standard label sequences.
- CRF-perf directly optimizes the expected retrieval performance over all sub-queries.
- Train set = $\{Q, \{P, m(P, M)\}\}$ where Q is the original query, P is a sub-query, $m(P, M)$ is the retrieval performance measure, M is a retrieval model.

[XHC10]

Subset Distributions using CRF-perf

- Typical CRF

$$P(P|Q) = \frac{exp\left[\sum_{k=1}^{K} \lambda_k f_k(Q,P)\right]}{Z(Q)}$$

$$Z(Q) = \sum_{P \in P^Q} exp\left[\sum_{k=1}^{K} \lambda_k f_k(Q,P)\right]$$

- CRF-perf

$$P_m(P|Q) = \frac{exp\left[\sum_{k=1}^{K} \lambda_k f_k(Q,P)\right] m(P)}{Z_m(Q)}$$

$$Z_m(Q) = \sum_{P \in P^Q} exp\left[\sum_{k=1}^{K} \lambda_k f_k(Q,P)\right] m(P)$$

[XHC10, XC11]

Subset Distributions using CRF-perf

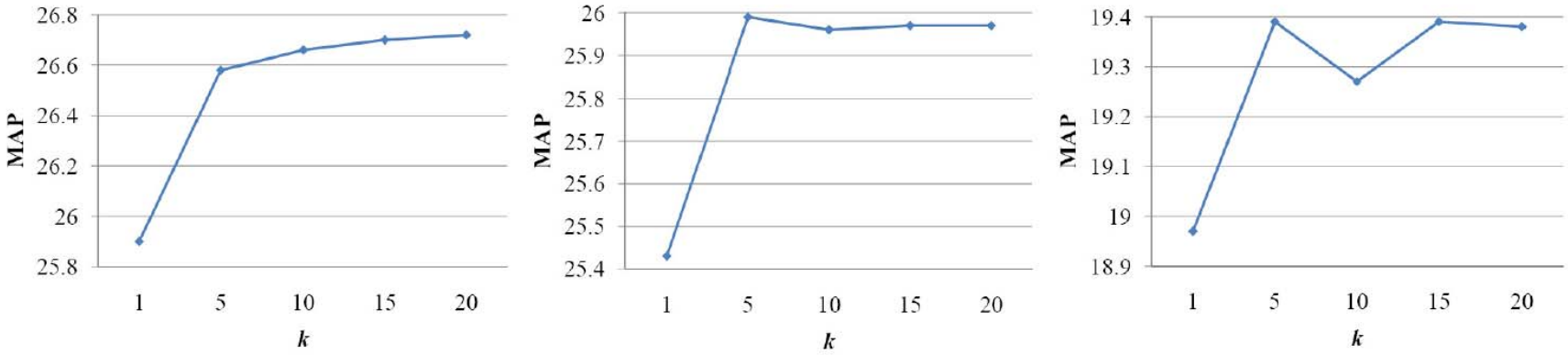
- Use a large number of features to train CRF.
- Metric optimized: Average Precision (AP)
- 4 retrieval models
  - SubQL: $score_{QL}(D,P) = \sum_{p_i \in P} \log\big(P(p_i|D)\big)$
  - SubDM: $score_{DM}(D,P) = \lambda_T \sum_{p_i \in P} \log\big(P(p_i|D)\big) + \lambda_O \sum_{o \in O(P)} \log\big(P(o|D)\big) + \lambda_U \sum_{u \in \mathrm{U}(P)} \log\big(P(u|D)\big)$
    - Usually λ_T, λ_O and λ_U are set to 0.85, 0.1 and 0.05.
  - QL+SubQL: $score(D,Q,P) = \alpha score_{QL}(D,Q) + (1-\alpha) score_{QL}(D,P)$
  - DM+SubQL: $score(D,Q,P) = \alpha score_{DM}(D,Q) + (1-\alpha) score_{QL}(D,P)$
- 2 strategies
  - Top 1: Select sub-query with highest CRF prob and feed it to M.
  - Top K: Select top k sub-queries, feed to M and compute combined score
    - $score_{QL}(D,\{P\}) = \sum_{i=1}^{k} P(P_i|Q) score_{QL}(D,P_i)$
- Use subset queries of size 3 to 6 after removing stop words.

[XHC10]

Subset Distributions using CRF-perf

[PC98]
[MC05]
[KC09]
[BC08]

	Gov2		Robust04		Wt10g	
	MAP	P@10	MAP	P@10	MAP	P@10
QL	25.43	52.21	25.49	43.13	19.61	32.68
DM	27.85	54.03	26.83	44.94	20.87	**35.77**
SRank	24.99	50.74	24.78	41.57	19.98	32.06
KeyConcept	27.52	53.83	25.97	41.65	21.01	34.02
SubQL(1)	25.90	51.88	25.43	40.84	18.97	31.55
SubQL(K)	26.66^{q}	53.36	25.96	41.93	19.27	31.75
QL+SubQL(1)	26.49^{q}	53.09	26.10	43.53	20.12	32.78
QL+SubQL(K)	26.76^{q}	53.15	26.20^{q}	43.21	19.94	33.20
SubDM(1)	28.17^{q}	53.49	26.56^{q}	42.69	20.26	33.92
SubDM(K)	28.60^{q}	53.76	27.07^{q}	43.69	20.70	34.74
DM+SubQL(1)	28.56^{q}	$\mathbf{55.91}^{q}_{d}$	27.36^{q}	$\mathbf{45.42}^{q}$	21.94^{q}	35.26^{q}
DM+SubQL(K)	$\mathbf{28.70}^{q}_{d}$	55.37^{q}	$\mathbf{27.37}^{q}_{d}$	45.14^{q}	$\mathbf{22.17}^{q}$	35.15^{q}



(a) Gov2 *(b) Robust04* *(c) Wt10g*

Effect of varying k

[XC11]

Subset Distribution using ListNet

- To estimate $P(P|Q)$
  - Assume that it is a linear combination of query features.
  - To learn the combination parameter for each query feature, we generate the corresponding retrieval feature by calculating the sum of the retrieval scores of using all subset queries weighted by their query feature values.
  - Use ListNet Learning to Rank approach to learn combination parameters.

	Gov2		Robust04	
	MAP	P@10	MAP	P@10
QL	25.43	52.21	25.49	43.13
DM	27.85	54.03	26.83	44.94
SRank	24.99	50.74	24.78	41.57
KeyConcept	27.52	53.83	25.97	41.65
QL+SubQL	26.76	53.15	26.20	43.21
DM+SubQL	28.70	55.37	27.37	**45.14**
QDist-QL	27.41^{qs}	53.42^{s}	26.07^{s}	42.69
QDist-DM	$\mathbf{29.59}^{qdsk}$	$\mathbf{55.84}^{qs}$	$\mathbf{27.55}^{qsk}$	44.94^{qsk}

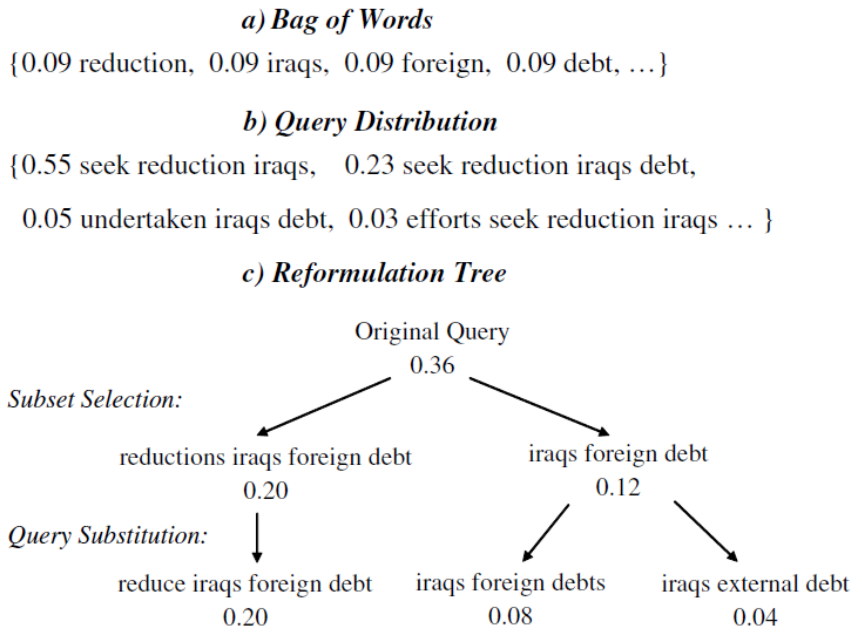
	Gov2		Robust04	
length	MAP	P@10	MAP	P@10
3	27.39	53.56	26.10	42.65
4	27.33	53.96	25.97	42.33
5	27.09	53.36	25.97	42.09
6	27.21	53.62	26.01	42.09
mix	27.41	53.42	26.07	42.69

ListNet: Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In ICML07, pages 129–136.

[XC12]

Reformulation Trees with Subset Selection and Query Substitution Operations

- A reformulation tree organizes multiple sequences of reformulated queries as a tree structure, where each path of the tree corresponds to a sequence of reformulated queries.
- A 2-level reformulation tree combines two query operations, i.e., subset selection and query substitution, within the same framework.
- A weight estimation approach assigns weights to each node of the reformulation tree by considering the relationships with other nodes and directly optimizing retrieval performance.
  - Subset distributions do not consider relationships between nodes.



Different query representations for a verbose query "identify any efforts proposed or undertaken by world governments to seek reduction of iraqs foreign debt"

[XC12]

Reformulation Trees with Subset Selection and Query Substitution Operations

- Each node of tree is a reformulated query.
- When tree T is used for retrieval, score of doc D is $sc(T, D) = \sum_{q_r \in T} w(q_r) sc(q_r, D)$
  - $w(q_r)$=weight of node corresponding to reformulated query q_r
  - $w(q_r) = w\big(parent(q_r)\big) \sum_k \lambda_k f_k(q_r)$
- Tree construction
  - Subset: remove stop words, select top 10 words by idf, generate subsets of length 3 to 6.
  - Select SubNum subset queries as nodes in tree.
  - Substitution: 3 ways
    - Morph: Morphologically similar words
    - Pattern: Words matching patterns extracted from original query
    - Wiki: Wikipedia redirect pairs
  - ModNum nodes at first level are substituted.
  - Weight assignment
    - $w(Q) = 1$ where Q is original query
    - ListNet using features aggregated across various subset queries
      - $w(q_{sub}) = \sum_k \lambda_k^{sub} f_k^{sub}(q_{sub})$
      - λs are estimated using features $F_k^{sub}(\{q_{sub}\}, D) = \sum_{q_{sub}} f_k^{sub}(q_{sub}) sc(q_{sub}, D)$
    - Weights for substituted queries
      - $w(q_{mod}) = w(q_{sub}) \sum_k \lambda_k^{mod} f_k^{mod}(q_{mod})$
      - λs are estimated using features $F_k^{mod}(\{q_{mod}\}, D) = \sum_{q_{mod}} w(q_{sub}) f_k^{mod}(q_{mod}) sc(q_{mod}, D)$
- SDM is used to compute $sc(q_r, D)$

[XC12]

Reformulation Trees with Subset Selection and Query Substitution Operations

	Gov2			Robust04			Wt10g			ClueWeb		
	MAP	P10	NDCG10	MAP	P10	NDCG10	MAP	P10	NDCG10	MAP	P10	NDCG10
QL	22.46	49.13	35.94	22.40	39.12	39.63	16.44	28.97	27.86	10.97	21.63	14.10
SDM	23.98	51.01	38.81	23.30	40.04	40.60	16.76	31.65	29.82	11.53	22.76	15.04
KC	24.88	50.87	37.72	23.87	40.76	40.75	17.45	30.82	29.55	n/a	n/a	n/a
QL+SubQL	23.36	50.81	37.96	22.85	39.28	39.98	16.81	29.79	28.48	11.01	21.84	14.16
DM+SubQL	24.82	53.36	40.55	23.65	40.32	41.15	18.25	31.13	30.71	11.54	21.94	14.85
RTree	**26.70**	**53.96**	**40.78**	**25.07**	**42.33**	**42.64**	**19.44**	**34.02**	**32.80**	**12.94**	**26.43**	**18.05**

Q: *what* allegations *have been* made about enrons culpability *in the* california energy crisis
0.194 *Q* 0.133 enrons culpability california energy crisis 0.047 california energy crisis 0.009 california gas prices 0.008 california electricity crisis
Q: *give information on* steps *to* manage control *or* protect squirrels
0.148 *Q* 0.060 control protect squirrels 0.048 control squirrels 0.013 control of ground squirrels 0.012 control squirrel
Q: *what is the* state *of* maryland *doing to* clean up *the* chesapeake bay
0.089 *Q* 0.063 maryland chesapeake bay 0.015 md chesapeake bay 0.009 maryland chesapeake bay watershed 0.034 chesapeake bay 0.007 chesapeake bay watershed 0.006 chesapeake bay river

Tutorial Overview

- Properties of Verbose Queries [30 min]
- Techniques for Processing Verbose Web Queries
  - Query Reduction to a Single Sub-Query [50 min]
  - Query Reduction by Choosing Multiple Sub-Queries [10 min]
  - **Break [30 min]**
  - Weighting Query Words [90 min]
  - Lunch Break [90 min]
  - Query Expansion by Including Related Concepts [20 min]
  - Query Reformulation [30 min]
  - Query Segmentation [40 min]
  - Break [30 min]
  - Query-Dependent Learning to Rank [40 min]
- Applications of Verbose Query Processing [40 min]
- Summary and Future Research Directions [10 min]

Tutorial Overview

- Properties of Verbose Queries [30 min]
- Techniques for Processing Verbose Web Queries
  - Query Reduction to a Single Sub-Query [50 min]
  - Query Reduction by Choosing Multiple Sub-Queries [10 min]
  - Break [30 min]
  - **Weighting Query Words [90 min]**
  - Lunch Break [90 min]
  - Query Expansion by Including Related Concepts [20 min]
  - Query Reformulation [30 min]
  - Query Segmentation [40 min]
  - Break [30 min]
  - Query-Dependent Learning to Rank [40 min]
- Applications of Verbose Query Processing [40 min]
- Summary and Future Research Directions [10 min]

Query Weighting

- Given a verbose query we thus far examined
  - Selecting a sub-query
  - Selecting multiple weighted sub-queries
- Rather than making a binary decision per term, query weighting assigns a weight to each query word or phrase
  - Assigns higher weights to terms / phrases that are more likely to retrieve relevant documents
- Relevance feedback involves weighting terms but it involves user interaction. Query weighting will not involve any user interaction.

Why not just IDF?

- Term weighting is context sensitive

Query (TREC 3 Ad-hoc retrieval track)	Term	P($t \mid R$)	idf
Oil ***Spills***	*spill*	0.9914	5.201
Term *limitations for US Congress members*	*term*	0.9831	2.010
Insurance Coverage which pays for Long ***Term*** *Care*	*term*	0.6885	2.010
School Choice Voucher System and its ***effects*** *on the US educational program*	*effect*	0.2821	1.647
Vitamin the cure or cause of human ***ailments***	*ailments*	0.1071	6.405

Why not just IDF?

- And does not always correspond to intuition

Query "civil war battle reenactments"

Concept	Importance Features			Weight
	TF	…	DF	
civil	16.9		14.1	*0.0619*
war	17.9		12.8	*0.1947*
battle	16.6		12.6	*0.0913*
reenactments	10.8		9.7	*0.3487*
civil war	14.5		10.8	*0.1959*
war battle	9.5		7.4	*0.2458*
battle reenactments	7.6		4.7	*0.0540*

[PO14]

A Fixed-Point Method

- Iteratively estimate which terms are most central to the query
- Use an initial set of retrieval results to define a recursion on the term weight vector that converges to a fixed point representing the vector that optimally describes the initial result set.
- Two key intuitions
  - Important terms are more frequent than the less important terms in the segment of the collection where original query terms are densely present
  - Importance of a term increases if it is more frequent than other important terms
- Iteratively update centrality weight using power iterations.
- Compute term weight by combining centrality factor with an IDF factor.
- Let $A(q_i)$ denote centrality of term q_i.

A Fixed-Point Method

$$A(q_i) = \sum_{j=1, i\neq j}^{|Q|} \sum_{d \in D} RF(q_i|q_j, d) A(q_j)$$

$$RF(q_i|q_j, d) = \begin{cases} \frac{\log_2(1+tf_d(q_i))}{\log_2(1+tf_d(q_j))}, & \text{if } tf_d(q_j) > 0 \\ \log_2(1 + tf_d(q_i)), & \text{otherwise} \end{cases}$$

Importance of the word q_i is $I(q_i) = A(q_i).\frac{idf(q_i)}{c + idf(q_i)}$

Metric	Method	TREC	WT10G	GOV2	ClueWeb
MAP	QL	0.191	0.184	0.256	0.124
	SD	0.200^{q}	0.193	0.276^{q}	0.129
	TA	$\mathbf{0.230}^{qrs}$(20%, 14%, 15%)	$\mathbf{0.231}^{qrs}$(25%, 19%, 20%)	$\mathbf{0.292}^{qr}$(14%, 11%, 6%)	$\mathbf{0.156}^{qrs}$(26%, 22%, 21%)
NDCG	QL	0.381	0.318	0.420	0.179
	SD	0.402^{q}	0.340^{q}	0.432^{q}	0.188
	TA	$\mathbf{0.413}^{q}$(8%, 6%, 3%)	$\mathbf{0.365}^{qrs}$(15%, 13%, 7%)	$\mathbf{0.451}^{qrs}$(7%, 6%, 4.4%)	$\mathbf{0.225}^{qrs}$(26%, 21%, 20%)

Metric	Method	TREC	WT10G	GOV2	ClueWeb
MAP	QL	0.191	0.184	0.256	0.124
	KC	0.212^{q}	0.201^{q}	0.273^{q}	-
	WSD	0.207^{q}	0.214^{q}	0.285^{q}	0.138^{q}
	TA	$\mathbf{0.230}^{kw}$ (9%, 11%)	$\mathbf{0.231}^{kw}$ (15%, 8%)	$\mathbf{0.292}^{k}$ (7%, 3%)	$\mathbf{0.156}^{w}$ (-, 13%)
NDCG@20	QL	0.381	0.318	0.420	0.179
	KC	0.382	0.330^{q}	0.432	-
	WSD	$\mathbf{0.416}^{q}$	0.360^{q}	0.439^{q}	0.188
	TA	0.413 (8%, -1%)	$\mathbf{0.365}^{k}$ (11%, 1%)	**0.451** (4%, 4%)	$\mathbf{0.225}^{w}$ (-, 20%)

	TREC	WT10G	GOV2	ClueWeB
SD	0.9	1.5	16.3	21.0
KC	0.7	1.2	9.8	15.7
WSD	1.2	1.5	22.7	28.3
TA	0.6	1.1	8.6	14.6

Computation time in minutes

[ZC10]

Word Necessity Prediction using Regression

- If a term is not **central** to a topic, it is unlikely to be necessary.
  - Let $S(t,w)$ denote cosine similarity between words t and w in SVD concept space. SVD is computed using topK relevant documents for query. Topic centrality(t)=$S(t,w)$ where w is the word with highest $S(t,w)$. Mostly $t = w$ and $S(t,w)$ indicates weight of term preserved after SVD.
- **Searchonyms (synonyms, anotonyms, hyponyms)** may replace the original query term in relevant documents, lowering the necessity of the query term.
  - Synonymy(t) = $\sum_{i=2}^{c+1} \frac{S(t,w_i)}{c}$
- If a term does not often co-occur with its synonyms, then it is often **replaceable** by the synonyms, and the term is less necessary.
  - Replaceability(t) = $\sum_{i=1..6}^{w_i \neq t} \frac{df(w_i)-n(t,w_i)}{df(w_i)} \times \frac{S(t,w_i)}{S(t,t)}$ where $n(t,w_i)$ is #docs matching both t and w_i
- Many **abstract** but **rare** terms are unnecessary, because they are at a different abstraction level than the relevant documents.
  - Rareness(t) = $idf(t) = \log \frac{N}{df(t)}$

TREC	4	6	8	9	10	12	14
Document collection	disk 2,3	disk 4,5	d4,5 w/o cr	WT10g		.GOV	.GOV2
Topic numbers	201-250	301-350	401-450	451-500	501-550	TD1-50	751-800
LM *desc* – Baseline	0.1789	0.1586	0.1923	0.2145	0.1627	0.0239	0.1789
LM *desc* – Necessity	**0.2703**	**0.2808**	**0.3057**	**0.2770**	**0.2216**	**0.0868**	**0.2674**
Okapi *desc* – Baseline	0.2055	0.1773	0.2183	0.1944	0.1591	0.0449	0.2058
Okapi *desc* – Necessity	**0.2679**	**0.2786**	**0.2894**	**0.2387**	**0.2003**	**0.0776**	**0.2403**

[LAC09]

Regression

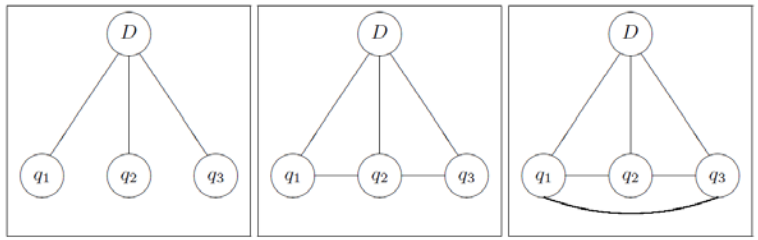
- RegressionRank [LAC09]
  - Generating labeled data given training queries with relevant and non-relevant documents
    - Each instance corresponds to a query word.
    - Various statistical, linguistic, and query features are used.
    - Computing regression label
      - Is equivalent to computing query likelihood model without using ML
      - For every training query Q, estimate $\widehat{\theta_Q} = \sum_s [Metric(\theta_s)\theta_s]$
      - θ_s are selected using grid search (query reduction candidates of sizes up to 6)

Query	Model	Robust04		W10g		GOV2	
		P@5	MAP	P@5	MAP	P@5	MAP
Title	ML	48.11	25.32	31.20	19.49	56.24	29.61
Description	ML	47.63	24.51	$39.20^{\ddagger}$	18.61	52.21	25.22
	Seq. Depend. [16]	$49.32_{\dagger}$	$25.64_{\ddagger}$	$38.80^{\ddagger}$	19.14	$56.38_{\dagger}$	$27.40_{\ddagger}$
	Key Concepts [2]	47.55	$25.91_{\ddagger}$	$41.40^{\ddagger}_{\dagger}$	$20.40_{\ddagger}$	$57.05_{\ddagger}$	$27.44_{\ddagger}$
	Regression Rank	$52.05^{\dagger}_{\ddagger}$	$27.33^{\ddagger}_{\ddagger}$	$40.60^{\ddagger}$	$22.01^{\dagger}_{\ddagger}$	54.50	$27.35_{\ddagger}$
	Oracle Regression	60.16	32.01	46.60	27.95	62.60	33.43
	Oracle Reduction		35.07		31.75		36.03

[MC05]

Sequential Dependence Model using Markov Random Fields

- Considering the Markov Random Field network between terms in a query
  - $P(D|Q) = \sum_{c \in C(G)} \lambda_c f(c)$
- 3 variants
  - Full independence
  - Sequential Dependence
  - Full Dependence
- Features (defined over cliques)



Single Terms

$$\begin{aligned}\psi_T(c) &= \lambda_T \log P(q_i|D) \\ &= \lambda_T \log \left[(1-\alpha_D)\frac{tf_{q_i,D}}{|D|} + \alpha_D \frac{cf_{q_i}}{|C|}\right]\end{aligned}$$

Exact Phrases

$$\begin{aligned}\psi_O(c) &= \lambda_O \log P(\#1(q_i, \ldots, q_{i+k})|D) \\ &= \lambda_O \log \left[(1-\alpha_D)\frac{tf_{\#1(q_i \ldots q_{i+k}),D}}{|D|} + \alpha_D \frac{cf_{\#1(q_i \ldots q_{i+k})}}{|C|}\right]\end{aligned}$$

Unordered Phrases

$$\begin{aligned}\psi_U(c) &= \lambda_U \log P(\#\text{uwN}(q_i, \ldots, q_j)|D) \\ &= \lambda_U \log \left[(1-\alpha_D)\frac{tf_{\#\text{uwN}(q_i \ldots q_j),D}}{|D|} + \alpha_D \frac{cf_{\#\text{uwN}(q_i \ldots q_j)}}{|C|}\right]\end{aligned}$$

Sequential Dependence Model using Markov Random Fields

Optimize

$$P_\Lambda(D|Q) \overset{rank}{=} \sum_{c \in C(G)} \lambda_c f(c)$$

$$= \sum_{c \in T} \lambda_T f_T(c) + \sum_{c \in O} \lambda_O f_O(c) + \sum_{c \in O \cup U} \lambda_U f_U(c)$$

s.t.

$$\lambda_T + \lambda_O + \lambda_U = 1$$

Coordinate Ascent Algorithm

- Initialize $\Lambda^0 = \{ \lambda_T^0, \lambda_O^0, \lambda_u^0 \}$
- While improvement in retrieval metric $\mathcal{M}$
  - For each $\lambda \in \Lambda$
    - Line search for optimal value of λ
    - At each search iteration, test $\mathcal{M}$
    - Renormalize the weights such that $\lambda_T^0 + \lambda_O^0 + \lambda_u^0 = 1$
  - Continue until convergence, or max iterations reached

[MC05]

Sequential Dependence Model using Markov Random Fields

Length	AP	WSJ	WT10g	GOV2
2	0.1860	0.2776	0.2148	0.2697
8	0.1867	0.2763	0.2167	0.2832
50	0.1858	0.2766	0.2154	0.2817
Unlimited	0.1857	0.2759	0.2138	0.2714

Mean average precision for various unordered window lengths with the sequential dependence variant.

	FI		SD		FD	
	AvgP	P@10	AvgP	P@10	AvgP	P@10
AP	0.1775	0.2912	0.1867* (+5.2%)	0.2980 (+2.3%)	0.1866* (+5.1%)	0.3068* (+5.4%)
WSJ	0.2592	0.4327	0.2776† (+7.1%)	0.4427 (+2.3%)	0.2738* (+5.6%)	0.4413 (+2.0%)
WT10g	0.2032	0.2866	0.2167* (+6.6%)	0.2948 (+2.9%)	0.2231** (+9.8%)	0.3031 (+5.8%)
GOV2	0.2502	0.4837	0.2832* (+13.2%)	0.5714* (+18.1%)	0.2844* (+13.7%)	0.5837* (+20.7%)
$(\tilde{\lambda}_T, \tilde{\lambda}_O, \tilde{\lambda}_U)$	(1.00, 0.00, 0.00)		(0.85, 0.10, 0.05)		(0.80, 0.10, 0.10)	

Comparison of the FI, SD and FD models

[Lea09]

Integrating Regression Rank with Markov Random Fields (MRFs)

- Although the weights for each feature class are learned from data in [MC05], feature weights within each class are estimated by the same uniform assumption as the standard unigram.
  - Another way to see this is that all feature within the same feature class are weighted by the same tied parameter λ_i
- MRF uses the ML based Dirichlet-smoothed unigram. Regression Rank provides a metric-optimized θ_Q rather than MLE. MRF's first term computation could use RegressionRank.
- λ_O and λ_U are still computed using the same ML estimates.

Model	Robust04		W10g		GOV2	
	Test	All	Test	All	Test	All
MRF [20]	38.92	30.09	19.99	20.02	32.37	30.26
RR [17]	37.03	30.52	21.77	22.48	30.36	28.96
MRF+RR	$39.13_{\ddagger}$	$31.82^{\ddagger}_{\ddagger}$	$23.19^{\dagger}_{\dagger}$	$23.05^{\ddagger}$	$32.91_{\ddagger}$	$31.20_{\ddagger}$

Quasi-synchronous Dependency Language Model

- $P(Q|D) = P(T_Q, A|T_D) = P(A|T_D)P(T_Q|A, T_D)$ where T_Q and T_D are dependency trees for query and document and A is a loose alignment.
- A is a set of all possible combinations of four syntactic configurations between a query and a document. It has 4*7 elements
  - Parent-child (2), ancestor-descendent (2), siblings, c-commanding (2)
- $P(A|T_D)P(T_Q|A, T_D) = \sum_{(syn_D, syn_Q)\in A} P(syn_D, syn_Q|T_{D,syn_D})P(T_{Q,syn_Q}|T_{D,syn_D})$
  - syn_D and syn_Q are one of the 4 syntactic configurations
  - T_{D,syn_D} represents a set of dependent terms having syn_D dependency
  - $P(syn_D, syn_Q|T_{D,syn_D})$ is prob. that a syntactic relation syn_D in a document is used in the form of syn_Q in query.
- This = $\sum_{(syn_D, syn_Q)\in A} \frac{1}{N} P(T_{Q,syn_Q}|T_{D,syn_D})$
  - N is number of elements in set A
- $P\left(T_{Q,syn_Q}\middle|T_{D,syn_D}\right) \equiv \sum_{(q_i,q_j)\in T_{Q,syn_Q}} \lambda(q_i, q_j)P(q_i, q_j|T_{D,syn_D})$
  - q_i and q_j are dependent terms with relation as syn_Q in parse tree of a query.
  - $P(q_i, q_j|T_{D,syn_D}) = (1-\alpha_D)\frac{tf_{q_i,q_j,syn_D}}{|D|} + \alpha_D \frac{cf_{q_i,q_j,syn_D}}{|C|}$
    - Where tf is document frequency and cf is collection frequency of term pairs.
- Combining with SDM, $P(Q, D) = \lambda . SD(Q, D) + (1-\lambda) . QuasiSync(Q, D)$
  - λ is learned using regression

Quasi-synchronous Dependency Language Model

Robust 2004

	MAP	nDCG	prec@10
QL	0.2414	0.5061	0.4096
SDM	0.2477	0.5097	0.4217
$QL + OW1 + UW8$	0.2462 † (-0.62%)	0.5067 † (-0.59%)	0.4177 (-0.95%)
Quasi-Synchronous Matching			
$QL + QuasiSync$	0.2754 $^{\dagger}_{\star}$ (11.19%)	**0.5473**$^{\dagger}_{\star}$ (7.38%)	0.4606$^{\dagger}_{\star}$ (9.24%)
$SDM + QuasiSync$	**0.2786** $^{\dagger}_{\star}$ (12.48%)	0.5472$^{\dagger}_{\star}$ (7.37%)	**0.4614**$^{\dagger}_{\star}$ (9.43%)
$QL + OW1 + UW8$ $+QuasiSync$	0.2765$^{\dagger}_{\star}$ (11.61%)	0.5440$^{\dagger}_{\star}$ (6.74%)	0.4582$^{\dagger}_{\star}$ (8.67%)
Exact Matching			
$QL + Exact$	0.2553 $_{\star}$ (3.07%)	0.5231 $^{\dagger}_{\star}$ (2.63%)	0.4273 (1.33%)
$SDM + Exact$	0.2590 $^{\dagger}_{\star}$ (4.54%)	0.5234 $^{\dagger}_{\star}$ (2.70%)	0.4345 $_{\star}$ (3.05%)
$QL + OW1 + UW8$ $+Exact$	0.2583 $^{\dagger}_{\star}$ (4.27%)	0.5218 $^{\dagger}_{\star}$ (2.39%)	0.4361 $_{\star}$ (3.43%)

$\star$ denotes significantly different with QL
† denotes significantly different with SDM

	Gov2		
	MAP	nDCG	P10
SDM	0.2654	0.5234	0.5195
QL+OW1+UW8	0.2674 (0.75%)	0.5246 (0.22%)	0.5228 (0.65%)
$SDM + QuasiSync$	0.2755 † (3.81%)	**0.5352** † (2.25%)	**0.5443** (4.78%)
$QL + OW1 + UW8+$ $QuasiSync$	**0.2764** † (4.14%)	0.5342 † (2.06%)	0.5396 (3.88%)

† means statistically significance difference with SDM

	Robust 2004 (MAP)		
$Length$	$\sim$ **10**	**11** $\sim$ **20**	**21** $\sim$
$\#$ $queries$	**43**	**147**	**59**
SDM	0.3062	0.2398	0.2248
$QL + OW1 + UW8$	0.3056 (-0.19%)	0.2380 (-0.75%)	0.2232 (-0.71%)
$QL + QuasiSync$	0.3268 (6.72%)	0.2613 (8.98%)	0.2731 (21.51%)
$SDM + QuasiSync$	0.3255 (6.29%)	0.2668 (11.27%)	0.2738 (21.81%)
$QL + OW1 + UW8+$ $QuasiSync$	0.3251 (6.17%)	0.2649 (10.47%)	0.2698 (20.04%)

"Quasi-Synchronous Matching" represents the experimental results where we allow inexact matching while "Exact Matching" shows the experimental results when we align between term pairs having only the same syntactic relation.

[BMC10]

Weighted Sequential Dependence Model

- Model query term dependencies and weigh generic term concepts (e.g., unigrams, bigrams, etc.)
- Represent each term/phrase using a parameterized feature vector
- Feature parameters are estimated by directly maximizing underlying retrieval metric such as MAP or DCG.

Concept	Importance Features			Weight
	TF	…	DF	
civil	16.9		14.1	*0.0619*
war	17.9		12.8	*0.1947*
battle	16.6		12.6	*0.0913*
reenactments	10.8		9.7	*0.3487*
civil war	14.5		10.8	*0.1959*
war battle	9.5		7.4	*0.2458*
battle reenactments	7.6		4.7	*0.0540*

Query "civil war battle reenactments"

[BMC10]

Weighted Sequential Dependence Model

- Sequential Dependence model has massive parameter tying – all terms of the same type are weighted equally.
- In contrast, In a verbose query, there will likely be a few concepts (terms or phrases) within the query that will carry the most weight.
- We would like to be able to weight the concepts appropriately, with regard to each other.

[BMC10]

Weighted Sequential Dependence Model

- Another extreme is to have a different λ_i for every term q_i and $\lambda_{i,i+1}$ for every bigram (q_i, q_{i+1}). But these are too many parameters.
  - Each λ now depends on 1 (or 2) query terms.
- WSDM
  - Parameterize λ
  - $g^u(q_i)$ and $g^b(q_i, q_{i+1})$ are features defined over unigrams and bigrams resp. They are doc independent and capture importance of concept within query.
  - w^u and w^b are free parameters to be estimated.

[BMC10]

Weighted Sequential Dependence Model

- Allow the parameters to depend on the concept
- Assume the parameters take a simple parametric form
  - maintains reasonable model complexity

$$\lambda(q_i) = \sum_{j=1}^{k_u} w_j^u g_j^u(q_i)$$

$$\lambda(q_i, q_{i+1}) = \sum_{j=1}^{k_b} w_j^b g_j^b(q_i, q_{i+1})$$

w - free parameters

g - concept importance features

[BMC10]

Weighted Sequential Dependence Model

Plugging in the new weights in the Sequential Dependence Model

$$P(D|Q) \overset{rank}{=} \sum_{i=1}^{k_u} w_i^t \sum_{q \in Q} g_i^u(q) f_T(q, D) +$$
$$\sum_{i=1}^{k_b} w_i^b \sum_{q_j, q_{j+1} \in Q} g_i^b(q_j, q_{j+1}) f_O(q_j, q_{j+1}, D) +$$
$$\sum_{i=1}^{k_b} w_i^b \sum_{q_j, q_{j+1} \in Q} g_i^b(q_j, q_{j+1}) f_U(q_j, q_{j+1}, D)$$

- Note that the ***Weighted Sequential Dependence*** model is also linear (with respect to ***w***)
- Use coordinate ascent for parameter optimization
- Optimize for retrieval metric such as **MAP** or **DCG**

[BMC10]

Weighted Sequential Dependence Model

	Data Source	Feature	Description
Endogenous	**Collection**	***cf(e)***	*Collection frequency for concept e*
		df(e)	*Document frequency for concept e*
Exogenous	**Google n-Grams**	***gf(e)***	*n-gram count of concept e*
	Query Log Sample	***qe_cnt(e)***	*# exact query matches for concept e*
		qp_cnt(e)	*# partial query matches for concept e*
	Wikipedia Titles	***we_cnt(e)***	*# exact title matches for concept e*
		wp_cnt(e)	*# partial title matches for concept e*

[BMC10]

Weighted Sequential Dependence Model

desc	ROBUST04			W10g			GOV2		
	prec@10	b-pref	MAP	prec@10	b-pref	MAP	prec@10	b-pref	MAP
QL	**42.69**	25.37	25.07	32.70	20.47	19.71	51.68	35.70	26.06
SD	41.77	25.90	25.58 *(+2.0/—)*	36.10*	20.73	20.32 *(+3.1/—)*	**53.56**	36.22	26.94* *(+3.4/—)*
WSD	**42.69**	**27.02**$^{*}_{\dagger}$	**27.18**$^{*}_{\dagger}$ *(+8.4/+6.3)*	**37.10***	**24.33**$^{*}_{\dagger}$	**25.23**$^{*}_{\dagger}$ *(+28.0/+24.1)*	51.81	**36.64**$^{*}_{\dagger}$	**27.38**$^{*}_{\dagger}$ *(+5.1/+1.6)*

∗ Statistically significant difference with QL
† Statistically significance difference with SD

desc	ROBUST04	W10g	GOV2
	MAP	MAP	MAP
WSD	**27.18**	**25.23**	**27.38**
WSD-UNI	27.17	24.86	26.77$_{\dagger}$
WSD-BI	26.02$_{\dagger}$	20.43$_{\dagger}$	27.00

desc	ROBUST04	W10g	GOV2
	MAP	MAP	MAP
WSD	27.18	**25.23**	**27.38**
WSD-ENDO	27.07	23.28	26.95$_{\dagger}$
WSD-EXO	**27.33**	24.68	27.33$_{\dagger}$

[BMC10]

Weighted Sequential Dependence Model

Results for web corpus (length 4+ queries)

	DCG@1	DCG@5	DCG
QL	0.629	1.691	5.844
SD	0.864	2.383	6.681 *(+14.3/—)*
WSD	**0.884**	**2.443**	**6.741** *(+15.4/+0.9)*

	DCG@1	DCG@5	DCG
WSD	0.884	2.443	6.741
WSD-ENDO	**0.885**	**2.455**	**6.760**
WSD-EXO	0.884	2.439	6.732

	DCG@1	DCG@5	DCG
WSD	0.884	**2.443**	**6.741**
WSD-UNI	0.864	2.379	6.677
WSD-BI	**0.888**	2.409	6.711

[BMC11]

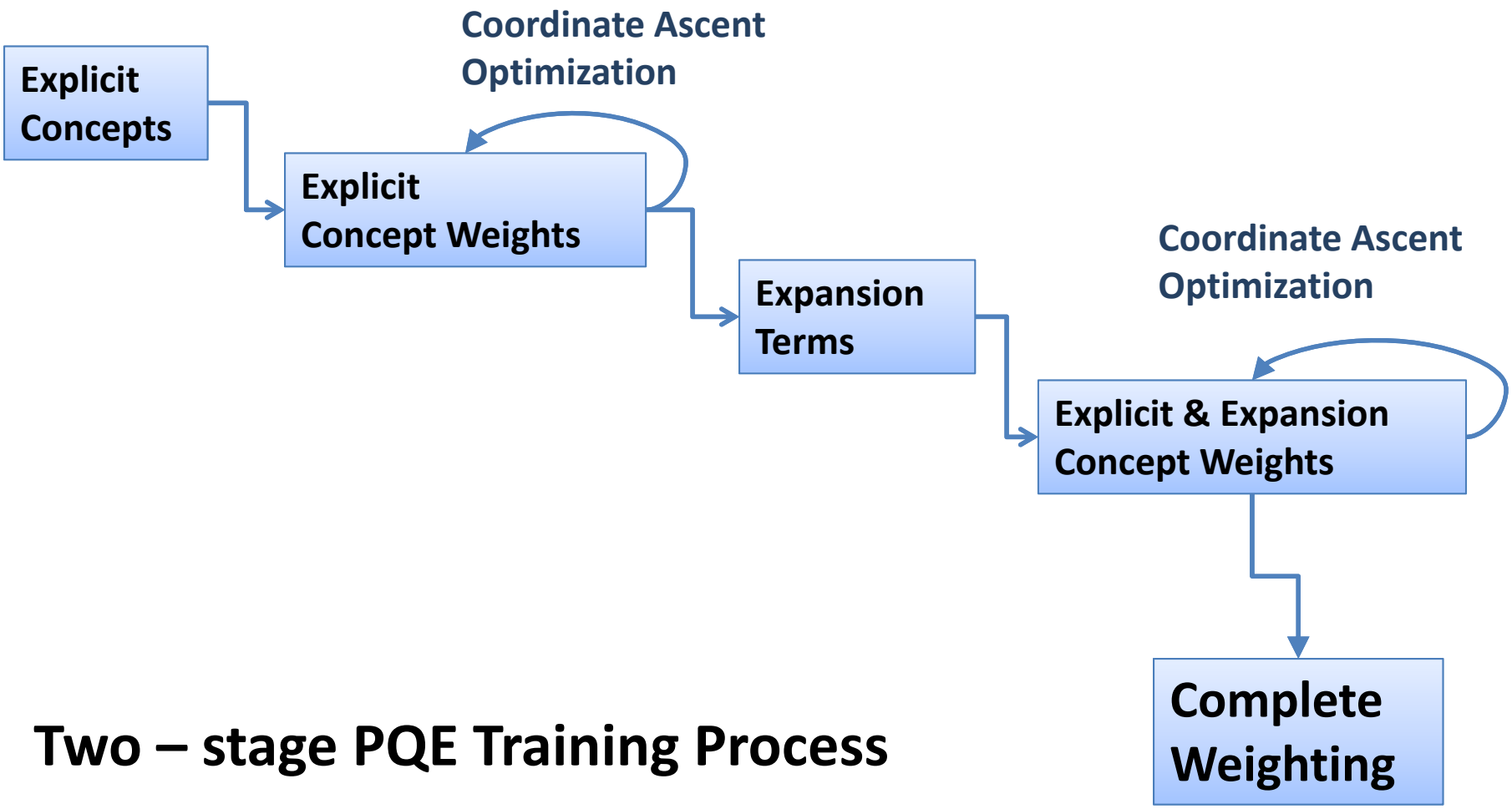
Parameterized Query Expansion Model

$$sc(Q, D) = \sum_{T \in \mathcal{T}} \sum_{\varphi \in \Phi^T} w_\varphi \sum_{\kappa \in T} \varphi(\kappa) f(\kappa, D)$$

- **Explicit Query Concepts**
  - Terms
  - Bigram Phrases
  - Bigram Proximity Matches
- **Expansion Terms**

[BMC11]

Parameterized Query Expansion Model



Two – stage PQE Training Process

[BMC11]

Parameterized Query Expansion Model

- Advantages of the model
  - Supervised query concept weighting is clearly better than the unsupervised IDF based weighting.
  - Concept weighting directly optimizes retrieval
  - Two-stage optimization enables better expansion
  - Unification of concept weighting and query expansion models.
  - Concept: Any syntactic expression including unigrams, bigrams, phrases, unordered phrases etc.
  - Previous work focused on few concept types only: noun phrases [BC08], terms [Lea09], query term spans [SKK10]

[BMC11]

Parameterized Query Expansion Model

- Supervised weighting often enables a more effective query expansion

"camels in North America in both prehistoric and modern times"

Explicit Concepts

weight	term
.8	camel
.8	north
.8	america
.2	"camel north"
.2	"north america"

Expansion with LCE (unsupervised weighting)

Mean Avg. Prec. 0.07

Expansion Terms

weight	term
.0178	indians
.0031	mexico
.0028	new
.0024	dress
.0021	clothing
…	…

[BMC11]

Parameterized Query Expansion Model

- Supervised weighting often enables a more effective query expansion

"camels in North America in both prehistoric and modern times"

Explicit Concepts

weight	term
.2591	camel
.1783	north
.1969	america
.0328	"camel north"
.0328	"north america"

Expansion with PQE (supervised weighting)

Mean Avg. Prec. 0.49

Expansion Terms

weight	term
.0314	bison
.0314	oil
.0306	nafta
.0305	fossil
.0269	expansion
…	…

[BMC11]

Parameterized Query Expansion Model

- Top K terms with highest LCE form set of ET concepts.
- 6 Features for all concepts where AP is set to LCE weight for ET concepts, 1 for other types.

Feature	Description
GF(κ)	Frequency of κ in Google n-grams
WF(κ)	Frequency of κ in Wikipedia titles
QF(κ)	Frequency of κ in a search log
CF(κ)	Frequency of κ in the collection
DF(κ)	Document frequency of κ in the collection
AP(κ)	A priori concept weight

- 2 stage optimization
  - Coordinate ascent to learn feature importance weights for explicit concept types.
  - Compute the topK ET concepts
  - Coordinate ascent to learn feature importance weights for explicit and latent concept types.

		Concept Types			
		QT	PH	PR	ET
Non-parameterized methods	QL	$\mathcal{N}$			
	SD	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	
	RM	$\mathcal{N}$			$\mathcal{N}$
	LCE	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
Parameterized methods	WSD	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	
	WRM	$\mathcal{P}$			$\mathcal{P}$
	PQE	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$

$\langle desc \rangle$	ROBUST04	WT10g	GOV2
WSD	27.49	**22.60**	29.46
WRM[10]	27.77	22.46	29.89
PQE[10]	$\mathbf{29.40}_r^s$	22.17	$\mathbf{31.68}_r^s$

Relevance model: RM [LC03]
WRM is weighted version of RM
RM and LCE are both PRF expansion models.
RM [10] means 10 expansion concepts

$\langle desc \rangle$	ROBUST04		WT10g		GOV2	
	prec@20	MAP	prec@20	MAP	prec@20	MAP
QL	33.05	24.22	26.35	18.87	47.62	25.66
SD	34.94^{q}	25.65^{q}	27.45	19.82	50.67^{q}	27.94^{q}
RM[10]	35.04^{q}	25.92^{q}	27.15	20.49^{q}	48.42	27.11^{q}
LCE[10]	37.57_r^{qs}	28.10_r^{qs}	28.50	21.26	52.52_r^{qs}	30.66_r^{qs}
PQE[10]	$\mathbf{38.98}_{rl}^{qs}$	$\mathbf{29.40}_{rl}^{qs}$ *(+11.6/+4.6)*	$\mathbf{29.90}_{rl}^{qs}$	$\mathbf{22.17}^{qs}$ *(+11.8/+4.3)*	$\mathbf{54.70}_{rl}^{qs}$	$\mathbf{31.68}_{rl}^{qs}$ *(+13.4/+3.3)*

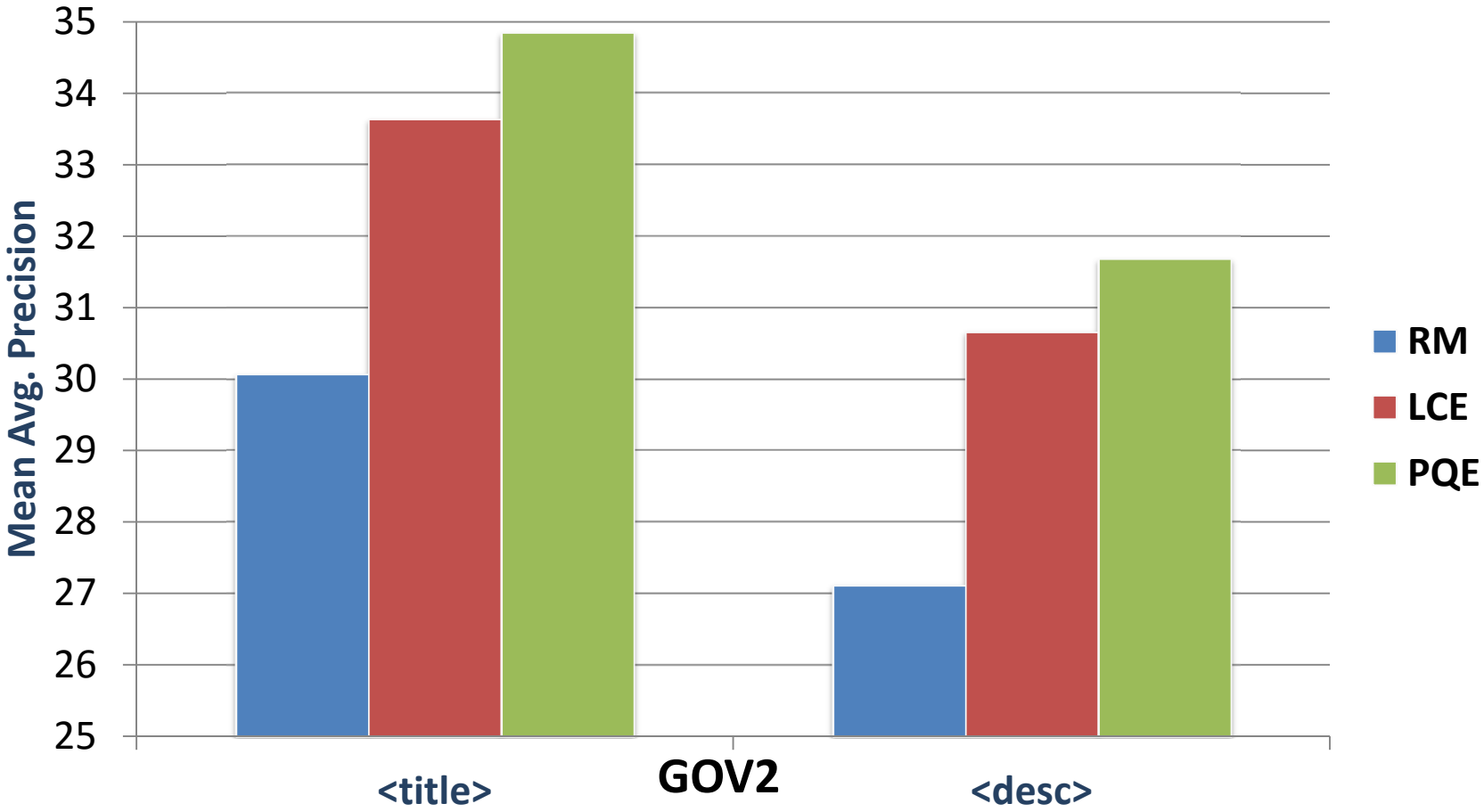
[BMC11]

Parameterized Query Expansion Model

	Query Terms	Exact Phrases	Proximity Matches	Expansion Terms
Relevance Model **(RM)**	$\mathcal{N}$			$\mathcal{N}$
Latent Concept Expansion **(LCE)**	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
Parameterized Query Expansion **(PQE)**	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$

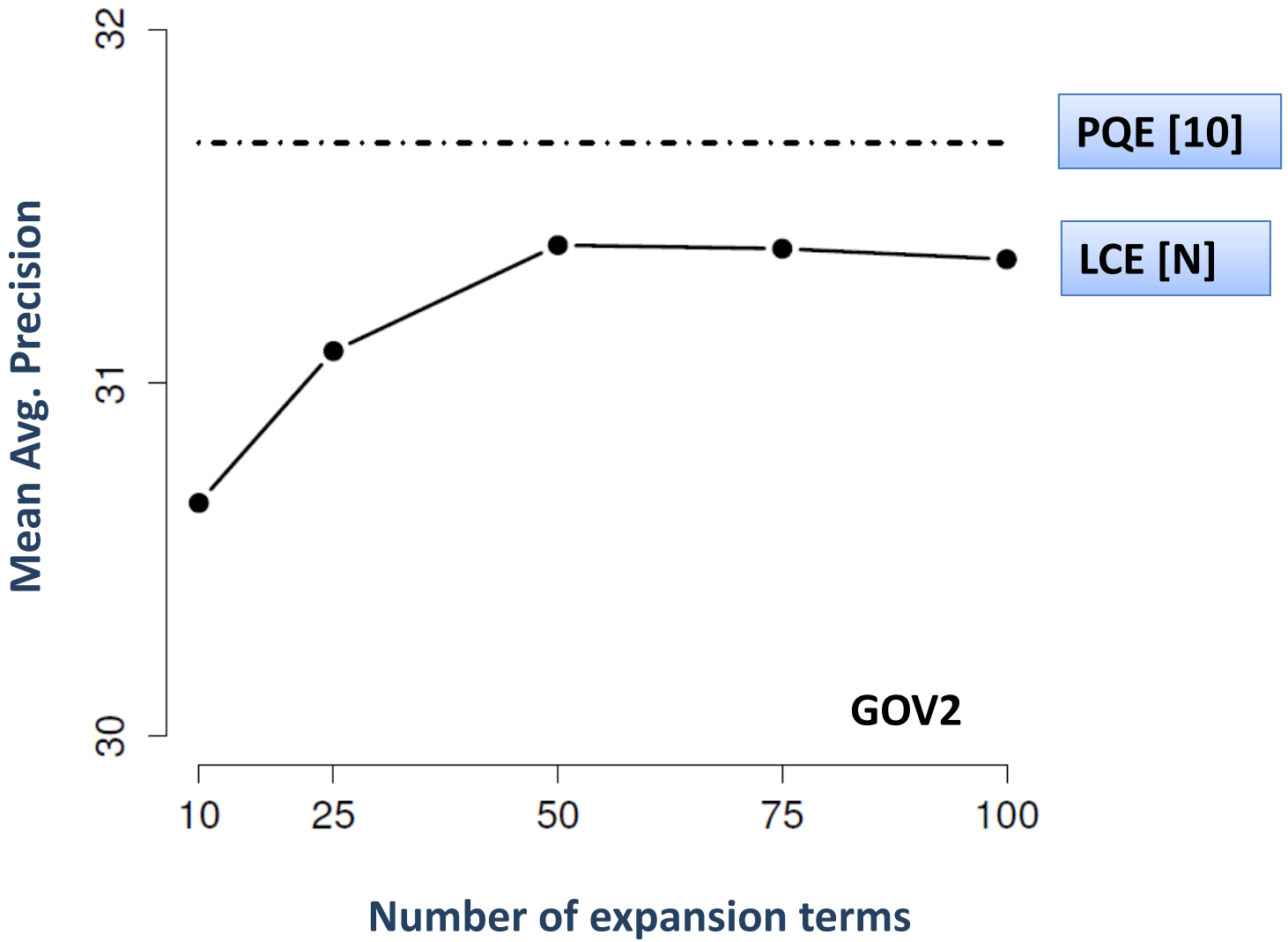
[BMC11]

Parameterized Query Expansion Model



[BMC11]

Parameterized Query Expansion Model



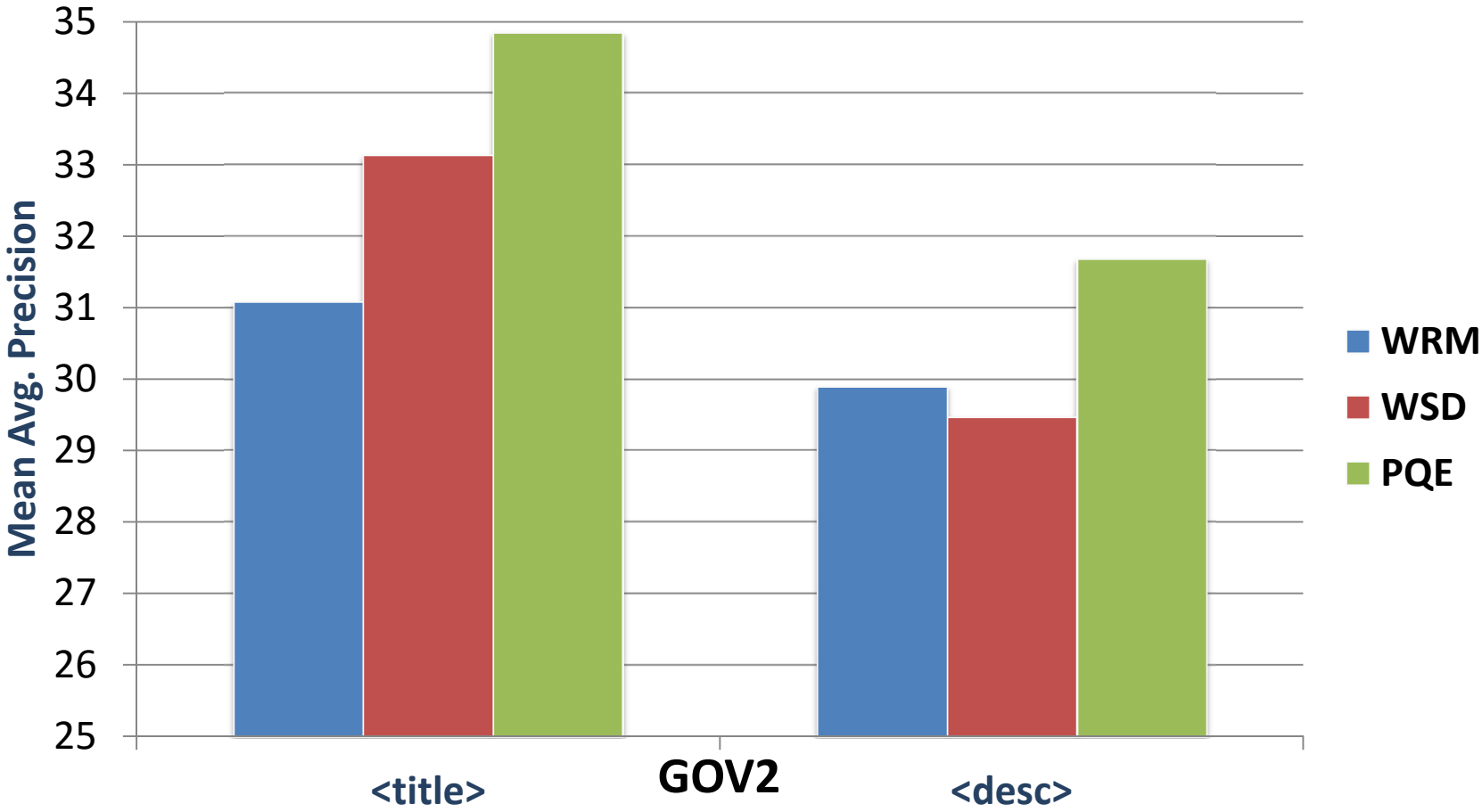
[BMC11]

Parameterized Query Expansion Model

	Query Terms	Exact Phrases	Proximity Matches	Expansion Terms
Weighted Relevance Model **(WRM)**	$\mathcal{P}$			$\mathcal{P}$
Weighted Sequential Dependence **(WSD)**	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	
Parameterized Query Expansion **(PQE)**	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$

[BMC11]

Parameterized Query Expansion Model



[BC12]

Query Hypergraphs

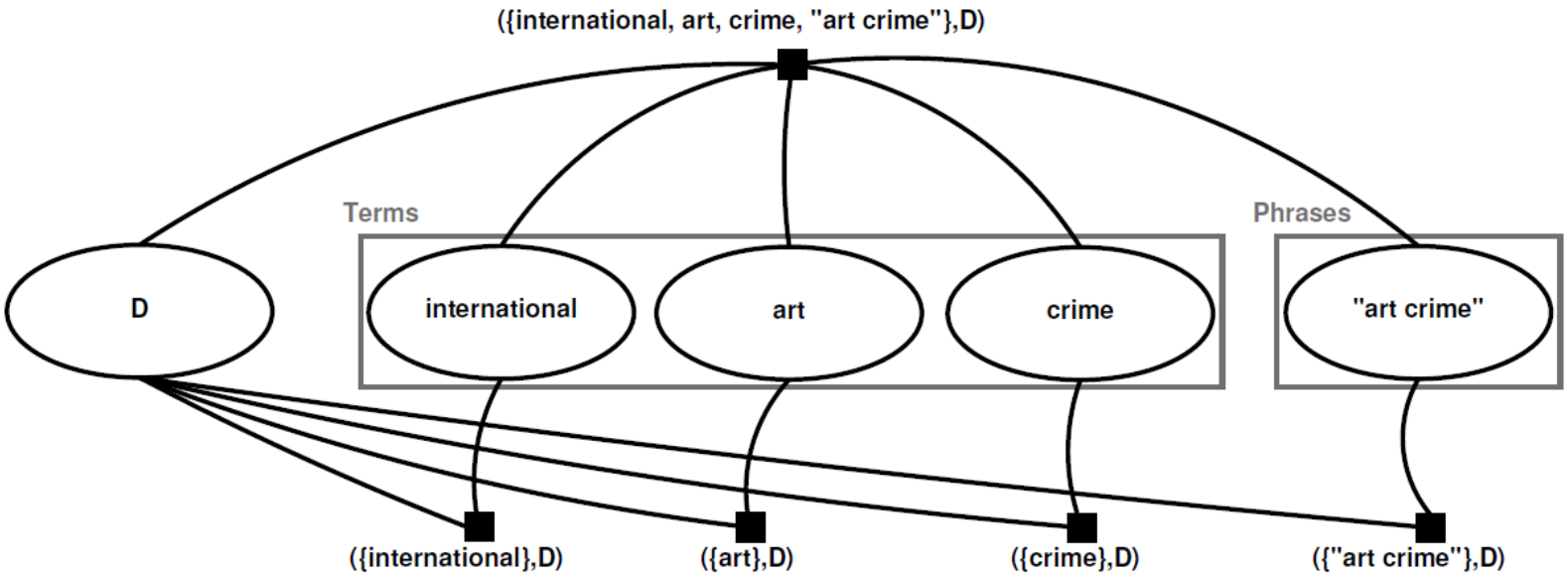
- Models higher order term dependencies
  - Vertex in hypergraph corresponds to an individual query concept
  - Dependency between a subset of concepts is modeled using a hyperedge.
- Models arbitrary term dependencies as concepts.
- Uses passage-level evidence to model dependencies between concepts.
- Assigns weights to both concepts and concept dependencies, proportionate to the estimate of their importance for expressing the query intent.

Structure $\boldsymbol{\sigma}$	Concepts $\{\kappa : \kappa \in \boldsymbol{\sigma}\}$
Terms	["members", "rock", "group", "nirvana"]
Bigrams	["members rock", "rock group", "group nirvana"]
Noun Phrases	["members", "rock group nirvana"]
Named Entities	["nirvana"]
Dependencies	["members nirvana", "rock group"]

Examples of the possible structures and the concepts they might contain for the query "members of the rock group nirvana" (stopwords removed).

[BC12]

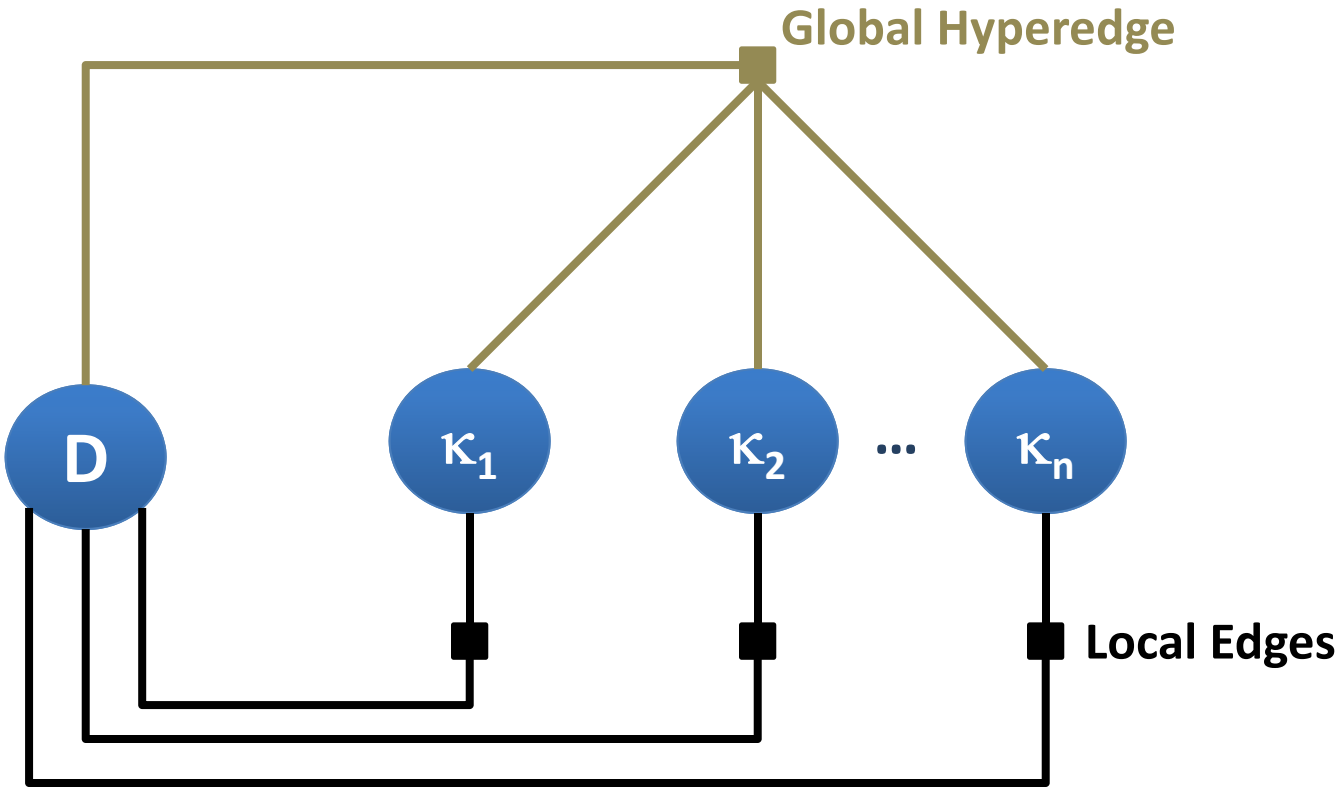
Query Hypergraphs



Example of a hypergraph representation for the query “international art crime”.

[BC12]

Query Hypergraphs



[BC12]

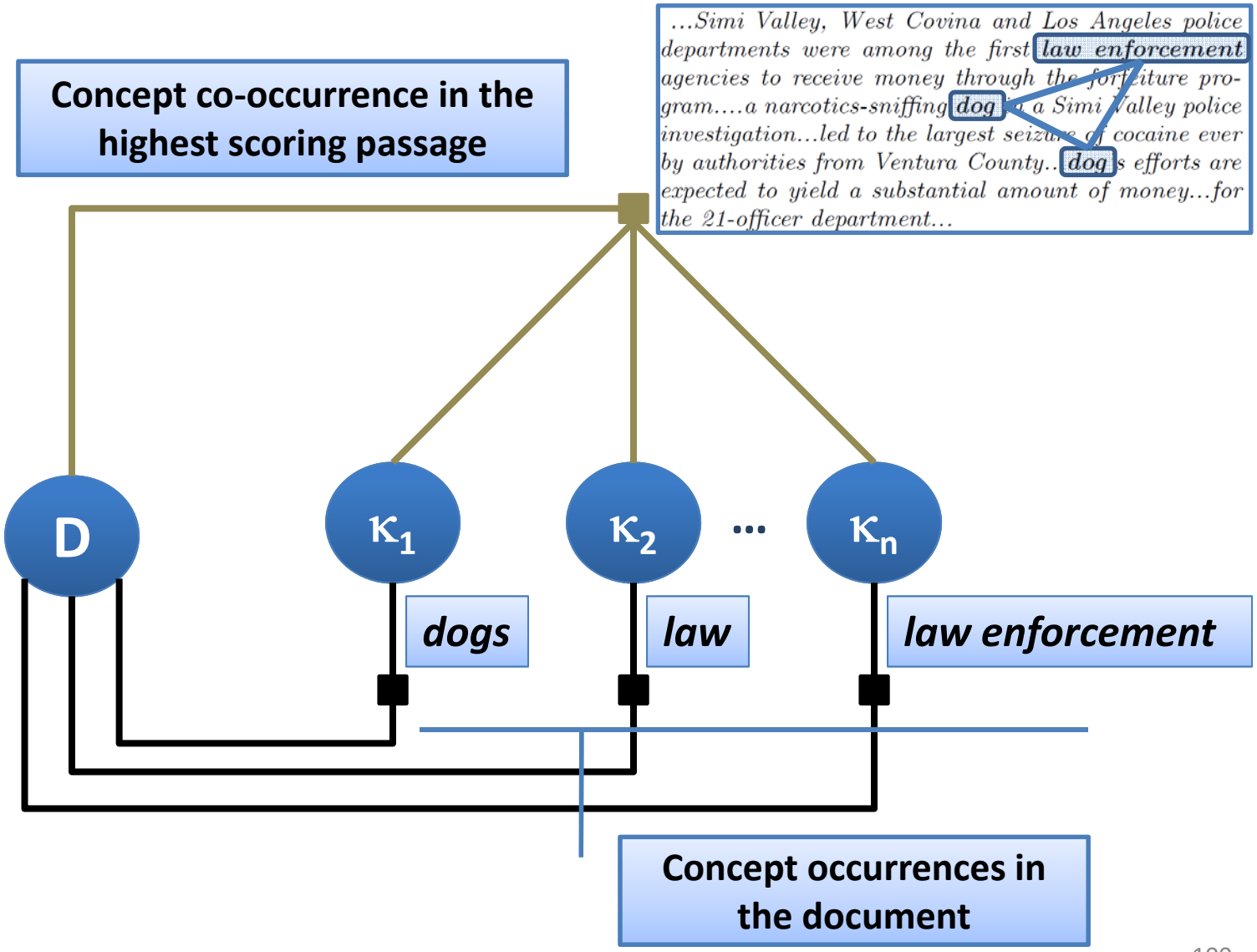
Query Hypergraphs

use of dogs for law enforcement purposes

...Simi Valley, West Covina and Los Angeles police departments were among the first ***law enforcement*** *agencies to receive money through the forfeiture program....a narcotics-sniffing* ***dog*** *in a Simi Valley police investigation...led to the largest seizure of cocaine ever by authorities from Ventura County...* ***dog's*** *efforts are expected to yield a substantial amount of money...for the 21-officer department...*

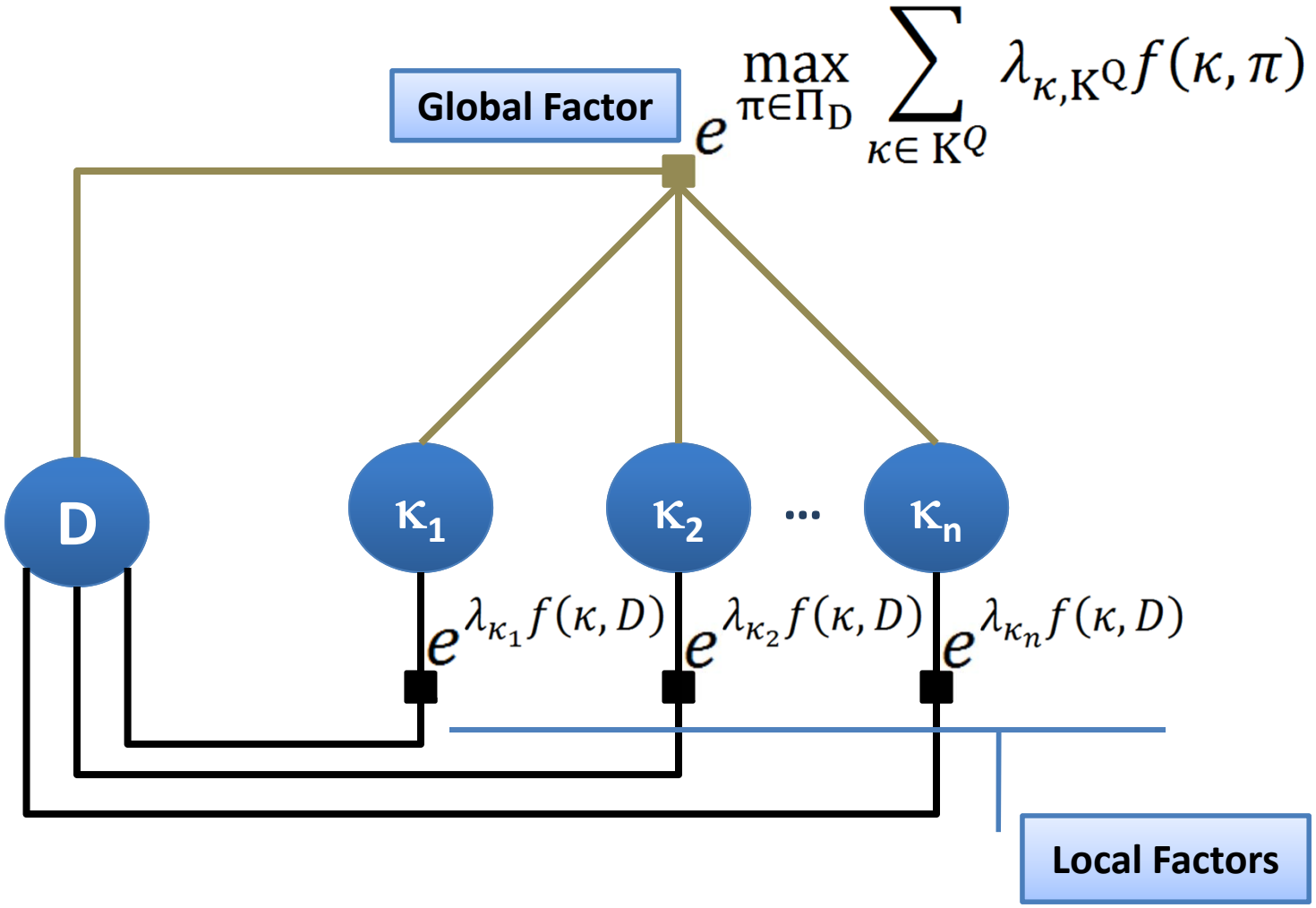
[BC12]

Query Hypergraphs



[BC12]

Query Hypergraphs



[BC12]

Query Hypergraphs

Global Factor

$$\max_{\pi \in \Pi_D} \sum_{\kappa \in K^Q} \lambda_{\kappa, K^Q} f(\kappa, \pi)$$

Highest scoring passage in the document

[BC12]

Query Hypergraphs

Global Factor

$$\max_{\pi \in \Pi_D} \sum_{\kappa \in K^Q} \lambda_{\kappa, K^Q} f(\kappa, \pi)$$

Passage Matching Function

[BC12]

Query Hypergraphs

Global Factor

$$\max_{\pi \in \Pi_D} \sum_{\kappa \in K^Q} \lambda_{\kappa,K^Q} f(\kappa, \pi)$$

Query-Dependent Concept Weight

[BC12]

Query Hypergraphs

Non-parameterized query hypergraph

Assume equal concept weights

All local factors are equally weighted

An additional weight for the global factor

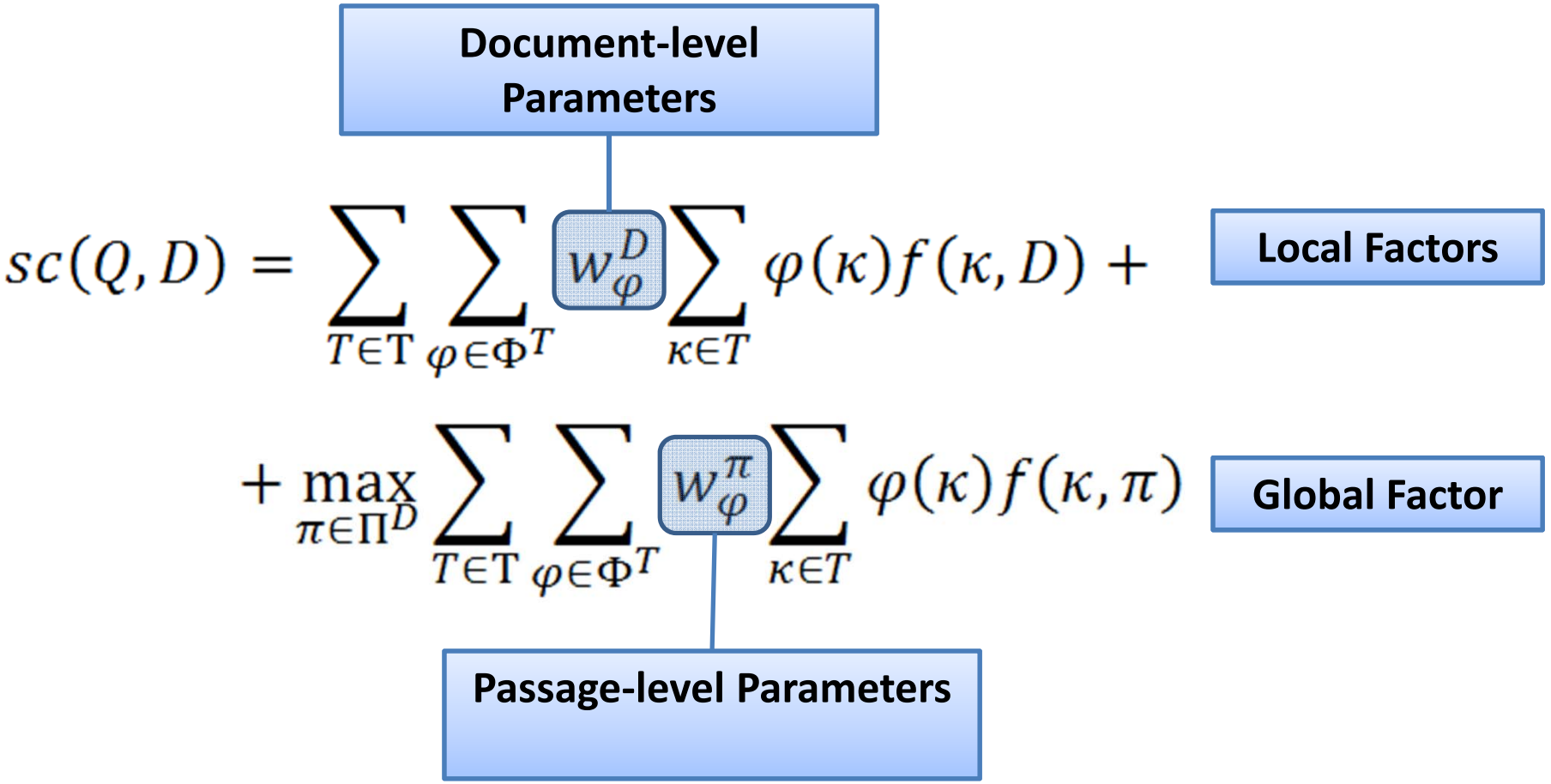
Parameterized query hypergraph

Concepts are parameterized using importance features

Twice as many parameters as in the parameterized concept weighting

[BC12]

Query Hypergraphs



[BC12]

Query Hypergraphs

	Robust04		*Gov2*		*ClueWeb-B*	
	ERR@20	**MAP**	**ERR@20**	**MAP**	**ERR@20**	**MAP**
QL	11.44	24.24	15.06	25.66	7.32	12.75
$\mathcal{H}$-QL	**11.66**	**25.49**$_q$ *(+5.2%)*	**15.33**	**27.24**$_q$ *(+6.2%)*	**7.63**	**13.07**$_q$ *(+2.5%)*

(a) Query likelihood (QL) and its hypergraph representation ($\mathcal{H}$-QL).

	Robust04		*Gov2*		*ClueWeb-B*	
	ERR@20	**MAP**	**ERR@20**	**MAP**	**ERR@20**	**MAP**
SD	11.76	25.62	15.73	27.97	7.58	12.99
$\mathcal{H}$-SD	**11.93**	**26.65**$_s$ *(+4.0%)*	**15.93**	**28.63**$_s$ *(+2.4%)*	**7.78**	**13.08** *(+0.7%)*

(b) Sequential dependence model (SD) and its hypergraph representation ($\mathcal{H}$-SD) parameterized by structure.

	Robust04		*Gov2*		*ClueWeb-B*	
	ERR@20	**MAP**	**ERR@20**	**MAP**	**ERR@20**	**MAP**
FD	11.87	25.69	**16.10**	28.25	**8.21**	13.28
$\mathcal{H}$-FD	**11.94**	**26.50**$_f$ *(+3.1%)*	16.02	**28.70**$_f$ *(+1.6%)*	8.15	**13.35** *(+0.5%)*

(c) Full dependence model (FD) and its hypergraph representation ($\mathcal{H}$-FD) parameterized by structure.

	Robust04		*Gov2*		*ClueWeb-B*	
	ERR@20	**MAP**	**ERR@20**	**MAP**	**ERR@20**	**MAP**
WSD	12.04	27.41	16.52	29.36	**8.58**	14.56
$\mathcal{H}$-WSD	**12.34**$_w$	**27.79**$_w$ *(+1.4%)*	**16.56**	**29.82**$_w$ *(+1.6%)*	8.31	**14.68** *(+0.8%)*

(d) Weighted sequential dependence model (WSD) and its hypergraph representation ($\mathcal{H}$-WSD) parameterized by concept.

[BC12]

Query Hypergraphs

(a) *What is the effect of Turkish river control projects on Iraqi water resources?*

Local Factor Weights

(0.0315 effect) (0.0451 turkish) (0.0508 river) (0.0313 control)
(0.0263 projects) (0.0413 iraqi) (0.0387 water) (0.0344 resources)
(0.0079 "effect turkish") (0.0079 "turkish river") (0.0096 "river control")
(0.0079 "control projects") (0.0079 "projects iraqi")
(0.0079 "iraqi water") (0.0194 "water resources")

Global Factor Weights

(0.0203 effect 0.0262 turkish 0.0284 river 0.0164 control
0.0248 projects 0.0255 iraqi 0.0266 water 0.0252 resources
0.0014 "effect turkish" 0.0014 "turkish river" 0.0011 "river control"
0.0014 "control projects" 0.0014 "projects iraqi"
0.0014 "iraqi water" −0.0007 "water resources")

[BMC12]

Multiple Source Formulation

- Simultaneously perform query weighting and query expansion across multiple information sources to optimize
  - Effectiveness
  - Efficiency
  - Result diversity

[BMC12]

Multiple Source Formulation

Concept Weighting Sources (similar to WSD / PQE)

Source	Description
GF(κ)	Frequency of concept κ in Google n-grams
WF(κ)	Frequency of concept κ in Wikipedia titles
QF(κ)	Frequency of concept κ in a search log
CF(κ)	Frequency of concept κ in the collection
DF(κ)	Document frequency of concept κ
AP(κ)	A priori concept weight

[BMC12]

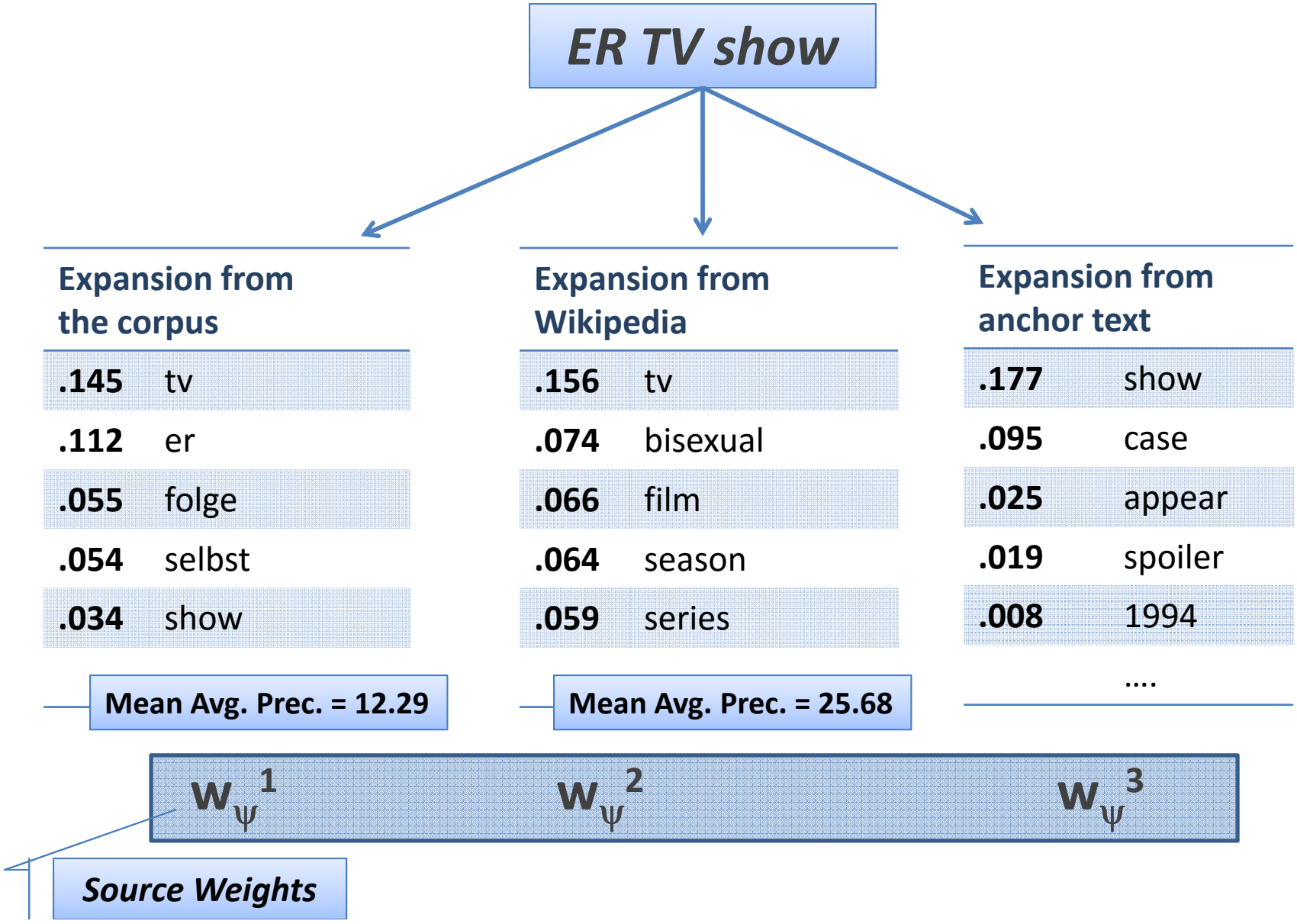
Multiple Source Formulation

Concept Expansion Sources

Source	Description
Web Headings	Text in the <h*> tags in HTML mark-up
Anchor Text	Text in the <a> tag in HTML mark-up
Wikipedia Corpus	Wikipedia articles
Retrieval Corpus	Documents in the retrieval corpus

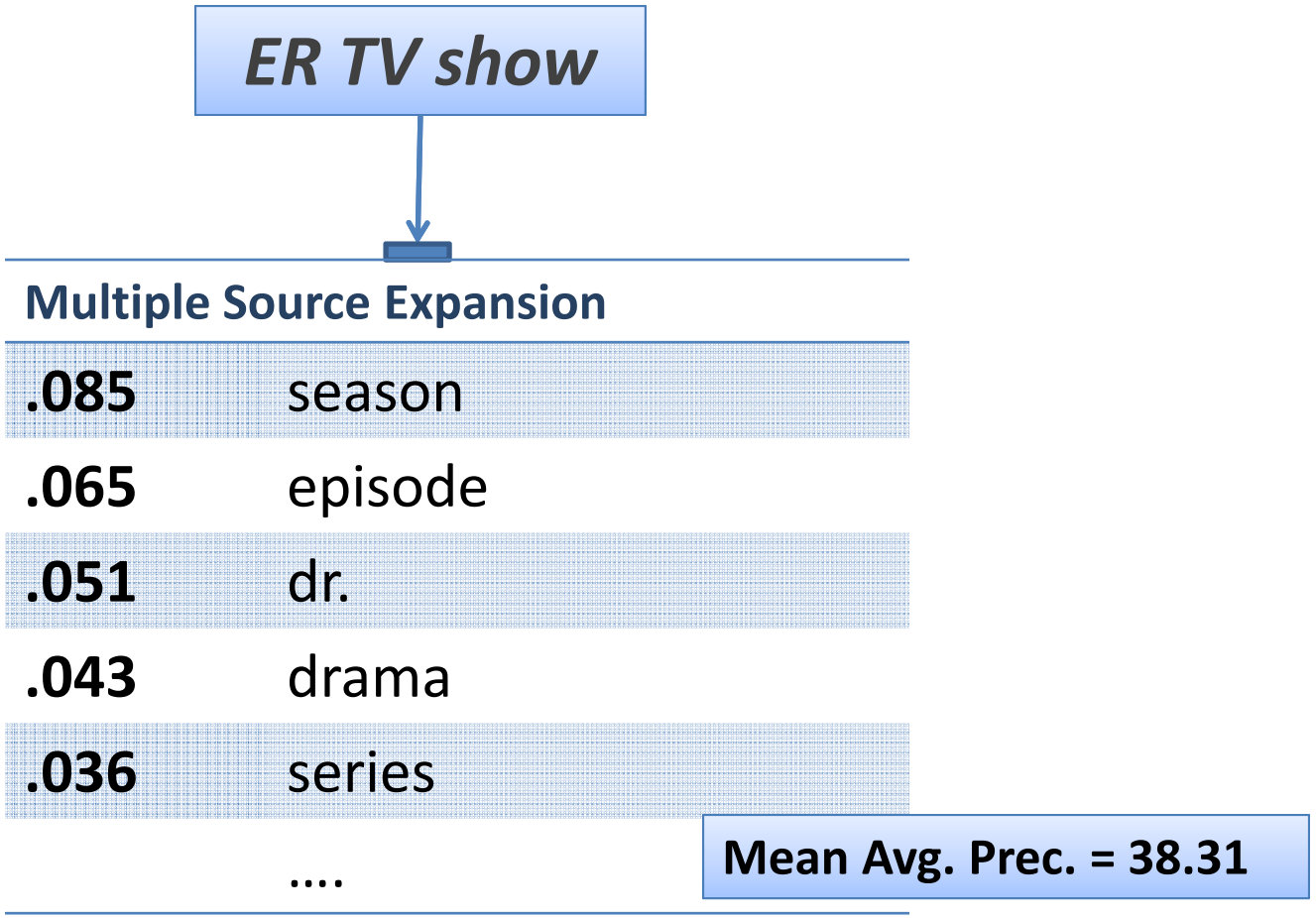
[BMC12]

Multiple Source Formulation



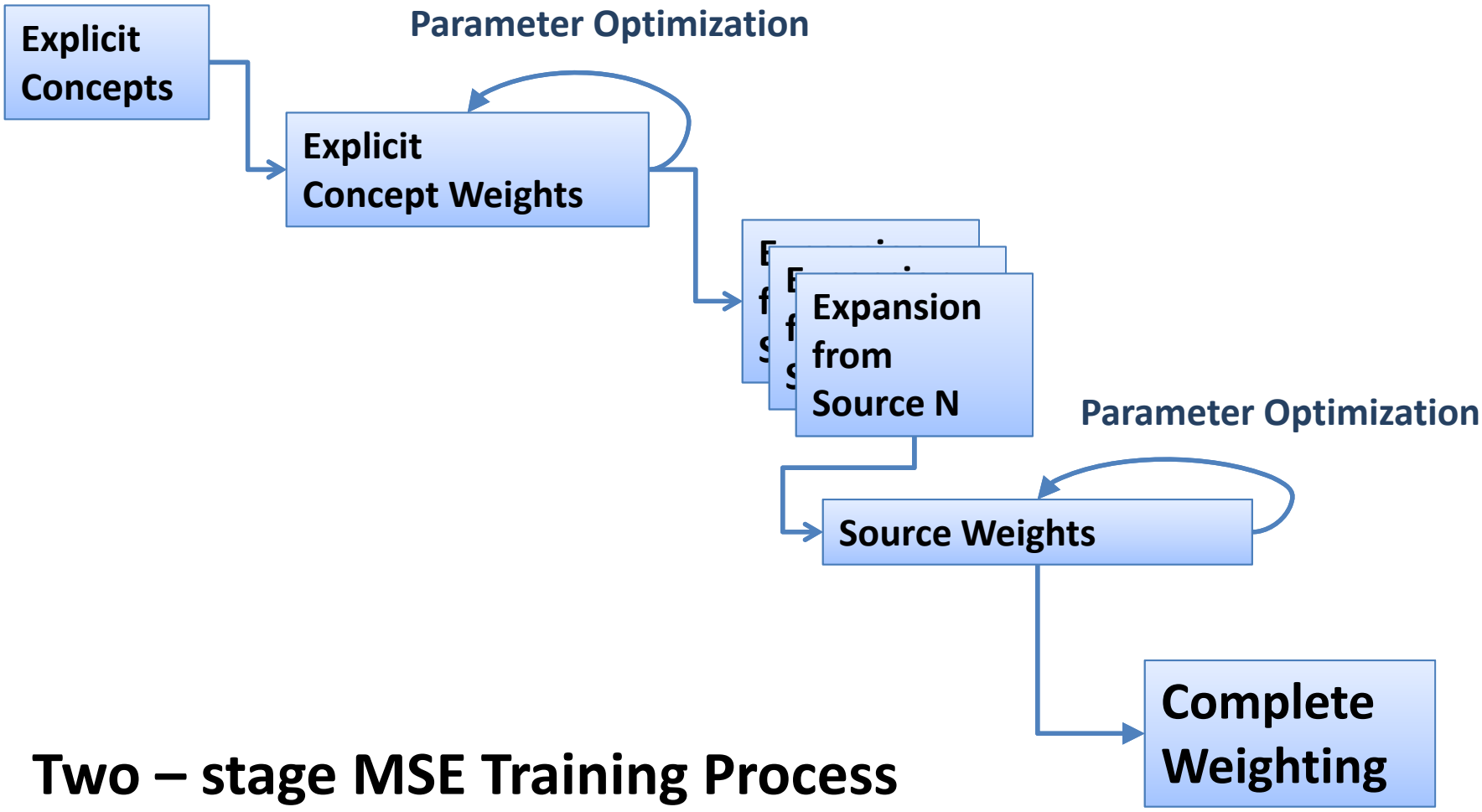
[BMC12]

Multiple Source Formulation



[BMC12]

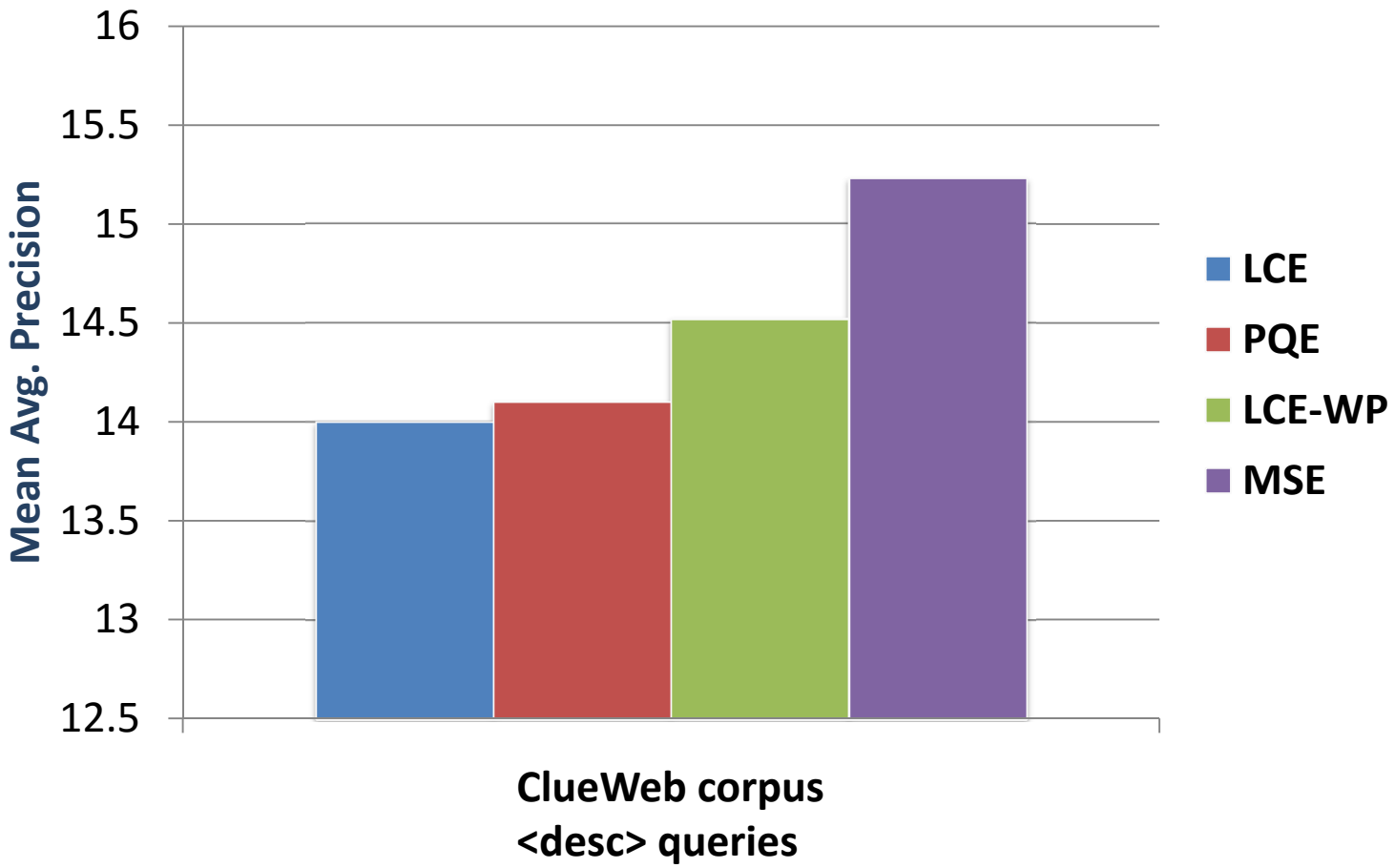
Multiple Source Formulation



Two – stage MSE Training Process

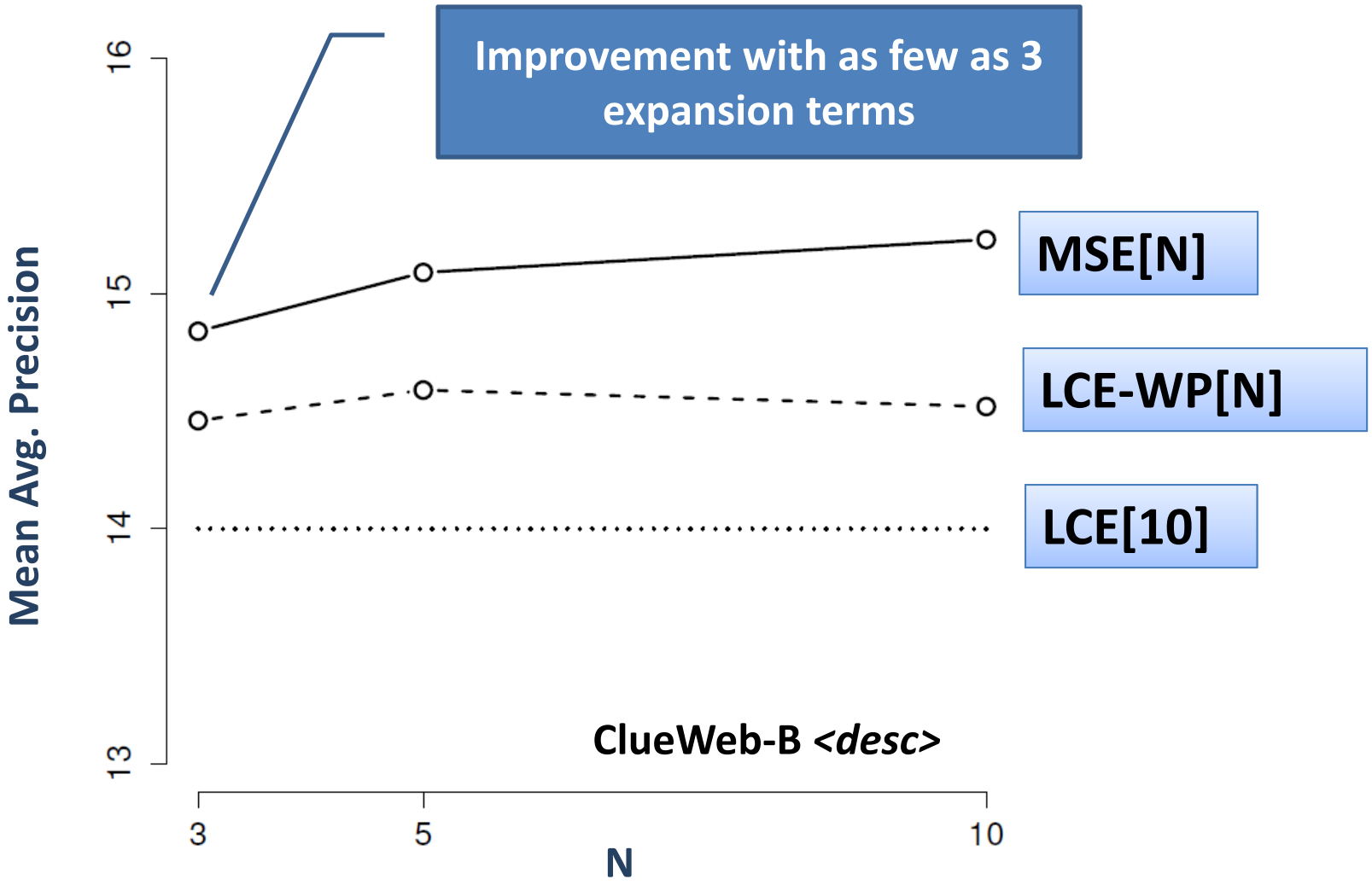
[BMC12]

Multiple Source Formulation



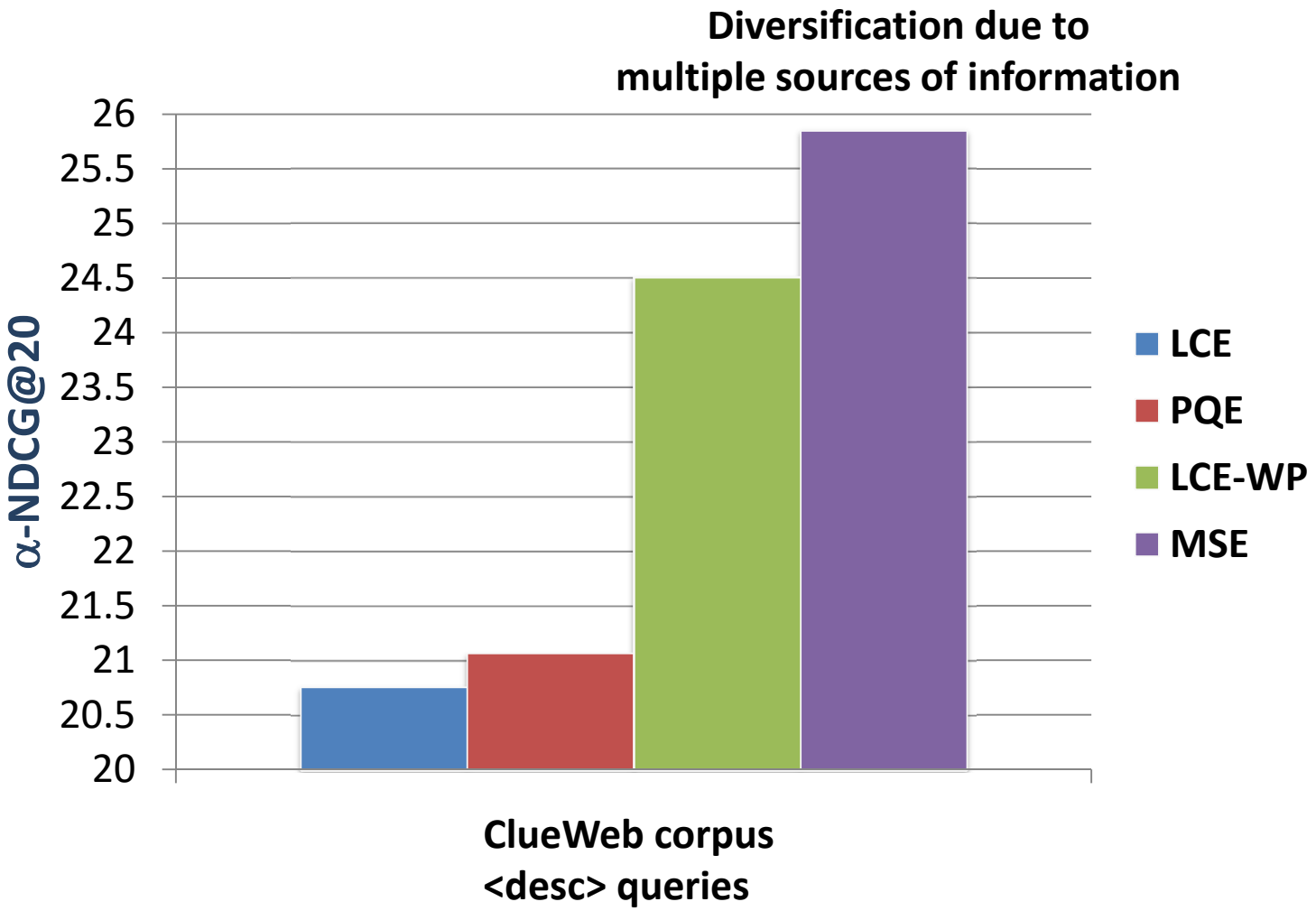
[BMC12]

Multiple Source Formulation



[BMC12]

Multiple Source Formulation



Tutorial Overview

- Properties of Verbose Queries [30 min]
- Techniques for Processing Verbose Web Queries
  - Query Reduction to a Single Sub-Query [50 min]
  - Query Reduction by Choosing Multiple Sub-Queries [10 min]
  - Break [30 min]
  - Weighting Query Words [90 min]
  - **Lunch Break [90 min]**
  - Query Expansion by Including Related Concepts [20 min]
  - Query Reformulation [30 min]
  - Query Segmentation [40 min]
  - Break [30 min]
  - Query-Dependent Learning to Rank [40 min]
- Applications of Verbose Query Processing [40 min]
- Summary and Future Research Directions [10 min]

Tutorial Overview

- Properties of Verbose Queries [30 min]
- Techniques for Processing Verbose Web Queries
  - Query Reduction to a Single Sub-Query [50 min]
  - Query Reduction by Choosing Multiple Sub-Queries [10 min]
  - Break [30 min]
  - Weighting Query Words [90 min]
  - Lunch Break [90 min]
  - **Query Expansion by Including Related Concepts [20 min]**
  - Query Reformulation [30 min]
  - Query Segmentation [40 min]
  - Break [30 min]
  - Query-Dependent Learning to Rank [40 min]
- Applications of Verbose Query Processing [40 min]
- Summary and Future Research Directions [10 min]

When could Query Expansion help?

- Query expansion (QE) does not work in all cases.
- How to define a measure to decide if QE will fail or succeed?
  - QE would be helpful when the initial set of top k retrieved documents (R) have a high precision
  - QE is useful when query is short.
- Use DFR (Divergence From Randomness) framework. Bose-Einstein distribution is one of the basic DFR models.
- $Info_{Bo2}(q_i) = -\log_2\left(\frac{1}{1+\lambda}\right) - tf_R(q_i).\log_2\left(\frac{\lambda}{1+\lambda}\right)$
  - Where $\lambda = |R|\frac{tf_C(q_i)}{|C|}$
  - When summed across all terms in query, it is significantly correlated to Average Precision (correlation=0.52)
- $Info(Q)$
  - $InfoPrior(Q) = \sum_{q_i \in Q} -\log_2 \frac{tf(q_i,C)}{tf(C)}$
  - $M_Q = \max\left(\frac{InfoPrior(Q) - \mu_{InfoPrior(Q)}}{\sigma_{InfoPrior(Q)}}, \max_{M \in DFR} \frac{Info_M(Q) - \mu_{Info_M(Q)}}{\sigma_{Info_M(Q)}}\right)$
  - $Info(Q) = \frac{1}{|Q|}\left(\frac{InfoPrior(Q) - \mu_{InfoPrior(Q)}}{\sigma_{InfoPrior(Q)}} + M_Q\right)$
- $Info(Q)$ is related to AP increase after QE activation
  - M denotes various DFR models like Divergence approximation of the binomial, Bose-Einstein distribution, Inverse Document Frequency model, etc.

When could Query Expansion help?

- In other words
  - The better is the initial retrieval stage / relevance feedback the more effective is the expanded query
  - Problematic for long queries where the initial set is likely to be noisy / non-relevant
  - One way to determine relevance is through query performance prediction metrics
    - Query clarity, weighted information gain, etc.

BWLK10

Adding ODP Category Label to Queries

- The proposed approach
  - leverages queries for which we can assign ODP labels
  - mines these queries for those that are similar to rare long queries of interest
  - propagates the labels from them to the rare long queries.
- For queries with successful URL trails:
  - assign a URL to one of the ODP categories using a back-off strategy. If URL does not match with an ODP category, remove successive path fragments from the end of URL until there is a match.
  - Thus a query is assigned a distribution over ODP categories. This is then called as aggregated labeled query trails (**alqts**)

BWLK10

Adding ODP Category Label to Queries

- The long query is matched to obtain multiple similar alqts to obtain ODP label for this long query. 4 matching algos
  - Term dropping: Drop least frequent term wrt query log.
  - Named entities: Match any past query with >1 named entity from long query
  - Language modeling: Long query is evaluated for match against all past queries to compute smoothed language model scores.
  - BM25 scoring
- All alqts for the long query are further aggregated to compute an alqt for long query.

BWLK10

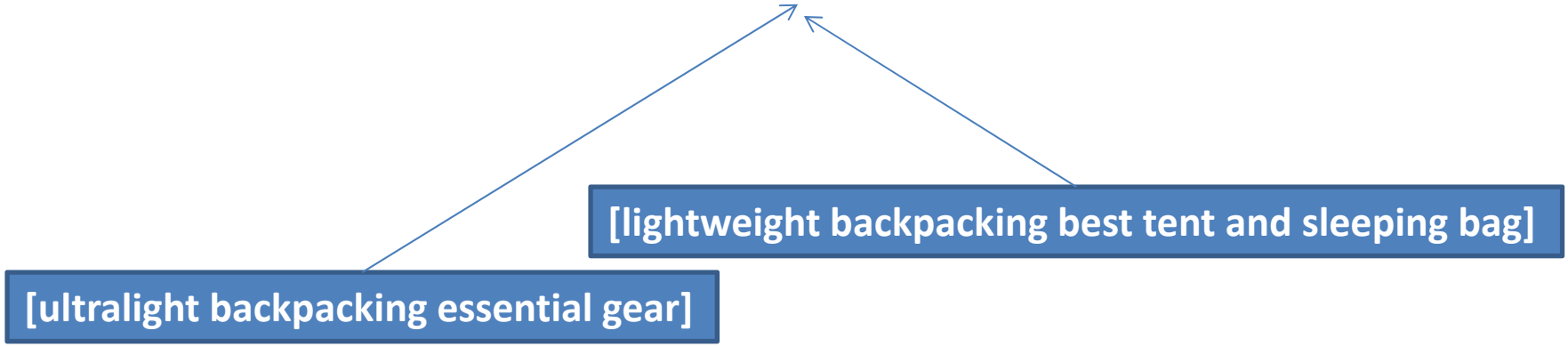
Adding ODP Category Label to Queries

Initial query: [ultralight backpacking]

Recreation/Outdoors/Hiking/Backpacking/Ultralight_Backpacking : 7

Recreation/Outdoors/Hiking/Organizations/North_America : 4

Recreation/Outdoors/Hiking/Trails : 2



BWLK10

Adding ODP Category Label to Queries

- Coverage and Relevance of Query Label Prediction Using Different Similarity Matching Algorithms

Algorithm	Cov(%)	P@1(top)	P@1(any)	P@3	MRR	nDCG	F1
Term dropping	49.7	15.5	19.4	25.8	22.0	17.4	17.0
Named entities	36.6	13.2	16.7	23.1	21.0	15.0	14.6
LM (T=3)	100	9.6	12.1	17.7	13.9	11.1	10.2
LM (T=10)	100	12.3	15.2	23.1	19.5	14.2	14.0
BM25 (T=3)	100	19.5	24.4	33.3	28.1	22.0	23.1
BM25 (T=5)	100	22.3	26.1	35.6	32.0	24.2	25.8
BM25 (T=10)	100	19.3	23.9	33.5	29.2	21.5	23.2
Standard errors		≤ 0.02	≤ 0.02	≤ 0.02	≤ 0.03	≤ 0.03	≤ 0.04

Method	nDCG@1	nDCG@3	nDCG@10
BM25F (baseline)	43.7	45.4	46.7
Click	**45.5**	**46.5**	47.2
ODP	**46.8**	**47.9**	**48.9**
BM25F+Click	**46.7**	**47.0**	47.1
BM25F+ODP	45.3	46.2	47.9
Click+ODP	**47.7**	**47.8**	**48.2**
All (BM25F+ODP+Click)	**46.4**	**47.7**	**49.0**

BW+08

Expansion using User-Supplied Reference Documents

- User information needs are expressed using a short query (of a few keywords) together with examples of key reference pages.
- For query likelihood model based document ranking, one computes $P(q_i|\theta_D) = (1-\lambda)P(q_i|D) + \lambda P(t|C)$
- Reference documents S can be used to influence setting of λ
  - Maximizing Average Precision assuming that *S* is the only set of relevant documents given query *Q*.
  - Maximizing Query Log Likelihood of the query *Q*, given the set of sample documents *S*.
- Reference documents S can be used to influence the pseudo-relevance feedback based query expansion
  - $P(t|S) = \sum_{D \in S} P(t|D)P(D|S)$
  - $P(t|D)$ can be computed using
    - Maximum likelihood estimate: $P(t|D) = P_{ML}(t|D)$
    - Smoothed estimate of a word: $P(t|D) = (1-\lambda)\, P_{ML}(t|D) + \lambda\, P_{ML}(t|C)$
    - Using unsupervised query expansion: $s(t) = \log \frac{P_{ML}(t|D)}{P_{ML}(t|C)}$ and $P(t|D) = \frac{s(t)}{\sum_{t'} s(t')}$
  - $P(D|S)$ can be computed as follows
    - Uniform: $P(D|S) = 1/|S|$
    - Query biased: $P(D|S) \propto P(D|Q)$
    - Inverse query-biased: $P(D|S) \propto 1 - P(D|Q)$

XU09

Query Dependent Pseudo-Relevance Feedback based on Wikipedia

1. Instead of retrieving / marking up a document set, rely on a single authoritative Wikipedia page

2. Expand the query using the terms from this page

3. Works best for queries that can be easily mapped to a single entity: “barack obama family tree”

4. Classify queries into three categories using Wikipedia itself
   - **EQ** entity queries
   - **AQ** ambiguous queries
   - **BQ** broad queries (the rest)

5. Use a separate ranking function for each category

XU09

Query Dependent Pseudo-Relevance Feedback based on Wikipedia

Improvement for EQ queries

Method	AP	Robust	WT10G	GOV2
QL	0.2208	0.3156	0.2749	0.3022
RMC	0.2484	0.3401	0.2458	0.3168
RMW	0.2335	0.3295	0.2821*	0.3453*
RE	0.2494**	0.3580**	0.2897*	0.3889**

Improvement for AQ queries

Method	AP	Robust	WT10G	GOV2
QL	0.1391	0.2258	0.1801	0.2892
RMC	0.1520	0.2485	0.1881	0.3101
RMW	0.1588	0.2619*	0.1868	0.3186*
RE	0.1692**	0.2728**	0.2037**	0.3329**

XU09

Query Dependent Pseudo-Relevance Feedback based on Wikipedia

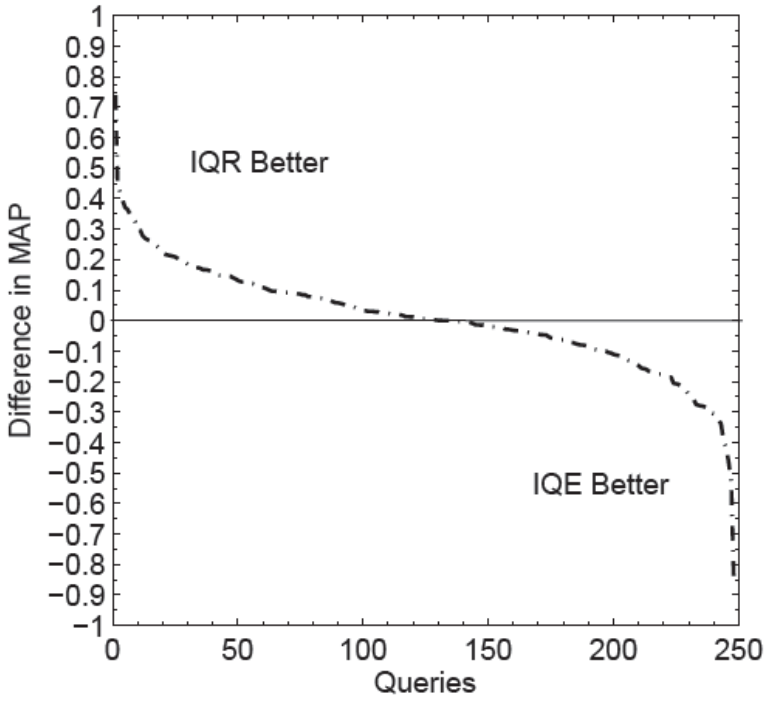
Improvement for all query types

		AP	Robust	WT10G	GOV2
RMC	MAP	0.1707	0.2823	0.1969	0.3141
	IMP	119	174	47	88
QD	MAP	0.1777	0.3002*	0.2194*	0.3348*
	IMP	116	191	67	96

Table 9: Query Dependent vs traditional Relevance Model. IMP is the number of queries improved by the method over QL. * indicates statistically significant improvements over RMC. We use the paired t-test with significant at $p<0.05$.

[KA08]

When could Query Expansion help?



Some queries are better suited for IQR, while others can be better improved through IQE.

Selective Interactive Reduction & Expansion

Query expansion works best when complemented with query reduction

System	P@5	P@10	NDCG@15	MAP	Avg. num. options
Baseline	0.472	0.397	0.379	0.240	-
IQR_5	0.554	0.469	0.448	0.274	4.9
IQE_5	0.555	0.466	0.435	0.292	5
$SIRE_{comb}$	0.65	0.552	0.521	0.347	9.9
IQR_{10}	0.634	0.528	0.498	0.300	9.7
IQE_{10}	0.571	0.480	0.450	0.292	10.0

Letting the user choose the best option among reduction / expansion leads to best results (in an ideal scenario)

LI12

QRU-1: A Public Dataset for Promoting Query Representation and Understanding Research

Even bigger gains can be achieved (in upper bound) for arbitrary query reformulations

```
Topic #1: obama family tree
--------
barack obama family
obama family
obama s family
barack obama family tree
the obama family
barack obama s family
obamas
obama genealogy
barack obama s family tree
barack obama ancestry
president obama s family
obamas family
obama family history
obama s family tree
barack obama genealogy
barack obama family history
```

```
Topic #95: earn money at home
---------
earn money from home
earn money at home
how to earn money at home
earn money on the internet
ways to earn money at home
how to earn money from home
earn extra money at home
earning money from home
earn extra cash at home
earning money at home
earn at home
earn money working from home
earn money from home free
how to earn money on the internet
earn cash at home
earn currency at home
```

QRU-1: A Public Dataset for Promoting Query Representation and Understanding Research

Even bigger gains can be achieved (in upper bound) for arbitrary query reformulations

SD	MAP	NDCG@20	ERR@20
Baseline metric	19.13	20.19	8.34
Best metric	25.00 *(+30.7%)*	32.88 *(+62.9%)*	15.09 *(+80.9%)*
% outperforming queries	12%	16%	18%
% topics improved	51%	63%	67%

Topic Title #1	*obama family tree*	**ERR@20**	*13.42*
Reformulations	barack obama ancestry		32.93
	obama s family		32.40
	barack obama s family		32.05

Reliably identifying these winning cases is still an open research problem

Tutorial Overview

- Properties of Verbose Queries [30 min]
- Techniques for Processing Verbose Web Queries
  - Query Reduction to a Single Sub-Query [50 min]
  - Query Reduction by Choosing Multiple Sub-Queries [10 min]
  - Break [30 min]
  - Weighting Query Words [90 min]
  - Lunch Break [90 min]
  - Query Expansion by Including Related Concepts [20 min]
  - **Query Reformulation [30 min]**
  - Query Segmentation [40 min]
  - Break [30 min]
  - Query-Dependent Learning to Rank [40 min]
- Applications of Verbose Query Processing [40 min]
- Summary and Future Research Directions [10 min]

Query Reformulation

- Query reduction and query weighting methods almost completely ignore user interaction with these queries, as can be evidenced from search logs, or other sources.
- “Find information about Mitchell College in Connecticut’’
  - “mitchell college new london” – expansion and reduction applied
  - “mitchell college new london ct” – abbreviation induction applied
  - “mitchell college mitchell.edu” – relevant URL added

XJC08

Translation-based Language Model

- Finding answer to a question from Question-answer archives.
- The proposed retrieval model combines a word-to-word translation-based language model for the question part with a query likelihood approach for the answer part.
- Major problem: word mismatch between the user's question and the question-answer pairs in the archive.
  - For example, "what is francis scott key best known for?" and "who wrote the star spangle banner?" are two very similar questions, but they have no words in common.

$$P(q_u|(q,a)) = \prod_{q_i \in q_u} P(q_i|(q,a)) \qquad P(q_i|(q,a)) = \frac{|(q,a)|}{|(q,a)|+\lambda} P_{mx}(q_i|(q,a)) + \frac{\lambda}{|(q,a)|+\lambda} P_{ml}(q_i|C)$$

$$P_{mx}(q_i|(q,a)) = (1-\beta) P_{ml}(q_i|q) + \beta \sum_{t \in q} P(q_i|t) P_{ml}(t|q)$$

Incorporating the answer part

$$P_{mx}(q_i|(q,a)) = \alpha P_{ml}(q_i|q) + \beta \sum_{t \in q} P(q_i|t) P_{ml}(t|q) + \gamma P_{ml}(q_i|a)$$

$$\alpha + \beta + \gamma = 1$$

XJC08

Translation-based Language Model

- Learning word-word translation prob.
  - Translation from English word e to French word f.
  - Parallel corpus: $S = \{(\boldsymbol{e_1}, \boldsymbol{f_1}), (\boldsymbol{e_2}, \boldsymbol{f_2}), \ldots, (\boldsymbol{e_n}, \boldsymbol{f_n})\}$
  - EM algorithm
    - $P(f|e) = \lambda_e^{-1} \sum_{i=1}^{N} c(f|e; \boldsymbol{f_i}, \boldsymbol{e_i})$
      - λ_e is a normalization factor across all f's
    - $c(f|e; \boldsymbol{f_i}, \boldsymbol{e_i}) = \frac{P(f|e)}{P(f|e_1)+\cdots+P(f|e_l)} \#(f, \boldsymbol{f_i}) \#(e, \boldsymbol{e_i})$
  - For Q&A archives, either question or answer could be the source and target leading to P(A|Q) or P(Q|A).
  - The 2 models could then be combined in 2 ways
    - Linear: $P_{lin}(w_i|w_j) = (1-\delta)P(w_i, Q|w_j, A) + \delta P(w_i, A|w_j, Q)$
    - Pooling: The dataset $\{(q,a)_1, \ldots, (q,a)_n, (a,q)_1, \ldots (a,q)_n\}$ is used with the EM to compute $P_{pool}(w_i|w_j)$

XJC08

Translation-based Language Model

Source	everest			xp			search		
TTable	$P(A\|Q)$	$P(Q\|A)$	P_{pool}	$P(A\|Q)$	$P(Q\|A)$	P_{pool}	$P(A\|Q)$	$P(Q\|A)$	P_{pool}
1	everest	mountain	everest	xp	xp	xp	search	search	search
2	29,035	tallest	mountain	drive	window	window	google	information	google
3	ft	everest	tallest	install	computer	install	page	website	information
4	mount	highest	29,035	click	system	drive	list	free	internet
5	8,850	mt	highest	system	pc	computer	engine	info	website
6	feet	discover	mt	window	version	system	internet	internet	web
7	measure	hillary	ft	computer	edition	click	click	web	list
8	expedition	edmund	measure	pc	install	pc	web	address	free
9	height	mountin	feet	program	software	program	information	picture	info
10	nepal	biggest	mount	microsoft	98	microsoft	result	online	page

	Wondir	
Trans Prob	MAP	P@10
$P(A\|Q)$	0.4059	0.2684
$P(Q\|A)$	0.379	0.2658
P_{lin}	0.4149	0.2842
P_{pool}	0.4238	0.2868

Word-word translation prob. accuracy

	Wondir	
Model	MAP	P@10
LM-Comb	0.3791	0.2368
TransLM	0.4238	0.2868
TransLM+QL	0.4885	0.3053

LM-Comb combines the LM for questions with LM for the answers.
TransLM does not consider the answer part when performing translation, while TransLM+QL does.

Random Walks
Query Log-based Term-Query Graph

- Term-Query Graph consists of terms and queries
  - Link a term to a query if it appears in the query
  - Link query q_2 from q_1 iff the likelihood of query q_2 appearing after q_1 is not null.
- Query suggestions for a query $q = \{t_1, \ldots, t_m\}$ are generated as follows
  - Compute m random walks from each term in q.
  - Compute Hadamard product of these m vectors to get a resultant score for every other query in the graph.
  - Return top ranking queries as suggestions.

Query: Social Katz index

Suggested Query	Score
katz index of activities of daily living	$2.7\ e^{-09}$
modified barthel index	$8.6\ e^{-13}$
barthel index score	$6.4\ e^{-13}$
modified barthel index score	$8.8\ e^{-14}$
youtube	$4.3\ e^{-19}$

Query: Menu restaurant design

Suggested Query	Score
free restaurant design software	$4.9\ e^{-12}$
free restaurant kitchen layout and design	$4.8\ e^{-12}$
restaurant menu design	$4.7\ e^{-12}$
restaurant menu design software	$4.7\ e^{-12}$
restaurant menu design samples	$4.4\ e^{-12}$

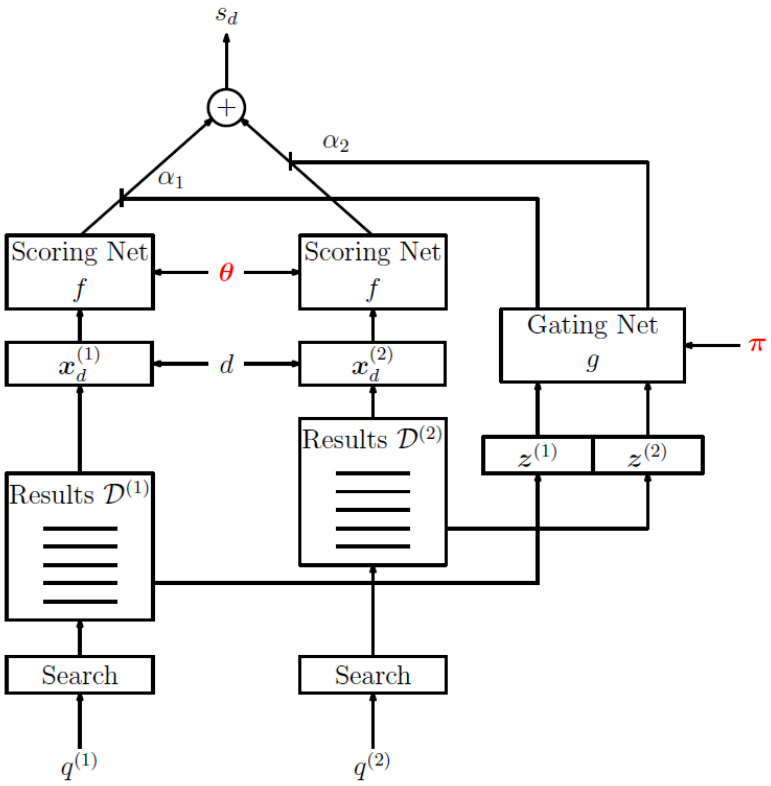
Query: Bill Clinton

Suggested Query	Score
president bill clinton speeches	$8.8\ e^{-11}$
famous bill clinton quotes	$8.4\ e^{-11}$
monica lewinsky and bill clinton scandal	$8.4\ e^{-11}$
bill clinton foundation website	$8.3\ e^{-11}$
former president bill clinton biography	$8.1\ e^{-11}$

Random Walks
LambdaMerge with Click Graph

- A supervised merging method to merge results from several reformulations generated by random walks on a click graph.
  - Directly optimizes a retrieval metric using features that describe both the reformulations and the documents they return.
  - Features: document quality (retrieval score), score distribution for the result list, overlap between result lists.
  - CombSum uses random walk probabilities as merging weights.

Random Walks
LambdaMerge with Click Graph



LambdaMerge Architecture

- For query q, let $q^{(1)}, q^{(2)}, \ldots, q^{(K)}$ be the different reformulations and let $D^{(1)}, D^{(2)}, \ldots, D^{(K)}$ be the corresponding result lists.
- Let $NormScore\left(q^{(k)}, d\right)$ be relevance score of d for $q^{(k)}$ between 0 and 1.
- $CombSum(d) = \sum_{k:d\in D^{(k)}} NormScore(q^{(k)}, d)$
- $CombRW(d) = \sum_{k:d\in D^{(k)}} NormScore\left(q^{(k)}, d\right) \times W(D^{(k)})$
  - Where $W(D^{(k)})$ is prob assigned to $q^{(k)}$ by a random walk in query-URL graph starting at q.
- Supervised merging: LambdaMerge provides a flexible class of gated neural-network score-combination functions that
  - (1) utilize multiple query-document features instead of just the retrieval score,
  - (2) weight contributions from each reformulation according to multiple features, such as those predicting list quality and query drift, so that contributions vary depending on the query and reformulation, and
  - (3) are trained to optimize a retrieval metric.
- 2 types of features
  - Query-doc features
  - Gating features that describe quality of reformulation and its results list (e.g. query clarity score).

Random Walks
LambdaMerge with Click Graph

- $x_d^{(k)}$ is a vector of query-doc features for doc d and k^{th} query reformulation $q^{(k)}$.
- Let $f(x;\theta)$ be scoring function with parameters θ.
- $z^{(k)}$ is a vector of gating features for reformulation $q^{(k)}$ which describe qualities of the k^{th} reformulation and its result list $D^{(k)}$
- Gating network determines contribution of each reformulation to final score. $\alpha_k = softmax\left(z^{(1)}, z^{(2)}, \dots, z^{k}; \pi\right) = \frac{\exp(\pi^T z^{(k)})}{\sum_p \exp(\pi^T z^{(p)})}$ for the k^{th} result list
  - Mixing weights are normalized to sum to 1
- Final score s_d for doc is given by $s_d = \sum_k \alpha_k . f(x_d^{(k)}; \theta)$
  - f could be any differentiable function like a linear function, a neural network, or a set of boosted decision trees.
  - They use a 2 layer neural network with tanh activation functions.
  - Scoring parameters θ and π are trained to optimize NDCG
    - Some tricks to use a smoothed version of NDCG so that gradient descent can be used effectively.

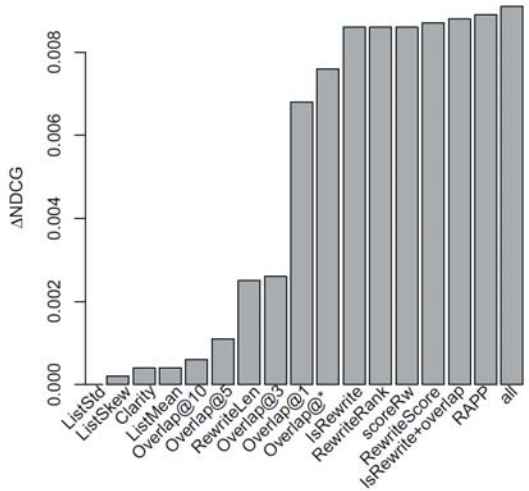
Random Walks
LambdaMerge with Click Graph

- Query-doc features
  - Bing Ranker Score
  - Rank
  - $NormScore_{[0,1]}$ is score normalized using min-max normalization
  - $NormScore_{\mathcal{N}(0,1)}$ is score normalizaed to fit a standard Gaussian
  - isTopN is binary feature to indicate if doc is within top 1,3,5, 10
- Gating Features
  - isRewrite: distinguishes between original query and reformulation
  - RewriteScore: Random walk based reformulation score
  - RewriteRank, ListMeanm ListStd, ListSkew, RewriteLen
  - Clarity: KL divergence between query language model (over top 10 snippets) and collection model
  - Overlap@N: overlap in top 10 docs for original query vs reformulated query
  - RAPP: Linear regression model score trained to predict NDCG using above features.
- Baselines
  - ORG: original query
  - CombSUM
  - RAPP-L: Linear regression trained to predict $\Delta NDCG$
  - RAPP-Ω: Oracle that chooses a single reformulation that gives highest NDCG@5

Query-document features	Score, Rank, $\text{NormScore}_{[0,1]}$, $\text{NormScore}_{\mathcal{N}(0,1)}$, IsTopN
Gating features (difficulty)	ListMean, ListStd, ListSkew, Clarity, RewriteLen, RAPP
Gating features (drift)	IsRewrite, RewriteRank, RewriteScore, Overlap@N

	1 Reformulation		5 Reformulations	
	NDCG@5	NDCG@10	NDCG@5	NDCG@10
ORG	0.550	0.535	0.550	0.535
CombSUM	0.521	0.496	0.497	0.455
CombRW	0.554	0.527	0.555	0.524
RAPP-L	0.546	0.531	0.550	0.535
λ-Merge	0.566	0.550	0.563	0.548
RAPP(Ω)	0.567	0.541	0.587	0.545

The performance of different techniques averaged over a subset of queries with at least 5 random-walk candidates (1872 queries).



Retrieval accuracy for various combinations of gating features

XTJL12

Question Reformulation Patterns from Query Logs

- [WZ08] extract term associations based on their context distribution in queries in query log. For a new query, the method will decide whether to substitute a term with one of its "similar" words based on whether this new word matches the context of the query better than the original term.
- Context sensitive stemming based on query logs is another type of query reformulation. [PALL07]
- Mine 5w1h question reformulation patterns from query logs.

Alternative expressions for the original question

Original Question:
how far is it from Boston to Seattle
Alternative Expressions:
how many miles is it from Boston to Seattle
distance from Boston to Seattle
Boston to Seattle
how long does it take to drive from Boston to Seattle

Question reformulation patterns generated for the query pair ("how far is it from Boston to Seattle","distance from Boston to Seattle").

S_1 = {Boston}:("how far is it from X_1 to Seattle","distance from X_1 to Seattle")
S_2 = {Seattle}:("how far is it from Boston to X_1","distance from Boston to X_1")
S_3 = {Boston, Seattle}:("how far is it from X_1 to X_2","distance from X_1 to X_2")

- Extract (q, q_r) pairs from query log such that they are issued by same user one after the other within a certain time period, and q is a 5w1h query.
- Patterns are extracted from pairs with many common words and that occur with a high frequency.
- Given a new query q^{new} pick best question pattern p^* according to #prefix words and total number of matching words in pattern. Query reformulations are ranked by frequency

$$P(p_r|p^\star) = \frac{f(p^\star, p_r)}{\sum_{p'_r} f(p^\star, p'_r)}$$

XTJL12

Question Reformulation Patterns from Query Logs

- Retrieval model
  - $score(q^{new}, D) =$
    $\lambda \log P(q^{new}|D) + (1-\lambda) \sum_{i=1}^{k} P(p_{r_i}|p^*) \log P(q_{r_i}^{new}|D)$

Examples of the question reformulations and their corresponding reformulation patterns

q^{new}: how good is the eden pure air system $p^\star$: how good is the X		q^{new}: how to market a restaurant $p^\star$: how to market a X	
q_r^{new}	p_r	q_r^{new}	p_r
eden pure air system	X	marketing a restaurant	marketing a X
eden pure air system review	X review	how to promote a restaurant	how to promote a X
eden pure air system reviews	X reviews	how to sell a restaurant	how to sell a X
rate the eden pure air system	rate the X	how to advertise a restaurant	how to advertise a X
reviews on the eden pure air system	reviews on the X	restaurant marketing	X marketing

	NDCG@1	NDCG@3	NDCG@5
Orig	0.2946	0.2923	0.2991
QDist	0.3032*	0.2991*	0.3067*

Table shows that using the question reformulations can significantly improve the retrieval performance of natural language questions. Note that, considering the scale of experiments (10,000 queries), around 3% improvement with respect to NDCG is a very interesting result for web search.

DC10

Query Reformulations using Anchor Text

- In absence of query logs, anchor text is a good approximation. [DC10] method is very similar to [WZ08]
- Estimate context distribution for all words in query log.
  - $count_w(c_i)$ is #times word c_i occurs in C context of w
  - Smoothed prob is $\tilde{P}_C(c_i|w) = \frac{count_w(c_i)+\mu P(c_i|\theta)}{\sum_{c_j \in C(w)} count_w(c_j)+\mu}$
- Learn a translation model to translate from one word to another based on their distributional similarity.

$$t(s|w) = \frac{e^{-D(P_C(.|w)||\tilde{P}_C(.|s))}}{\sum_u e^{-D(P_C(.|w)||\tilde{P}_C(.|u))}}$$

$$D(P_C(.|w)||\tilde{P}_C(.|s)) = \sum_{c \in C(w)} P(c|w) log \frac{P(c|w)}{\tilde{P}_C(c|s)}$$

Query Reformulations using Anchor Text

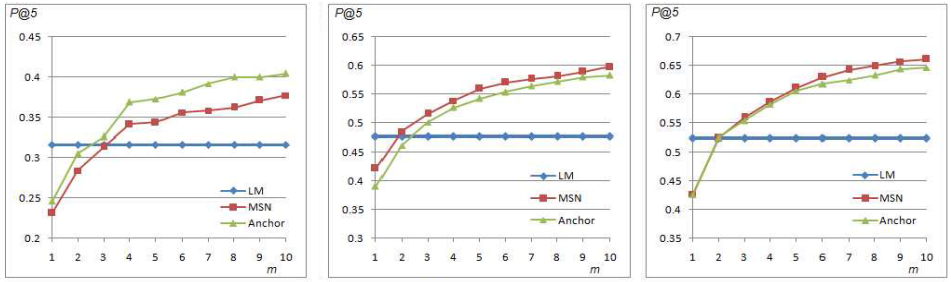
- A substitution model is built on top of the translation model.

$$P(w_i \rightarrow s|q) \propto t(s|w_i) \times P(w_1 \ldots w_{i-1}\_w_{i+1} \ldots w_n|s)$$

$$P(w_1 \ldots w_{i-1} w_{i+1} \ldots w_n|s)$$
$$= \tilde{P_{L_2}}(w_{i-2}|s) \times \tilde{P_{L_1}}(w_{i-1}|s) \times \tilde{P_{R_1}}(w_{i+1}|s) \times \tilde{P_{R_2}}(w_{i+2}|s)$$

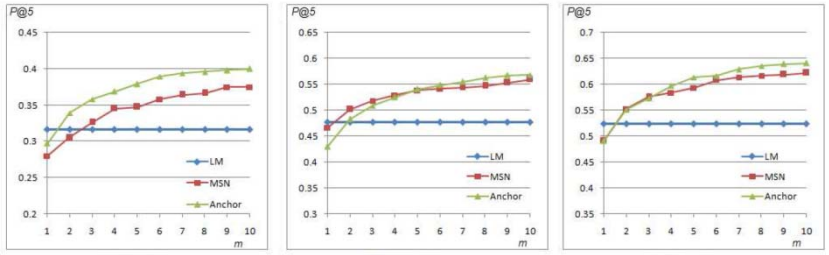
- Given a query term and a query context, substitution model decides whether to make a substitution, and if so, which candidates among those suggested by the translation model should be used.
  - For each w_i in query, try to replace each one of them
    - Consider only top M translation candidates s_i sorted by $t(s_i|w_i)$
    - Remove all s_i that have $NMI(s_i, w_i) < \tau$ where $NMI(s, w) = \frac{MI(s,w)}{MI(w,w)}$ and MI is computed over query log sessions.
    - Makes substitutions if $\frac{P(w_i \rightarrow s_i|q)}{P(w_i \rightarrow w_i|q)} > 1$
- Besides substitution one could also do expansion

Query Reformulations using Anchor Text



(a) WT10G (b) Robust04 (c) Gov-2

Query substitution results (P@5) with long queries on three collections



(a) WT10G (b) Robust04 (c) Gov-2

Query expansion results (P@5) with long queries on three collections

The best P@5 obtained by these m candidates is recorded as the P@5 of the substitution solution for q. We varied m from 1 to 10 in our experiment.

Query Context	Reformulation
_ cancer treatments	prostate → breast
_ cancer treatments	prostate → lung
best retirement _	country → state
_ security	airport → museum
_ school success	magnet → charter
pearl _	farming → planting
_ world war ii	portugal → soccer
controlling acid _	rain → plant
_ edwards womens issues	john → james

Bad substitutions

Examples of good expansion

	Original Query	Expanded Query	Original Query	Expanded Query
MSN Log	hunting deaths	hunting #syn(deaths accidents)	railway accidents	#syn(railway train) accidents
	new fuel sources	new #syn(fuel energy) sources	oscar winner selection	oscar winner #syn(selection promotion)
	educational standards	#syn(educational teaching) standards	marine vegetation	marine #syn(vegetation plants)
	automobile recalls	#syn(automobile auto) recalls	overseas tobacco sales	overseas #syn(tobacco cigarettes) sales
	doctor assisted suicides	#syn(doctor physicians) assisted suicides	food drug laws	food drug #syn(laws act)
	cheese production	cheese #syn(production companies)	volkswagen mexico	#syn(volkswagen vw) mexico
	illegal immigrant wages	illegal immigrant #syn(wages working)	chevrolet trucks	#syn(chevrolet chevy) trucks
Anchor Log	hunting deaths	hunting #syn(deaths accidents)	railway accidents	#syn(railway railroad) accidents
	new fuel sources	new #syn(fuel energy) sources	pearl farming	pearl #syn(farming industry)
	educational standards	#syn(educational teaching) standards	eskimo history eskimo	#syn(history culture)
	automobile recalls	#syn(automobile auto)	international art crime	international art #syn(crime fraud)
	doctor assisted suicides	#syn(doctor physicians) assisted suicides	wildlife extinction	#syn(wildlife animals) extinction
	cheese production	cheese #syn(production prices)	blood alcohol fatalities	blood alcohol #syn(fatalities deaths)
	illegal immigrant wages	illegal #syn(immigrant workers) wages	windmill electricity	windmill #syn(electricity power)

Tutorial Overview

- Properties of Verbose Queries [30 min]
- Techniques for Processing Verbose Web Queries
  - Query Reduction to a Single Sub-Query [50 min]
  - Query Reduction by Choosing Multiple Sub-Queries [10 min]
  - Break [30 min]
  - Weighting Query Words [90 min]
  - Lunch Break [90 min]
  - Query Expansion by Including Related Concepts [20 min]
  - Query Reformulation [30 min]
  - **Query Segmentation [40 min]**
  - Break [30 min]
  - Query-Dependent Learning to Rank [40 min]
- Applications of Verbose Query Processing [40 min]
- Summary and Future Research Directions [10 min]

Segmentation of Verbose Queries

- A verbose query often contain multiple pieces of information.
- Rather than reducing or expanding the whole query, it might make more sense to split the query into multiple segments and then process each segment separately.
- Consider a long query, like new ac adapter and battery charger for hp pavilion notebook; it often suffers from zero-recall on ebay, a large e-Commerce marketplace but breaking the query into meaningful phrases, such as {new | ac adapter | and | battery charger | for | hp pavilion notebook} helps in reformulating the query as new battery charger hp pavilion; this shortened query retrieves more than four thousand products. [PSAH13]
- In this section, we will study various ways of segmenting long queries.
- 3 broad types
  - Statistical methods: term freq based or based on mutual information between terms. [PSAH13]
  - Supervised and NLP based approaches: Given query with n terms, whether to partition at position i can be modeled as a binary classification problem [BCS11, BW07]
  - IR based method: [MSRG+11] use bag of words model as null model, and use this null model to find the prob of a multi-word expression (MWE) to be a phrase. Given a set of n queries where each query contains each of the words of w, the method finds the number of queries m in this set that contain the MWE. MWE score is computed as $-\log \delta$ where δ is Hoeffdings' upper bound on prob that using null model $\geq$m queries may contain w as MWE.

Segmentation of Verbose Queries

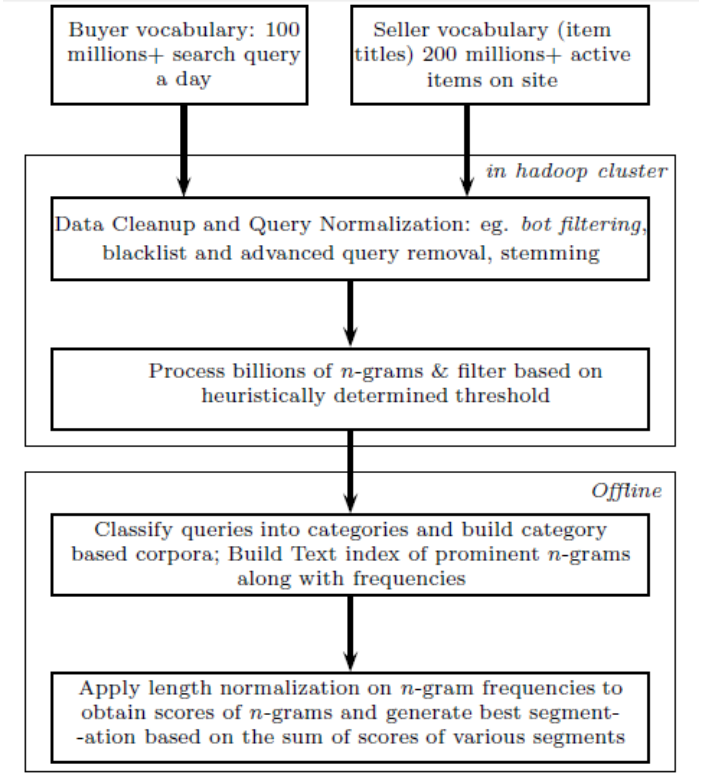
- Consider a query $\mathrm{Q} = \{q_1, q_2, \ldots, q_N\}$ consisting of N query tokens.
- Segmentation is a mapping $S: Q \rightarrow y \in Y_N$, where y is a segmentation from the set Y_N.
- Since we can either have or not have a segmentation break at each of the $N - 1$ spaces between N tokens, $|Y_N| = 2^{N-1}$
- Methods:
  - Statistical
  - Supervised
  - Generative
  - NLP-based

Statistical Methods for Segmentation

- Based on term frequency or on mutual information (MI) between terms
- [JRMG06] uses MI between pairs of tokens: If MI is below a threshold, insert a segmentation break.
- [RMB03] combined frequency count of a segment with MI
- [MRGL+11] compute prob of a multi-word expression (MWE) to be a phrase.

Statistical Segmentation using Category-wise n-Gram Frequencies

- QSegment focuses on domain-specific segmentation of e-commerce queries using frequency data from query log and product titles.
- Recent efforts use Wikipedia titles to find noun phrases and then perform segmentation. But this can't be used for e-commerce (for new products, products without a wiki page).
- Different phrases in a query are permuted more often in an e-commerce query vs web query.



Statistical Segmentation using Category-wise n-Gram Frequencies

- Segmentation model:

$$score(S) = \begin{cases} \sum_{s \in S, |s| \geq 2} |s|^{|s|} \cdot freq(s) & \text{if } freq(s) > 0 \text{ for all } s \in S, |s| \geq 2 \\ -1 & \text{else} \end{cases}$$

- Each product belongs to a category. Let S be set of results for q. $\alpha_s(q)$ is recall distribution of query. Let R be set of results in which user clicks. $\alpha_R(q)$ is the user intent distribution.
- Let q_s be one of the segmented forms of q. User intent score is $KL(\alpha_R(q), \alpha_s(q) - KL(\alpha_R(q_s), \alpha_s(q_s))$
- MI baseline computes MI at different segment boundary and based on that finds the best segmentation.

Comparing methods	Tie	1st Method Wins	2nd Method Wins
QSEGMENT Vs Frequency	12759	7828	4413
QSEGMENT Vs MWE	10766	9077	5157
QSEGMENT Vs MI	14553	6309	4138
QSEGMENT Vs No-segment	3628	14999	6373
Frequency Vs MWE	12796	6517	5687
Frequency Vs MI	11611	6079	7310
Frequency Vs No-Segment	10662	10506	3832
MWE Vs MI	12238	5342	7420
MWE Vs No-Segment	12249	9185	3566
MI Vs No-Segment	6386	13342	5272

Comparing methods	Query count	Length	Tie	1st Method Wins	2nd Method Wins
QSEGMENT	17120	3	7901	6158	3061
Vs	5769	4	2454	1998	1317
MWE	2110	>= 5	410	921	779
QSEGMENT	17120	3	11263	3489	2368
Vs	5769	4	2717	1887	1165
MI	2110	>=5	572	933	605

Table 5: Performance Comparison between different algorithms for various query lengths

Performing Segmentation jointly with Parts-of-speech Tagging and Capitalization

- They exploit the dependency between different unsupervised annotations to improve accuracy of the entire set of annotations.
  - E.g., Leverage information about estimated parts of speech tags and capitalization of query terms to improve accuracy of query segmentation.
- Start with an initial set of independently computed annotations $Z_Q^{*(I)}$
- Given $Z_Q^{*(I)}$, we are interested in obtaining an annotation set $Z_Q^{*(J)}$ which jointly optimizes prob of all annotations $Z_Q^{*(J)} = argmax_{\mathrm{Z_Q}}\ \mathrm{p}\left(\mathrm{Z_Q}\middle|Z_Q^{*(I)}\right)$

$$\mathcal{Z}_Q = \{\mathbf{CAP}, \mathbf{TAG}, \mathbf{SEG}\}$$

(a)

Term	CAP	TAG	SEG
who	L	X	B
won	L	V	I
the	L	X	B
2004	L	X	B
kentucky	C	N	B
derby	C	N	I

(b)

Term	CAP	TAG	SEG
kindred	C	N	B
where	C	X	B
would	C	X	I
i	C	X	I
be	C	V	I

(c)

Term	CAP	TAG	SEG
shih	C	N	B
tzu	C	N	I
health	L	N	B
problems	L	N	I

Figure 1: Examples of a mark-up scheme for annotating capitalization (L – lowercase, C – otherwise), POS tags (N – noun, V – verb, X – otherwise) and segmentation (B/I – beginning of/inside the chunk).

Performing Segmentation jointly with Parts-of-speech Tagging and Capitalization

- This formulation of joint query annotation is basically a stacked classification in which a second, more effective, classifier is trained using the labels inferred by the first classifier as features.
- 2 methods for Independent query annotations
  - Query-based estimation
    - Use large n-gram corpus to estimate $p(\zeta_i|q_i)$ for annotating query with capitalization and segmentation markup, and a standard POS tagger for POS tagging
  - PRF based estimation
    - z_Q can be computed based on all retrievable sentences r from collection C. This is approximated by considering top R sentences matching Q.
    - $p(\zeta_i|q_i)$ is a smoothed estimator that combines the information from the retrieved sentence r with the information about unigrams (for capitalization and POS tagging) and bigrams (for segmentation) from a large n-gram corpus

Input:	$\mathbf{Q}_t$ — training set of queries. $\mathcal{Z}_{\mathbf{Q}_t}$ — ground truth annotations for the training set of queries. $\mathbf{Q}_h$ — held-out set of queries.
(1)	Obtain a set of independent annotation estimates $\mathcal{Z}^{*(I)}_{\mathbf{Q}_t}$
(2)	Initialize $\mathcal{Z}^{*(J)}_{\mathbf{Q}_h} \leftarrow \emptyset$
(3)	*for each* $\mathbf{z}^{*(I)}_{\mathbf{Q}_t} \in \mathcal{Z}^{*(I)}_{\mathbf{Q}_t}$:
(4)	$\mathcal{Z}'_{\mathbf{Q}_t} \leftarrow \mathcal{Z}^{*(I)}_{\mathbf{Q}_t} \setminus \mathbf{z}^{*(I)}_{\mathbf{Q}_t}$
(5)	Train a CRF model $\mathcal{CRF}(\mathbf{z}_{\mathbf{Q}_t})$ using $\mathbf{z}_{\mathbf{Q}_t}$ as a *label* and $\mathcal{Z}'_{\mathbf{Q}_t}$ as *features*.
(6)	Predict annotation $\mathbf{z}^{*(J)}_{\mathbf{Q}_h}$, using $\mathcal{CRF}(\mathbf{z}_{\mathbf{Q}_t})$.
(7)	$\mathcal{Z}^{*(J)}_{\mathbf{Q}_h} \leftarrow \mathcal{Z}^{*(J)}_{\mathbf{Q}_h} \cup \mathbf{z}^{*(J)}_{\mathbf{Q}_h}$.
Output:	$\mathcal{Z}^{*(J)}_{\mathbf{Q}_h}$ — predicted annotations for the held-out set of queries.

$$\mathbf{z}_Q^{*(QRY)} = \underset{(\zeta_1,\ldots,\zeta_n)}{\operatorname{argmax}} \prod_{i\in(1,\ldots,n)} p(\zeta_i|q_i).$$

$$p(\mathbf{z}_Q|Q) = \sum_{r\in C} p(\mathbf{z}_Q|r)p(r|Q).$$

$$p(\mathbf{z}_Q|Q) \approx \sum_{r\in R} p(\mathbf{z}_Q|r)p(r|Q).$$

$$\mathbf{z}_Q^{*(PRF)} = \underset{(\zeta_1,\ldots,\zeta_n)}{\operatorname{argmax}} \sum_{r\in R} \prod_{i\in(1,\ldots,n)} p(\zeta_i|r)p(r|Q)$$

Performing Segmentation jointly with Parts-of-speech Tagging and Capitalization

- I=independent, j=joint, mean of classification accuracies per query (MQA)
- SEG-1 is [HPSB10]

$$S_Q^* = \underset{S \in \mathcal{S}_Q}{\operatorname{argmax}} \sum_{s \in S, |s|>1} |s|^{|s|} count(s)$$

- SEG-2 is [BW07]
  - Superviseed segmentation using a large number of features.

CAP	F1 *(% impr)*	MQA *(% impr)*
i-QRY	0.641 *(-/-)*	0.779 *(-/-)*
i-PRF	0.711*(+10.9/-)	0.811*(+4.1/-)
j-QRY	0.620†(-3.3/-12.8)	0.805*(+3.3/-0.7)
j-PRF	**0.718***(+12.0/+0.9)	**0.840***†(+7.8/+3.6)
TAG	Acc. *(% impr)*	MQA *(% impr)*
i-QRY	0.893 *(-/-)*	0.878 *(-/-)*
i-PRF	0.916*(+2.6/-)	0.914*(+4.1/-)
j-QRY	0.913*(+2.2/-0.3)	0.912*(+3.9/-0.2)
j-PRF	**0.924***(+3.5/+0.9)	**0.922***(+5.0/+0.9)
SEG	F1 *(% impr)*	MQA *(% impr)*
i-QRY	0.694 *(-/-)*	0.672 *(-/-)*
i-PRF	0.753*(+8.5/-)	0.710*(+5.7/-)
j-QRY	0.817*†(+17.7/+8.5)	**0.803***†(+19.5/+13.1)
j-PRF	**0.819***†(+18.0/+8.8)	**0.803***†(+19.5/+13.1)

SEG	F1	MQA
SEG-1	0.768	0.754
SEG-2	**0.824***	0.787*
j-PRF	0.819* *(+6.7%/-0.6%)*	**0.803*** *(+6.5%/+2.1%)*

Supervised Segmentation using Decision-boundary, Context and Dependency Features

- Previous unsupervised approaches
  - Combine frequency count of a segment and mutual information between pairs of words in the segment in a heuristic scoring function [RMB03]
    - Drawback: Heavy dependence on corpus statistics.
  - Use MI between pairs of tokens as the sole factor in deciding on segmentation breaks. If the MI is above a threshold (optimized on a small training set), the pair of tokens is joined in a segment. Otherwise, a segmentation break is made.[JRMG06]
    - Drawback: Cannot model dependencies beyond the pair of words
- Supervised approach
  - SVM is trained using features generated from a window of 6 tokens in the query.
    $\{\dots, x_{L_2}, x_{L_1}, x_{L_0}, x, x_{R_0}, x_{R_1}, x_{R_2}, \dots\}$

1. [two man power saw]
2. [two man] [power saw]
3. [two] [man] [power saw]
4. [two] [man power] [saw], etc.

Supervised Segmentation using Decision-boundary, Context and Dependency Features

Table 1: Indicator features.

Name	Description
is-the	token x = "the"
is-free	token x = "free"
POS-tags	Part-of-speech tags of pair x_{L0} x_{R0}
fwd-pos	position from beginning, i
rev-pos	position from end $N - i$

Table 2: Statistical features.

Name	Description
web-count	count of "x" on the web
pair-count	web count "w x"
definite	web count "the w x"
collapsed	web count "wx" (one word)
and-count	web count "w and x"
genitive	web count "w's x"
Qcount-1	Counts of "x" in query database
Qcounts-2	Counts of "w x" in database

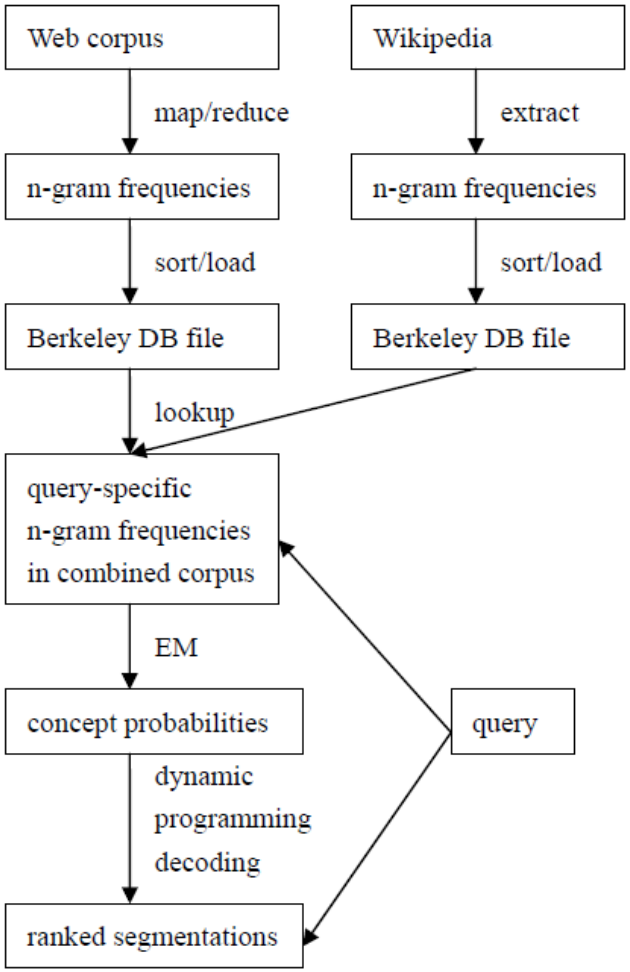
Feature Type	Feature Span	Test Set	
		Seg-Acc	Qry-Acc
MI	Decision-Boundary	68.0	26.6
Basic	Decision-Boundary	71.7	29.2
Basic	Decision-Boundary, Context	80.2	52.0*
Basic	Decision-Boundary, Context, Dependency	81.1	53.2
All	Decision-Boundary	84.3	57.8*
All	Decision-Boundary, Context	86.3	63.8*
All	Decision-Boundary, Context, Dependency	85.8	61.0

- Decision boundary features (as described in tables above)
- Context features
  - Features to indicate what parts of the size-6 window are available.
  - Token-level, pairwise, trigram and fourgram counts for all subsequences in the size-6 window
- Dependency features
  - Pairwise counts between x_{L_0} and x_{R_1}; and between x_{L_1} and x_{R_0}
- Results
  - Basic Decision boundary features include the counts used in computing MI
  - MI approach is the [JRMG06] approach.
  - Seg-Acc is Segmentation decision accuracy; Qry-Acc is Query Segmentation accuracy

Unsupervised Segmentation Using Generative Language Models and Wikipedia

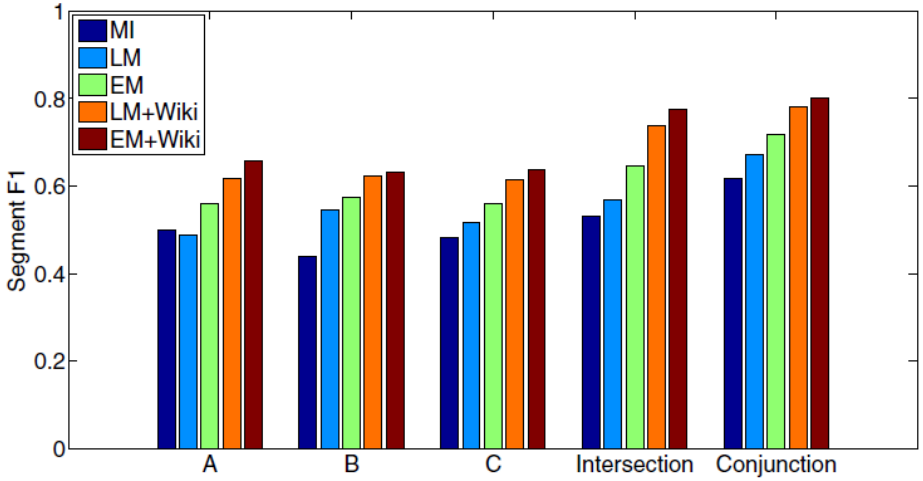
- Drawback of supervised algorithms: Need a large amount of training data and well-designed domain dependent features.
- They use a generative query model to recover a query's underlying concepts that compose its original segmented form.
  - Queries are made up of unigrams (1 gram=concept)
  - How to compute parameters of the unigram model (i.e., prob of these concepts)?
    - Given a huge web crawl, compute frequencies of all possible n-grams up to a certain length (n=1,2,…,5) that occur at least once in the corpus. $P_C(x) = \frac{\#x}{\sum_{x' \in V} \#x'}$
      - Prob of a segmentation is $P(S^Q) = \prod_{s_i \in S^Q} P_C(s_i)$
      - Prob of a query is $P(Q) = \sum_{S^Q} P(S^Q)$ where S^Q is one of the 2^{n-1} different segmentations where n=#query words
      - All possible segmentations can be evaluated using a dynamic programming algorithm in O(nkm log(km)) time where m is max allowed segment length.
    - 2 problems: 1. How to compute V? 2. n-gram is to appear in a piece of text *as an independent concept*. But P_C(York times) will be $\geq P_C$(new York times)

Unsupervised Segmentation Using Generative Language Models and Wikipedia



- The model's parameters are estimated using an expectation-maximization (EM) algorithm, optimizing the minimum description length objective function on a partial corpus that is specific to the query.
  - In E step, the corpus is segmented using current set of estimated parameter values.
  - In M step, new set of parameter values (concept prob) are calculated to maximize the complete likelihood of the data which is augmented with segmentation information.
  - Efficient EM: When a new query arrives, we extract parts of the corpus that overlap with it (query-relevant partial corpus), which are then segmented into concepts, so that probabilities for n-grams in the query can be computed.
    - This is a list of n-grams from the query, associated with their longest match count $D = \{(x, c(x)) | x \in Q\}$
  - MAP estimate of concept prob is then $\theta = argmax\ P(D|\theta)P(\theta) = argmin(-\log P(D|\theta) - \log P(\theta))$
    - $-\log\ P(D|\theta)$ is description length of corpus and $-\log P(\theta)$ is description length of parameters
    - Details of computing these factors are in paper.
- To augment this unsupervised learning, they incorporate evidence from Wikipedia.

Unsupervised Segmentation Using Generative Language Models and Wikipedia



- Segment F1 of different query segmentation algorithms.
- 3 annotators labeled query segments A, B, C
- Intersection is the set of queries when all 3 agreed.
- Conjunction: Segmentation is considered correct if it matches what is given by any of the 3 annotators.

Tutorial Overview

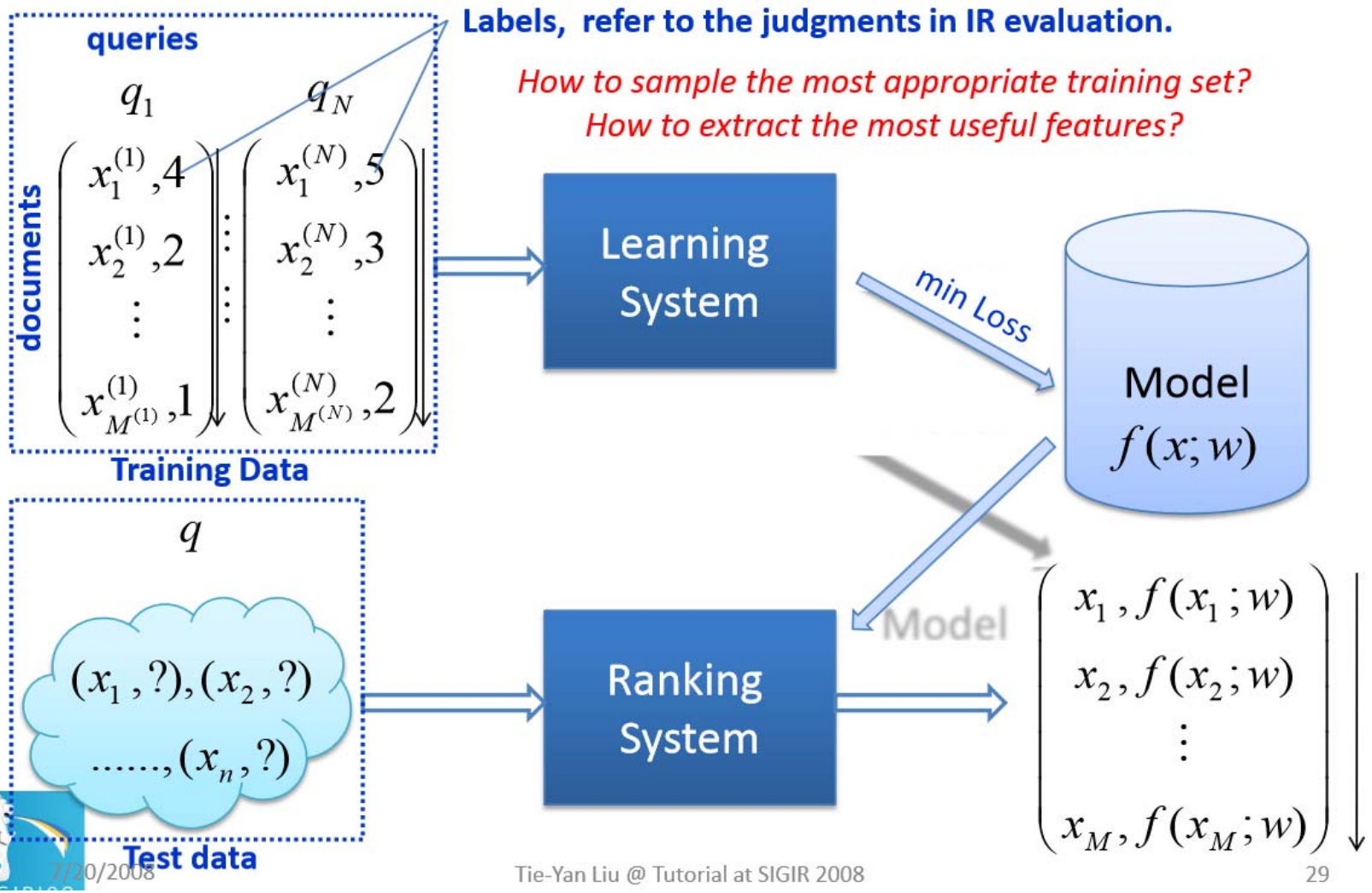
- Properties of Verbose Queries [30 min]
- Techniques for Processing Verbose Web Queries
  - Query Reduction to a Single Sub-Query [50 min]
  - Query Reduction by Choosing Multiple Sub-Queries [10 min]
  - Break [30 min]
  - Weighting Query Words [90 min]
  - Lunch Break [90 min]
  - Query Expansion by Including Related Concepts [20 min]
  - Query Reformulation [30 min]
  - Query Segmentation [40 min]
  - **Break [30 min]**
  - Query-Dependent Learning to Rank [40 min]
- Applications of Verbose Query Processing [40 min]
- Summary and Future Research Directions [10 min]

Tutorial Overview

- Properties of Verbose Queries [30 min]
- Techniques for Processing Verbose Web Queries
  - Query Reduction to a Single Sub-Query [50 min]
  - Query Reduction by Choosing Multiple Sub-Queries [10 min]
  - Break [30 min]
  - Weighting Query Words [90 min]
  - Lunch Break [90 min]
  - Query Expansion by Including Related Concepts [20 min]
  - Query Reformulation [30 min]
  - Query Segmentation [40 min]
  - Break [30 min]
  - **Query-Dependent Learning to Rank [40 min]**
- Applications of Verbose Query Processing [40 min]
- Summary and Future Research Directions [10 min]

Learning to Rank – Crash Course

Framework of Learning to Rank



Learning to Rank – Crash Course

- Conventional machine learning
  - Classification / Regression.
- Beyond conventional machine learning
  - Ordinal regression (*a pointwise approach*).
  - Preference learning (*a pairwise approach*).
  - Listwise ranking (*a listwise approach*).

Learning to Rank – Crash Course

- Features extracted for each query-document pair
  - **Query-dependent** – query class, length, terms
  - **Document-dependent** – link and content based priors
  - **Query & document dependent** – match scores

Learning to Rank – Crash Course

- Standard approach for generating training data for learning to rank
  - Run a bag-of-words retrieval algorithm, e.g. BM25 for a set of queries **Q**
  - Retain Top-K results (10 < K < 1000) for each query $\mathbf{q_i}$
  - For each result $\mathbf{x_{ij}}$ extract a feature vector
  - Label each result $\mathbf{x_{ij}}$

Learning to Rank – Crash Course

- Issues that arise for learning to rank for verbose queries
  - Training set may be of lower quality – less relevant documents are retrieved
  - Initial relevance scores are less reliable
  - Learning the same model across all the queries may be detrimental (especially for linear models)
  - Potential query transformations: weighting, reduction, expansion, segmentation, are not accounted for

Learning to Rank – Crash Course

- In this section of the tutorial, we will present some work that starts to tackle these issues
  - Better training set generation
  - Query – dependent learning to rank
  - Learning better query representations

- Also worth looking at "Machine Learning for Query-Document Matching in Web Search" tutorial by Li & Xu, WSDM2012

Learning to Rank – Crash Course

- Learning to rank for verbose natural language is still a relatively unexplored field
  - Current approaches are still too simplistic
  - Plenty of room for innovative research
  - Many practical applications
    - Question answering systems
    - Dialogue systems
    - Digital assistants

Two-Stage Learning to Rank for Information Retrieval

- Standard learning to rank pipeline
  - **Retrieval (Stage A)** Run a bag-of-words model (e.g. BM25) over the entire index
  - **Re-ranking (Stage B)** Re-score top-K retrieved results by a learning to rank model with arbitrary number of features
  - Works reasonably well for short queries
  - However, for verbose queries, as we have seen before, bag-of-words models may not be sufficient

Two-Stage Learning to Rank for Information Retrieval

- If the relevant document recall at the initial retrieval stage is low, re-ranking stage will suffer.

	Query "lower heart rate"	
	BM25	*MSE*
Relevant retrieved (*Stage A*)	30	73
NDCG@20 (*Stage B*)	19.88	58.88

Two-Stage Learning to Rank for Information Retrieval

- Instead of the standard bag-of-words retrieval followed by a learning stage, learn a two-stage ranking model
- Formally

Ranker A:
Defined over the entire corpus C

$$M_A : T \times \mathcal{C} \rightarrow \mathbb{R}$$

$$M_A^* = \arg\max_{M_A} \sum_{q \in T} \Lambda(M_A).$$

Ranker B:
Defined over the output of Ranker A

$$M_B : (q_i \in T) \times (d_j \in D_{M_A^*}^{(i)}) \rightarrow \mathbb{R}$$

$$M_B^* = \arg\max_{M_B} \sum_{q \in T} \Lambda(M_B, M_A^*)$$

Two-Stage Learning to Rank for Information Retrieval

Ranker A	Ranker B
BM25	RankBoost
Weighted Sequential Dependence (WSD)	Coordinate Ascent
Multiple Source Expansion (MSE)	LambdaMart

- Clearly, any pairwise combination of these (and other) rankers is possible.
- Main trade-offs are in effectiveness / efficiency

Two-Stage Learning to Rank for Information Retrieval

Ranker A	Ranker B
BM25	RankBoost
Weighted Sequential Dependence (WSD)	Coordinate Ascent
Multiple Source Expansion (MSE)	LambdaMart

- Multi-tiered models are also possible
  - Ranker A: BM25 over the entire corpus
  - Ranker B: WSD over Top-10K results
  - Ranker C: MSE over Top-5K results
  - Ranker D: LambdaMart over Top-1K results

Two-Stage Learning to Rank for Information Retrieval

Gov2				
	⟨*title*⟩		⟨*description*⟩	
	NDCG@20	MAP	NDCG@20	MAP
CA[BM25]	47.80	30.09	40.69	26.40
CA[WSD]	48.24 *(+0.92%)*	34.26* *(+13.86%)*	43.87* *(+7.82%)*	29.93* *(+13.37%)*
CA[MSE]	50.19* *(+5.0%)*	36.12* *(+20.0%)*	45.27* *(+11.3%)*	32.41* *(+22.7%)*

ClueWeb-B				
	⟨*title*⟩		⟨*description*⟩	
	NDCG@20	MAP	NDCG@20	MAP
CA[BM25]	28.36	24.14	22.07	15.32
CA[WSD]	30.93* *(+9.06%)*	25.58 *(+5.97%)*	22.17 *(+0.45%)*	15.90 *(+3.79%)*
CA[MSE]	32.20* *(+13.5%)*	27.19* *(+12.6%)*	24.65* *(+11.7%)*	16.74* *(+9.2%)*

Query Dependent Ranking Using K-Nearest Neighbor

- Proposes k-Nearest Neighbors algorithm for query-dependent learning to rank
- Explores two families of methods
  - Online – find nearest queries at rank time
  - Offline – pre-computed approximation of the online method.

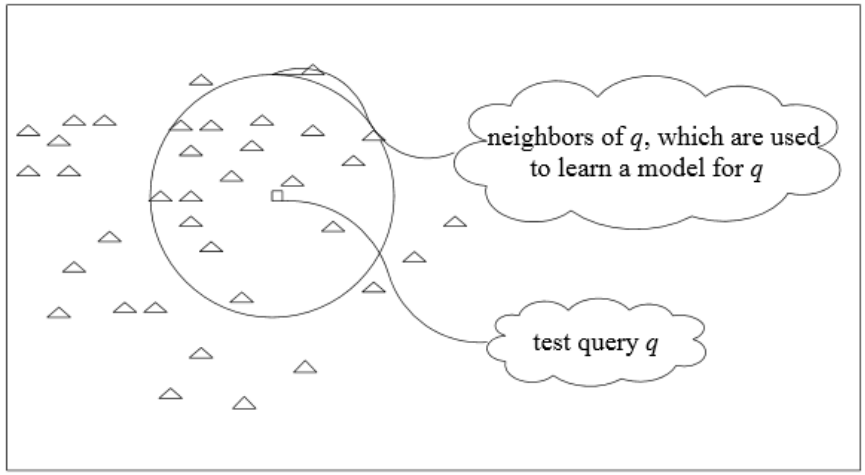
Query Dependent Ranking Using K-Nearest Neighbor

- Hard classification of search queries is not feasible
  - Queries often mix categories and tasks
  - Empirically, it is impossible to draw hard boundaries between query representations in feature space
  - Infinitely many unique query variations

- Instead use soft-classification rules

Query Dependent Ranking Using K-Nearest Neighbor

- Online model



Clearly, infeasible in low-latency web search

Algorithm: KNN Online

Input:

(1) A test query q and the associated documents to be ranked.

(2) Training data $\{S_{q_i}, i = 1, ..., m\}$.

(3) Reference model h_r (currently BM25).

(4) Number of nearest neighbors k.

Output: Ranked list for query q.

Algorithm:

Offline pre-processing:
For each training query q_i, use reference model h_r to find its top T ranked documents, and compute its query features from these documents.

Online training and testing:

(a) Use reference model h_r to find the top T ranked documents for query q, and compute q's query dependent features from these documents.

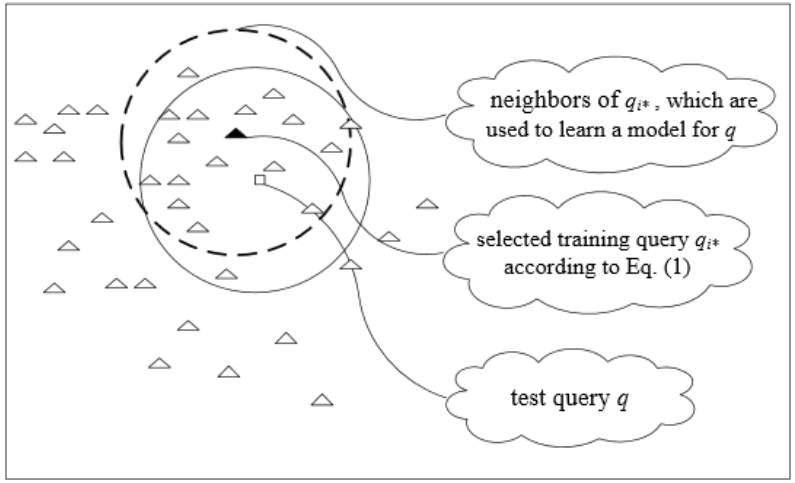
(b) Within the training data find k nearest neighbors of q, denoted as $N_k(q)$, with distance computed in the query feature space.

(c) Use the training set $S_{N_k(q)} \triangleq \cup_{q' \in N_k(q)} S_{q'}$ to learn a local model h_q.

(d) Apply h_q to the documents associated with query q, and obtain the ranked list.

Query Dependent Ranking Using K-Nearest Neighbor

- Offline model



Algorithm: KNN Offline-1

Input:

(1) A test query q and the associated documents to be ranked.

(2) Training data $\{S_{q_i}, i = 1, ..., m\}$.

(3) Reference model h_r (currently BM25).

(4) Number of nearest neighbors k.

Output: Ranked list for query q.

Algorithm:

Offline training:

(1) For each training query q_i, use reference model h_r to retrieve its top T documents, and compute its query features.

(2) For each training query q_i, find k nearest neighbors of q_i, denoted as $N_k(q_i)$, in the training data in the query feature space, and use training set $S_{N_k(q_i)}$ to learn a local model h_{q_i}.

Online testing:

(a) Use reference model h_r to find top T documents for query q, and compute its query features.

(b) Find k nearest neighbors of q, denoted as $N_k(q)$ in the training data in the query feature space.

(c) Find the most similar training set $S_{N_k(q_{i*})}$ by using Eq.(1).

(d) Apply $h_{q_{i*}}$ to the documents associated with query q, and obtain the ranked list.

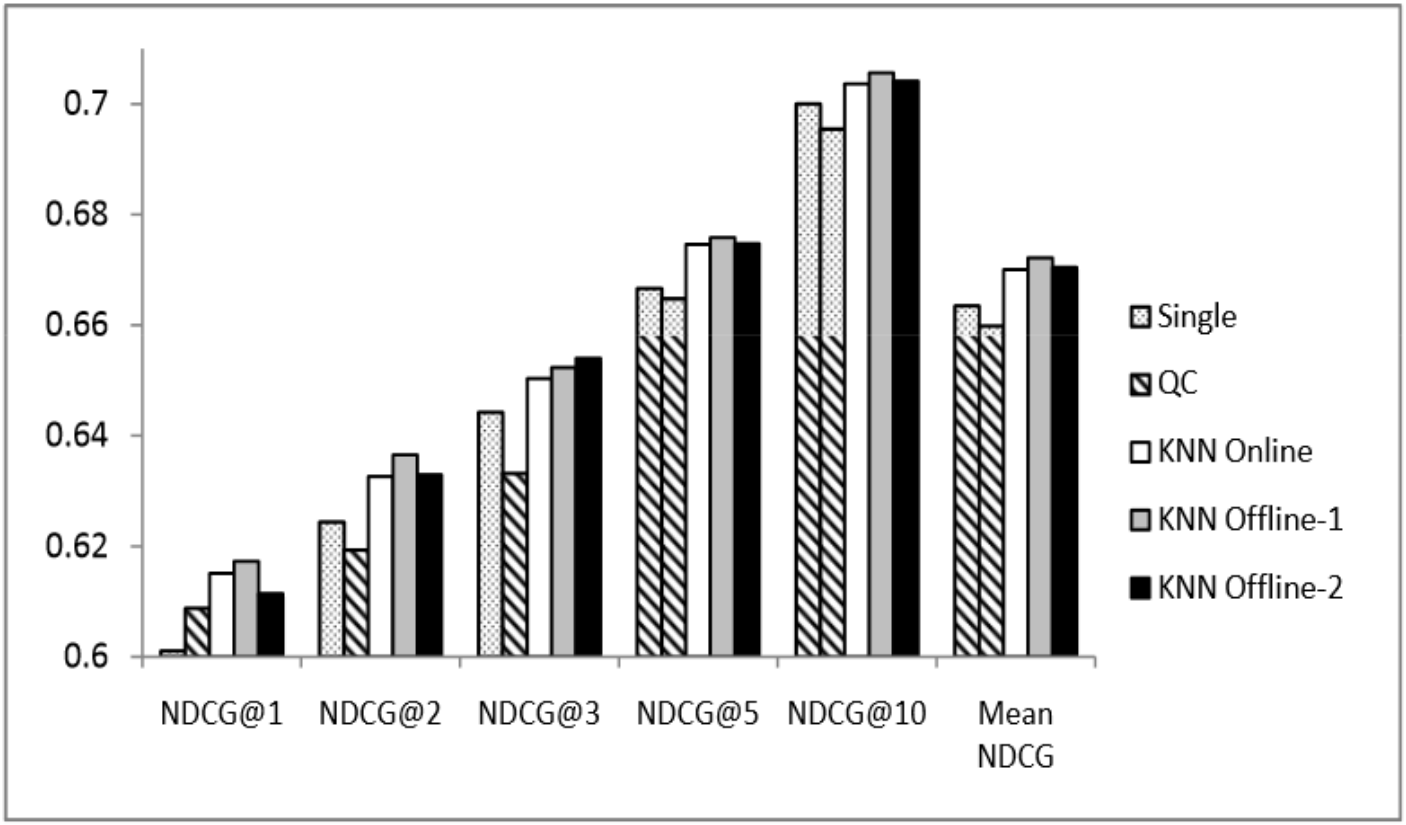
$$S_{N_k(q_{i*})} = \arg\max_{S_{N_k(q_i)}} |S_{N_k(q_i)} \cap S_{N_k(q)}| \qquad (1)$$

Query Dependent Ranking Using K-Nearest Neighbor

- The authors also describe a faster offline algorithm **KNN Offline-2**
  - directly finds the most similar query in the training data
  - avoids the cost of k-nearest neighbors

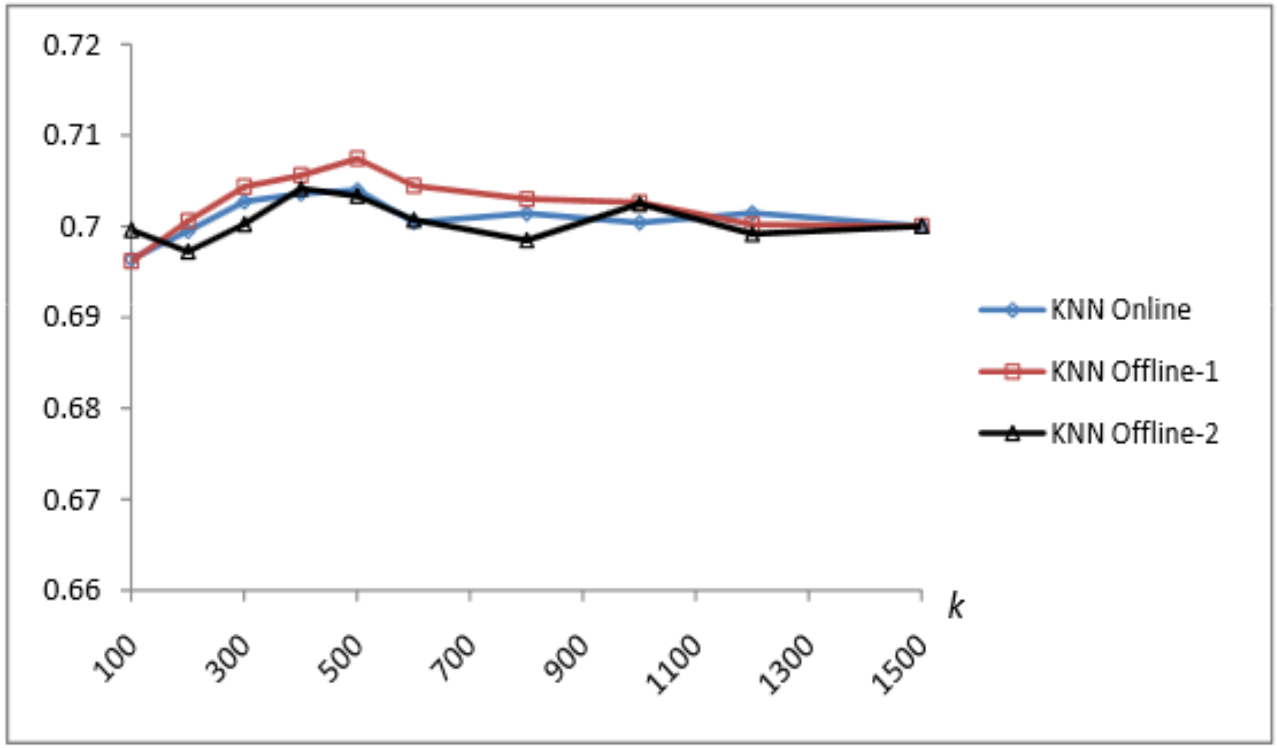
Query Dependent Ranking Using K-Nearest Neighbor

- Comparison with single and query classification (QC) models



Query Dependent Ranking Using K-Nearest Neighbor

- Comparison of kNN variants



(b) Ranking Accuracies at NDCG@10

Selecting Good Expansion Terms for Pseudo-Relevance Feedback

- Classification of Expansion Terms
  - Attempt to learn whether a given expansion term is useful

$$chg(e) = \frac{MAP(q \cup e) - MAP(q)}{MAP(q)}$$

  - For useful terms: **chg(e) > 0.005**
  - Use a classification approach with features based on the pseudo – relevance feedback

Selecting Good Expansion Terms for Pseudo-Relevance Feedback

- Features
  - Term distribution in the PR set
  - Co-occurrence with other query terms
  - Weighted proximity to other query terms
  - Document frequency for query terms and expansion terms

Selecting Good Expansion Terms for Pseudo-Relevance Feedback

- Results – Oracle

Models	AP	WSJ	Disk4&5
LM	0.2407	0.2644	0.1753
REL	0.2752^{L}	0.2843^{L}	0.1860^{L}
REL+Oracle	$\mathbf{0.3402}^{R,L}$	$\mathbf{0.3518}^{R,L}$	$\mathbf{0.2434}^{R,L}$
MIX	0.2846^{L}	0.2938^{L}	0.2005^{L}
MIX+Oracle	$\mathbf{0.3390}^{M,L}$	$\mathbf{0.3490}^{M,L}$	$\mathbf{0.2418}^{M,L}$

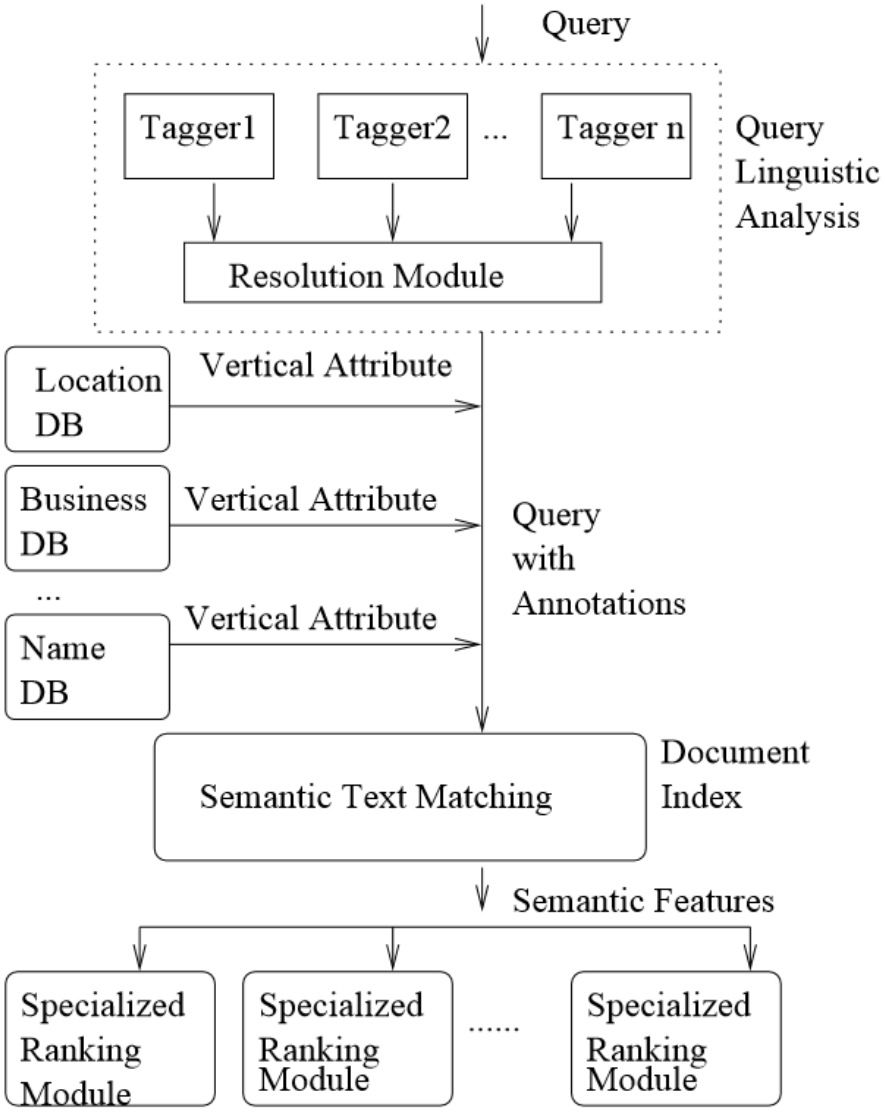
Table 2. The impact of oracle expansion classifier

- Actual improvements are much smaller, but still substantial.

Improving Web Search Relevance with Semantic Features

- How to deal with semantic feature sparsity across queries
  - Almost never feasible to learn per-term / concept weights
  - One approach is concept parameterization (presented before)
  - Another approach is identifying semantic classes to which query belongs and using semantic classes as query proxies.

Improving Web Search Relevance with Semantic Features



Improving Web Search Relevance with Semantic Features

$$San\ Francisco\ colleges$$

$$[San\ Francisco]_{CityName}\ [colleges]_{BusinessCategory}$$

- Add semantic features into a learning to rank model with the following weighted loss function

$$L_C^w(h) \equiv w \sum_{i \in C} (y_i - h(x_i))^2 + \sum_{i \in \bar{C}} (y_i - h(x_i))^2$$

Class of semantically rich long queries

Improving Web Search Relevance with Semantic Features

- Results

Weight	w/o semantic features		w/ semantic features	
	DCG(5)	Impr.	DCG(5)	Impr.
$w = 0$	8.09^b	-	8.25	**2.0%**
$w = 2$	8.09	0.02%	8.26	**2.1%**
$w = 4$	8.13	0.49%	8.34	**3.1%**
$w = 8$	8.13	0.49%	8.42	**4.1%**
$w = 16$	8.13	0.49%	8.30	**2.6%**
$w = 32$	8.04	−0.60%	8.27	**2.2%**

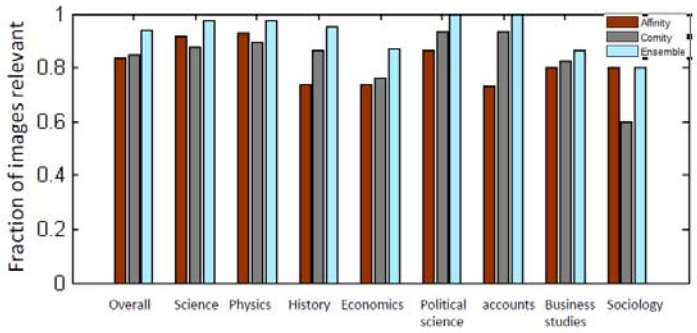
Tutorial Overview

- Properties of Verbose Queries [30 min]
- Techniques for Processing Verbose Web Queries
  - Query Reduction to a Single Sub-Query [50 min]
  - Query Reduction by Choosing Multiple Sub-Queries [10 min]
  - Break [30 min]
  - Weighting Query Words [90 min]
  - Lunch Break [90 min]
  - Query Expansion by Including Related Concepts [20 min]
  - Query Reformulation [30 min]
  - Query Segmentation [40 min]
  - Break [30 min]
  - Query-Dependent Learning to Rank [40 min]
- **Applications of Verbose Query Processing [40 min]**
- Summary and Future Research Directions [10 min]

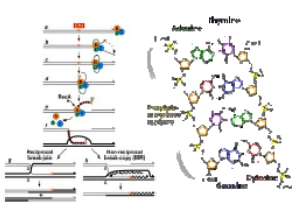
Finding Images for Books

- Given section text, get a ranked list of top k images. 2 algorithms: Affinity and Comity
- Affinity
  - Obtain key concept phrases from section text (linguistic pattern A^*N^+ where A=adjective, N=noun)
  - Get images from articles with high document similarity to section
  - Relevance score for an image is computed by analyzing the overlap between the concept phrases and the cumulative metadata associated with the various copies of the image present in the narrowed set of articles.
- Comity
  - All combinations of 2 and 3 words fired as queries to e search engines.

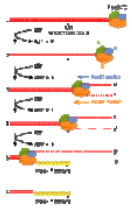
1: Obtain (up to) top c concept phrases from section j.
2: Form queries consisting of two and three concepts phrases each ($\binom{c}{2} + \binom{c}{3}$ queries in total).
3: Obtain (up to) top t image search results for each of the queries from e different search engines.
4: Aggregate over (potentially $e(\binom{c}{2} + \binom{c}{3})$) lists of images, to obtain λ_{ij} values for each image.
5: Return top k images along with their λ_{ij} values.

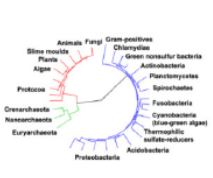


Grade X Science 8: Do Organisms Create Exact Copies of Themselves?


Molecular model for the RecBCD pathway of recombination.

The chemical structure of DNA


Beginning of the RecBCD pathway


Phylogenetic tree


DNA, molecular basis for inheritance

Grade XII History 7: Rayas, Nayaks and Sultans


Poetic inscription by Vijayanagara poet Manjaraja (1398 CE)


Bijapur Sultanate territories under Ibrahim II in 1620 CE


Lord Rama breaking Shiva's bow in Hazare Rama Temple at Hampi.


Rashtrakuta Empire in 800 CE, 915 CE.

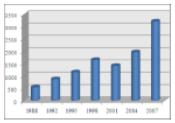

Western Chalukya Empire in 1121 CE

Grade XII Economics 6: Foreign Exchange Market


A gold-standard 1928 one-dollar bill.


Reserves of SDR, forex and gold in 2006


Gold standard widely adopted


Yearly Forex turn over


Exchange rate display

Question Answering

- [HC10] uses stop structure identification and stopword removal for reducing the verbose query to a single sub-query.
  - "please tell me why the sky is blue" → "sky blue"
  - "for a year ive been getting some tightening from within my chest" → "tightening chest"
- [XJC08] use translation-based language models for query reformulation.
  - Top 5 retrieved questions:
    - Query: who is the leader of india
      - who is the prime minister of india
      - who is current vice prime minister of india
      - who is the army chief of india
      - who is the finance minister of india
      - who is the first prime minister of india
    - Query: who made the first airplane that could fly
      - what is the oldest aiirline that still fly airplane
      - who was the first one who fly with plane
      - who was the first person to fly a plane
      - who the first one fly to the spase
      - who the first one who fly to sky

Querying a Database of Meeting Dialogues

- Given a database of meeting scripts (speech to text processed) with semantic labels.
- Queries
  - Let me see the moment when _XX_ and _YY_ strongly disagree.
  - How did we arrive at the conclusion that _XX_?
  - What were the objection to the proposal _ZZ_?
  - What was the position of _XX_ on subject _ZZ_?
  - What open questions remained?
  - Which criteria were adopted to take the decision _D1_?
  - Which arguments did the members who disagreed on the decision _D1_invoke?
  - Give me for each topic the list of people that said something related to it. Then give me their main point (agreement / disagreement with _XX_).
  - For each topic, give me the conclusion or the decision that was made.

Type of Dialogue	Initial Situation	ParticipantGoal	Goal of Dialogue
Persuasion	Conflict of opinions	Persuade other party	Resolve or clarify issue
Inquiry	Need to have proof	Find and verify evidence	Prove (disprove) hypothesis
Negotiation	Conflict of interests	Get what you most want	Reasonable settlement
Information seeking	One party lacks information	Acquire or give information	Exchange information
Deliberation	Dilemma or practical choice	Co-ordinate goals or action	Decide best course of action
Eristic	Personal conflict	Verbally hit out at opponent	Reveal deeper basis of conflict

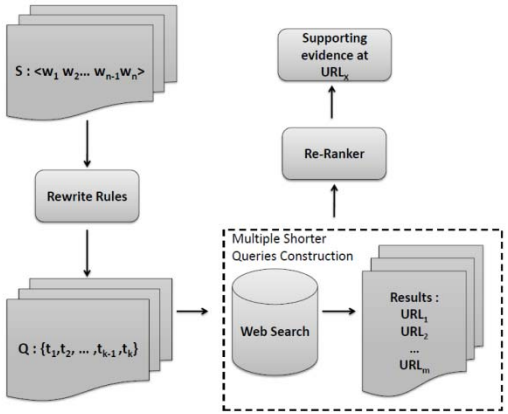
Classification of dialogue situations

Searching for Cancer Information

- Analyzed 3 months of cancer queries on Ask.com
- Examples
  - "Where can I find information about breast cancer?"
  - "Where can I find a Web site with information on using high protein food to fight Breast cancer?"
  - "How do antioxidants prevent cancer that may be caused by free radicals?"
  - "where can I have an ultra fast CT scan done"
  - "When do people who have pancreatic cancer know that they have the disease?"
- Users ask questions using whole sentences and keywords, often misspelling words.
- Types of queries
  - Cancer (N = 59619, 78.37%)
  - General Research (N = 7808, 10.26%)
  - Treatment (N = 3832, 5.04%)
  - Diagnosis and Testing (N = 3315, 4.36%)
  - Cause/Risk/Link (N = 1249, 1.64%)
  - Coping (N = 254, 0.33%)

Fact Verification

- "In 1981, Obama transferred to Columbia University in New York City, where he majored in political science with a specialty in international relations"
- Given a factual statement, the proposed system first transforms it into a set of semantic terms by using boosted decision trees with multiple syntactic, encyclopedic and heuristic features.
- It then employs a quasi-random strategy for selecting subsets of the semantic terms according to topical likelihood.
- These semantic terms are used to construct queries.
- Retrieved documents are aggregated and re-ranked by employing additional measures of their suitability to support the factual statement.
- Wikipedia has in-sentence and end-of-sentence citations. The sentences (or their parts) are facts and the references serve as fact verification documents.



Method	Top 1	Not Retrieved
Verbatim	1.36	95.67
Discard stopwords	3.48	90.08
Oracle: overlapping words	8.48	71.67

The percentage of Wikipedia references that are retrieved by Bing on the top position for the corresponding statement when using as query (a) the verbatim form of the statement; (b) the statement from which stopwords were removed; and (c) only the words that are shared between the statement and the referenced document (the latter is seen as an oracle). Not retrieved (within top 50 results).

Fact Verification

Features	Description
Offset(t, S)	the backward distance (measured in semantic terms) from the end of the statement (the citation marker in Wikipedia)
TermCount(S)	the number of unique semantic terms
Keyphraseness(t)	$\frac{N_{sd}(t)}{N_s(t)}$, where $N_{sd}(t)$ is the # of times t is an overlapping term, $N_s(t)$ is # of statements in which t occurs in training
WikitextFreq(t)	the term's frequency in the associated Wikipedia article (or, generally, the article where the term is found)
POS(t, S)	the part-of-speech tag (Penn Treebank) of the term in the statement
TFIDF(t, S)	The product of the term's frequency in factual statement and its IDF derived from two corpora, BNC and NYT
Multiword(t)	binary value indicating whether semantic term is a multi-word compound

- To drop terms and select only k
  - Pachinko Allocation model is used to compute similarity between the term and the fact in topic space.
  - Prob. of a term being dropped depends on this topic match score.
  - Multiple such size k queries are fired.
  - Results from multiple searches are combined and re-ranked using $Score(d) = \frac{1}{minrank(d)} \times Cosine(d, S) \times PageRank(d)$ where S is the fact.

Fact Verification

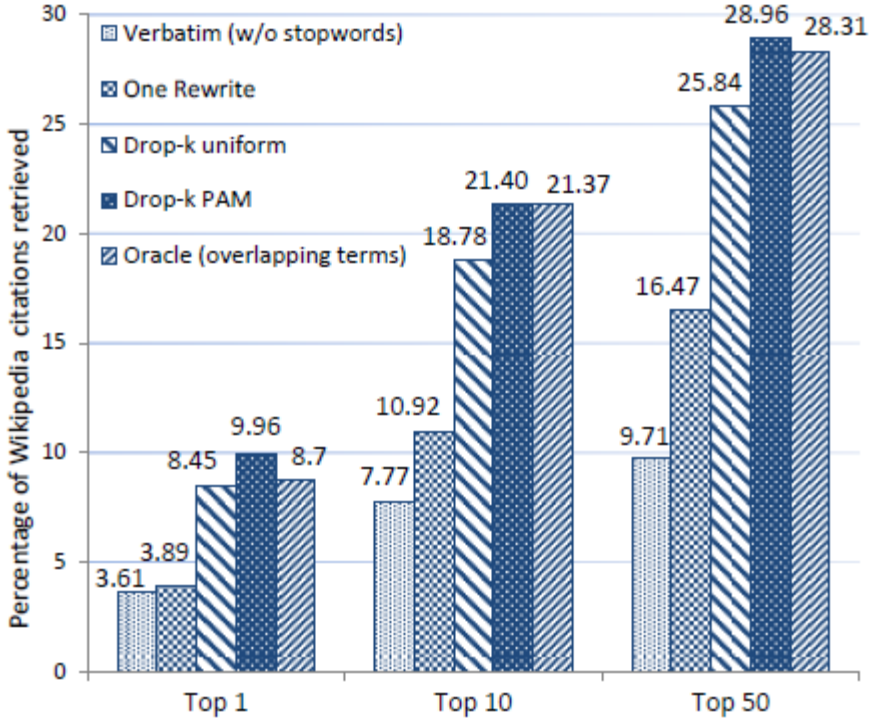


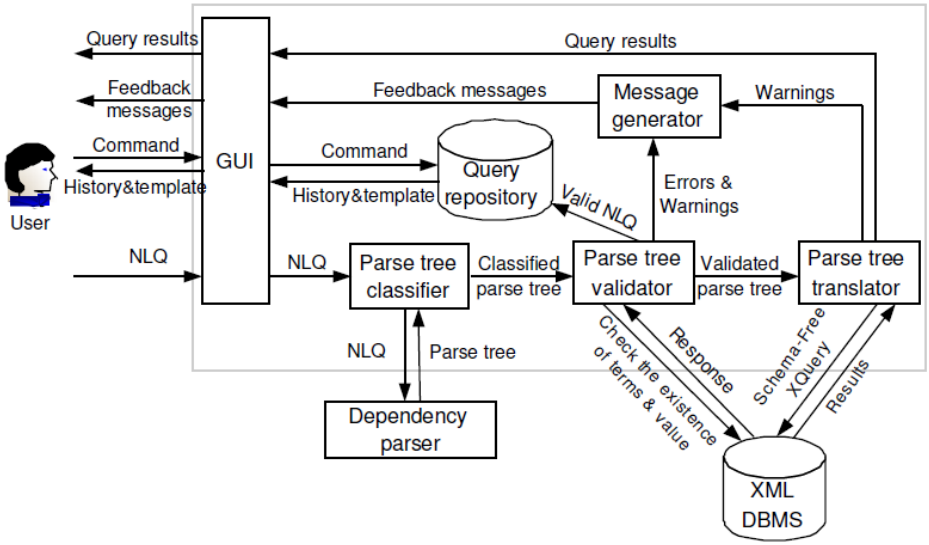
Figure 6: Bar chart summary for the percentage of Wikipedia citations retrieved in Top *n* results for all models proposed.

Reference Type	Quality	Ease of Effort	Trustworthiness
Wikipedia	83.0%	79.3%	90.0%
One Rewrite	71.0%	75.3%	87.0%
Drop-k PAM	71.0%	74.7%	88.7%

One Rewrite is the system with just query rewrite but no term dropping

Natural Language Interface for Databases

- NaLIX is a generic interactive natural language query interface to an XML database.
- It can accept an arbitrary English language sentence as query input, which can include aggregation, nesting, and value joins, among other things. This query is translated, potentially after reformulation, into an XQuery expression that can be evaluated against an XML database.
- The translation is done through mapping grammatical proximity of natural language parsed tokens to proximity of correspond- ing elements in the result XML.



- Query Translation
  - Parse tree classification
    - Identifies words/phrases in parse tree of original sentence that can be mapped into Xquery components (tokens) or not (markers)
  - Parse tree validation
    - Checks if the parse tree is mappable to an Xquery
    - Checks if element/attribute names or values contained in user query can be found in the database
  - Parse tree translation
    - Utilize structure of NL constructions as reflected in parse tree to generate appropriate structure in Xquery expression.
    - Notions of core token, token attachment and token relationship help perform semantic grouping and hence identification of query nesting and structural relationships between result elements when mapping tokens to Xquery.

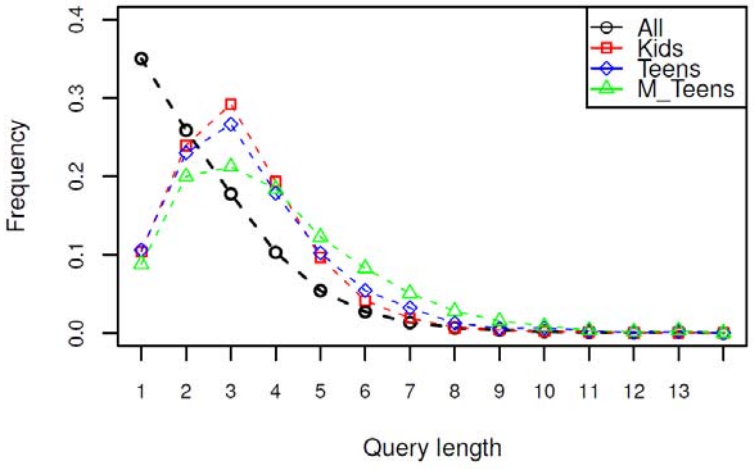
PSAH13, YPSS14

E-Commerce

- Consider a long query, like new ac adapter and battery charger for hp pavilion notebook; it often suffers from zero-recall on ebay, a large e-Commerce marketplace but breaking the query into meaningful phrases, such as {new | ac adapter | and | battery charger | for | hp pavilion notebook} helps in reformulating the query as new battery charger hp pavilion; this shortened query retrieves more than four thousand products. [PSAH13]
  - Qsegment uses frequency data from query log and product titles for domain-specific segmentation.
- Query reduction by single term deletion [YPSS14]
  - Accurate prediction of term deletion can potentially help users recover from poor search results and improve shopping experience.
    - ”small carry on bag for air plane” results in no search results while the query ”carry on bag” leads to many relevant items.
  - Learn domain-specific classifiers using multiple term-dependent and query-dependent features.

Search Queries from Children

- Query length is an indicator of the complexity of the query and the difficulty of the user to express information needs using keywords. The average query length found for the kids, teens and mature teens datasets were 3.8, 3.4 and 3.2, respectively; average being 2.5 words (AOL query log).
- These studies suggest that query reduction and query segmentation techniques can be particularly beneficial for children content queries since it has been shown that longer queries are less efficient.
- Similarly query reformulation techniques based on morphological and syntactical features of the queries can improve the search process by mapping phrases to concepts and keywords



	All	Kids	Teens	M_Teens
n.q	48.1%	25.9%	26.5%	24.5%
w.a	1.5%	1.37%	1.8%	1.8%
w.r	0.1%	0.21%	0.2%	0.2%
w.c	6.3%	7.8%	8%	8.5%
m.r	38.8%	59.6%	57.6%	59%
p.q	3%	3.3%	3.7%	3.5%
s.c	1.8%	1.5%	2.1%	2.1%

Nq=New query, wa=Words added, wr=Words removed, wc=Words changed, mr=More results, pq=Return to prev query, sc=Spell correction

Music Search

- People often first encounter new music from various media (e.g., a movie, TV commercial, radio) without ever knowing the artist name and song title, or they often forget the information over the course of time. This brings challenges to known-item searches for music as those users must attempt to describe the sought music other than using the bibliographic information. People who had no formal music education or people who seek music from different cultures or music in non-native languages can experience difficulties describing the music they seek.
- Types of needs: identify work/artist, get recommendations, "Give me some music similar to this particular song(s) or artist(s)", etc.
- Original query: I heard this song by a female singer in an ARBY's. I believe it is from the 70s or early 80s. The main chorus of the song says, "over and over again." Kind of a sad, slow, easy listening love song.

Group	Description
Identification	Questions asking for assistance in identifying information objects
Location	Questions asking about the location of a specific information object and/or source
Verification	Questions asking for assistance in verifying some information the user has
Recommendation	Questions asking for a list of music related information objects
Evaluation	Questions asking for an evaluative discussion of a particular subject (e.g., review)
Ready reference	Questions asking for simple, factual answers (e.g., birth/death date of an artist, record label)
Reproduction	Questions asking for text, taken directly from an information source and unchanged (e.g., lyrics)
Description	Questions asking for a description of something (e.g., description of artist, album)
Research	Questions asking for involved answers requiring some effort and wide use of information sources to formulate.
Other	(Questions that do not fit into any of the categories above)

Forms of needs

Music Search

- Dormant searches refer to relatively inactive searches that may become active again, typically triggered by a certain event. For example, a user may first hear some music and want to know what it is (they may even go on to attempt to search for it unsuccessfully); they give up due to lack of information about the music, or in time simply forget about the search. However, when the user happens to hear the music again, perhaps from some other source such as radio, television, or a movie, it reminds them about their previous music information need and reactivates their search for the music.
  - "I've heard the spooky tune, The Death March, several times tonight for Halloween. There are not words, just music. I've also heard the tune used in B-rated movies or cartoons to signify that someone or something has died.What is the origin of this tune? Who wrote it, when, and for what reason?"
- Many users also seem to search for music based on information from other people, especially when they are searching for lullabies or folk songs. For example, they heard a song repeatedly from a family member when they were younger and later they try to search for the right song often based on the fuzzy memory they have from their childhood (e.g., "My grandfather, who was born in 1899, used to sing me to sleep with this song and I can't remember the words"). This type of search can turn out to be quite difficult because often the information that was relayed to them in the first place might not be precise or accurate.
  - Q1: It has a female voice similar to Lora Logic or Linder Sterling from Ludus (I always thought this was a song by Ludus but now I have their complete discography and it isn't there).
  - Q2: Sounds like the Temptations or some band from the 1960s.
  - Q3: The whole thing sounds a little John Mayersque. In that genre, with Jack Johnson, Jason Mraz, etc.
  - Q4: They were the sort of hyper skinny 26 year old white guys you think of when you think of Red Hot chili peppers. . . sort of a repetitive pop song in the spirit of, say, FatBoy Slims "wonderful night". . . .

Music Search

TABLE 2. TOPICs of needs (FINAL).

Group	Description and examples of the typical forms of expressions
Lyrics	Questions asking for lyrics of a song • *I want to know the words for the song X.* • *What are the lyrics of the song X?*
Translation	Questions asking for the translation of lyrics of a song • *I need the translation of the lyrics of song X.*
Meaning	Questions asking for the meaning of lyrics of a song • *What is the meaning of the lyrics in song X?* • *In song X, artist Y refers to Z. What does Y mean by that?*
Score	Questions asking for/about music scores • *Where can I buy/download the score for work X?* • *I am looking for a guitar tab for song X.*
Work	Questions asking for musical works and/or information about musical works • *Name this song./What is the song that goes like. . .?* • *Where can I buy/download song X?* • *Can you recommend a list of songs suitable for novice Jazz listeners?* • *I am looking for the details of this work X. (e.g., origin of the song, released date)*
Version	Questions asking for a particular version of a musical work • *Where can I buy/download a techno remix version of the song X?* • *I am looking for a cover of song X by artist Y.*

Music Search

Recording	Questions asking for music recordings and/or information about music recordings • *Which album has the song X?* • *Where can I buy/download the album X?* • *Is the album X available on CD?* • *Can someone tell me more about the album X? (e.g., track listing, musicians featured, artwork, date released, availability in certain region)*
Related Work	Questions asking for information about non-musical/non-printed works related to music • *I want to locate a music video that is about. . .*
Genre	Questions asking for information about a musical genre • *What is the genre of song X?*
Artist	Questions asking for information about artists • *Who composed X?* • *Who sings the song that goes like. . .?* • *Which Symphony Orchestra is better? A or B? And why?* • *I want to know more about this composer Y. (e.g., birth/death date, discography)*
Publisher	Questions asking for information about record labels/publishers • *Which record company released the album X?* • *Who published this score X?*
Instrument	Questions asking for information about musical instruments • *What is the instrument played in this song X?* • *What is artist X playing in this album Y? Is it instrument A or B?* • *I need short summaries of the history of instrument X and Y.*
Statistics	Questions asking for music related statistical information • *I need the international sales chart for year 19xx.* • *What was the number one hit in the . . . chart in 19xx?*
Background	Questions asking for background information related to music (e.g., history, musical term) • *What is the difference between the music styles in region X and Y?* • *Please help me to understand the characteristics and differences of genre X and Y.* • *What does musical term X mean?*
Resource	Questions asking for specific information sources • *I want a list of websites where I can find more information about. . .* • *What are some good places to go to find new reggae music/download music/purchase albums/search by lyrics?*
Other	Questions that do not fit into any of the categories above

Queries from User Selected Text

Text Segment: TS	Chunks: C
Knee joint replacement may be recommended for: Severe arthritis (osteoarthritis or rheumatoid arthritis) of the knee that has not gotten better with medicine, injections, and physical therapy after 6 months or more of treatment. Your doctor may recommend knee replacement for these problems: Inability to sleep through the night because of knee pain. Knee pain that has not improved with other treatment. Knee pain that limits or keeps you from being able to do your normal activities, especially your daily activities such as bathing, preparing meals, household chores, and other things. Some tumors that affect the knee	household chores, knee replacement, knee pain, Severe arthritis, osteoarthritis, rheumatoid arthritis, injections, tumors, joint replacement, bathing, Inability, physical therapy, normal activities, knee, meals, daily activities, other treatment, medicine, treatment, 6 months

- People browsing the web or reading a document may see text passages that describe a topic of interest, and want to know more about it by searching.
- Generate effective queries from content of an arbitrary user selected text passage.
- Use CRF (proposed by [XHC10]) with rich features to identify discriminative phrases or words which could form the query.
- They tried various ways of weighting query terms – ones that incorporate #results from search API, CRF prob scores, removing stop words.

Tutorial Overview

- Properties of Verbose Queries [30 min]
- Techniques for Processing Verbose Web Queries
  - Query Reduction to a Single Sub-Query [50 min]
  - Query Reduction by Choosing Multiple Sub-Queries [10 min]
  - Break [30 min]
  - Weighting Query Words [90 min]
  - Lunch Break [90 min]
  - Query Expansion by Including Related Concepts [20 min]
  - Query Reformulation [30 min]
  - Query Segmentation [40 min]
  - Break [30 min]
  - Query-Dependent Learning to Rank [40 min]
- Applications of Verbose Query Processing [40 min]
- **Summary and Future Research Directions [10 min]**

Summary

- With various new ways of “querying”, there is a need to solve the problem of “answering verbose queries” more effectively.

- In this tutorial, we discussed six main ways of handling verbose queries
  - **query reduction**
  - **query weighting**
  - **query expansion**
  - **query reformulation**
  - **query segmentation**
  - **query – dependent learning to rank.**

- Finally, we also discussed various applications where supporting search for verbose queries can make a significant difference.

Future Research Directions

- **Unified Query Processing Framework**
  - We discussed multiple methods for query reduction, weighting, expansion, segmentation and reformulation. Each of these methods was shown to be effective on its own.
  - However, unifying all of them into a single principled framework remains an open research question.
  - Bundling all of these transformations within a query-dependent learning to rank framework is yet another open research question.

Future Research Directions

- **Multi-modal Verbose Query Processing**
  - Voice queries / mobile queries
    - Can retrieval methods effectively deal with speech recognition error, fat finger errors and bad auto-completes

Future Research Directions

- **Multi-modal Verbose Query Processing**

What is the name of the blue flower in this photo?
(*Geranium?*)

Future Research Directions

- **Search Personalization**
  - A simplistic way to model a user, in the context of search personalization, is a weighted combination of her past search queries
  - Query expansion algorithms can take into account prior user queries in addition to other information sources
  - Query weighting can be biased towards concepts which are known to be important to the user
  - Data sparsity can be addressed by user / query clustering or dimensionality reduction

Future Research Directions

- **Natural Language Query Understanding**
  - Truly general theoretical framework for NLP and IR synthesis has the potential to truly revolutionize search
    - Can answer arbitrarily complex requests
    - Applies to all types of tasks without a need for domain – specific manual tuning
    - Scales to large corpora
  - Some of the work presented today on information retrieval with verbose query may help to pave the way toward such a framework.

Thanks!

References

[ACC+96] James Allan, James P Callan, W Bruce Croft, Lisa Ballesteros, John Broglio, Jinxi Xu, and Hongming Shu. INQUERY at TREC-5. In Proc. of the Text REtrieval Conference (TREC), 1996.

[ACC+03] Susan Armstrong, Alexander Clark, Giovanni Coray, Maria Georgescul, Vincenzo Pallotta, Andrei Popescu-Belis, David Portabella, Martin Rajman, and Marianne Starlander. Natural Language Queries on Natural Language Data: A Database of Meeting Dialogues. In Proc. of the Intl. Conf. on Application of Natural Language to Information Systems (NLDB), volume 3, pages 14-27, 2003.

[ACR04] Giambattista Amati, Claudio Carpineto, and Giovanni Romano. Query Difficulty, Robustness, and Selective Application of Query Expansion. In Proc. of the 25th European Conf. on Information Retrieval (ECIR), pages 127-137, 2004.

[AGKK11] Rakesh Agrawal, Sreenivas Gollapudi, Anitha Kannan, and Krishnaram Kenthapadi. Enriching Textbooks with Images. In Proc. of the 20th ACM Intl. Conf. on Information and Knowledge Management (CIKM), pages 1847-1856, 2011.

[AGQ13] Elena Agapie, Gene Golovchinsky, and Pernilla Qvarfordt. Leading People to Longer Queries. In Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (SIGCHI), pages 3019-3022, 2013.

[AK08] Avi Arampatzis and Jaap Kamps. A Study of Query Length. In Proc. of the 31st Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 811-812, 2008.

[BC08] Michael Bendersky and W Bruce Croft. Discovering Key Concepts in Verbose Queries. In Proc. of the 31st Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 491-498, 2008.

[BC09] Michael Bendersky and W Bruce Croft. Analysis of Long Queries in a Large Scale Search Log. In Proc. of the 2009 Workshop on Web Search Click Data (WSCD), pages 8-14, 2009.

[BC12] Michael Bendersky and W Bruce Croft. Modeling Higher-Order Term Dependencies in Information Retrieval using Query Hypergraphs. In Proc. of the 35th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 941-950, 2012.

[BCS11] Michael Bendersky, W Bruce Croft, and David A Smith. Joint Annotation of Search Queries. In Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (HLT), pages 102-111, 2011.

References

[BKC10] Niranjan Balasubramanian, Giridhar Kumaran, and Vitor R Carvalho. Exploring Reductions for Long Web Queries. In Proc. of the 33rd Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 571-578, 2010.

[BMC10] Michael Bendersky, Donald Metzler, andWBruce Croft. Learning Concept Importance using aWeighted Dependence Model. In Proc. of the 3rd ACM Intl. Conf. on Web Search and Data Mining (WSDM), pages 31-40, 2010.

[BMC11] Michael Bendersky, Donald Metzler, and W Bruce Croft. Parameterized Concept Weighting in Verbose Queries. In Proc. of the 34th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 605-614, 2011.

[BMC12] Michael Bendersky, Donald Metzler, and W Bruce Croft. Effective Query Formulation with Multiple Information Sources. In Proc. of the 5th ACM Intl. Conf. on Web Search and Data Mining (WSDM), pages 443-452, 2012.

[BPS+11] Francesco Bonchi, Raffaele Perego, Fabrizio Silvestri, Hossein Vahabi, and Rossano Venturini. Recommendations for the Long Tail by Term-Query Graph. In Proc. of the 20th Intl. Conf. Companion on World Wide Web (WWW), pages 15-16, 2011.

[BT03] Judith L Bader and Mary Frances Theofanos. Searching for Cancer Information on the Internet: Analyzing Natural Language Search Queries. Journal of Medical Internet Research, 5(4), 2003.

[BW07] Shane Bergsma and Qin Iris Wang. Learning Noun Phrase Query Segmentation. In Proc. of the 2007 Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), volume 7, pages 819-826, 2007.

[BW+08] Krisztian Balog, Wouter Weerkamp, and Maarten de Rijke. A Few Examples go a Long Way: Constructing Query Models from Elaborate Query Formulations. In *Proc. of the* 31*st Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 371–378. ACM, 2008.

[BWLK10] Peter Bailey, Ryen W White, Han Liu, and Giridhar Kumaran. Mining Historic Query Trails to Label Long and Rare Search Engine Queries. ACM Transactions on the Web (TWEB), 4(4):15, 2010.

[CAO2008] Cao et al., Selecting Good Expansion Terms for Pseudo-Relevance Feedback, In proceedings of SIGIR 2008

[CD06] Fabio Crestani and Heather Du. Written versus Spoken Queries: A Qualitative and Quantitative Comparative Analysis. Journal of the American Society for Information Science and Technology (JASIST), 57(7):881-890, 2006.

[CLOJ11] Ronan Cummins, Mounia Lalmas, Colm O'Riordan, and Joemon M Jose. Navigating the User Query Space. In Proc. of the 18th Intl. Symposium on String Processing and Information Retrieval (SPIRE), pages 380-385, 2011.

[CZ09] Yan Chen and Yan-Qing Zhang. A Query Substitution-Search Result Refinement Approach for Long Query . In Proc. of the 2009 IEEE/WIC/ACM Intl. Joint Conf. on Web Intelligence and Intelligent Agent Technology-Volume 01 (WI-IAT), pages 245-251, 2009.

References

[DC10] Van Dang and Bruce W Croft. Query Reformulation using Anchor Text. In Proc. of the 3rd ACM Intl. Conf. on Web Search and Data Mining (WSDM), pages 41-50, 2010.

[DTHS10] Sergio Duarte Torres, Djoerd Hiemstra, and Pavel Serdyukov. An Analysis of Queries Intended to Search Information for Children. In Proc. of the 3rd Symposium on Information Interaction in Context, pages 235-244, 2010.

[DV11] Sudip Datta and Vasudeva Varma. Tossing Coins to Trim Long Queries. In Proc. of the 34th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 1255-1256, 2011.

[FK00] Kristofer Franzen and Jussi Karlgren. Verbosity and Interface Design. SICS Research Report, 2000.

[GENG2008] Geng et al., Query Dependent Ranking Using K-Nearest Neighbor. In proceedings of SIGIR 2008

[HC10] Samuel Huston and W Bruce Croft. Evaluating Verbose Query Processing Techniques. In Proc. of the 33rd Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 291-298, 2010.

[HPSB10] Matthias Hagen, Martin Potthast, Benno Stein, and Christof Braeutigam. The Power of Naive Query Segmentation. In Proc. of the 33rd Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 797-798, 2010.

[JF03] Rosie Jones and Daniel C Fain. Query Word Deletion Prediction. In Proc. of the 26th Annual Intl. ACM SIGIR Conf. on Research and Development in Informaion Retrieval (SIGIR), pages 435-436, 2003.

[JRMG06] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. Generating Query Substitutions. In Proc. of the 15th Intl. Conf. on World Wide Web (WWW), pages 387-396, 2006.

[JZ07] Jing Jiang and Chengxiang Zhai. An Empirical Study of Tokenization Strategies for Biomedical Information Retrieval. Information Retrieval, 10(4-5):341-363, 2007.

[KA07] Giridhar Kumaran and James Allan. A Case For Shorter Queries, and Helping Users Create Them. In Proc. of the Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL), pages 220-227, 2007.

[DBC13] Van Dang, Michael Bendersky and Bruce Croft, Two-stage learning to rank for information retrieval, In proceedings of ECIR, 2013

References

[KA08] Giridhar Kumaran and James Allan. Effective and Efficient User Interaction for Long Queries. In Proc. of the 31st Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 11-18, 2008.

[KC09] Giridhar Kumaran and Vitor R Carvalho. Reducing Long Queries using Query Quality Predictors. In Proc. of the 32nd Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 564-571, 2009.

[LAC09] Matthew Lease, James Allan, and W Bruce Croft. Regression Rank: Learning to Meet the Opportunity of Descriptive Queries. In Proc. of the 31th European Conf. on IR Research on Advances in Information Retrieval (ECIR), pages 90-101, 2009.

[LC03] Victor Lavrenko and W Bruce Croft. Relevance Models in Information Retrieval. In Language Modeling for Information Retrieval, pages 11-56. Springer, 2003.

[LC12a] Chia-Jung Lee and W Bruce Croft. Generating Queries from User-Selected Text. In Proc. of the 4th Symposium on Information Interaction in Context, pages 100-109, 2012.

[LC12b] Chee Wee Leong and Silviu Cucerzan. Supporting Factual Statements with Evidence from the Web. In Proc. of the 21st ACM Intl. Conf. on Information and Knowledge Management (CIKM), pages 11531162, 2012.

[LCKC09] Chia-Jung Lee, Ruey-Cheng Chen, Shao-Hang Kao, and Pu-Jen Cheng. A Term Dependency-based Approach for Query Terms Ranking. In Proc. of the 18th ACM Conf. on Information and Knowledge Management (CIKM), pages 1267-1276, 2009.

[Lea09] Matthew Lease. An Improved Markov Random Field Model for Supporting Verbose Queries. In Proc. of the 32nd Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 476-483, 2009.

[Lee10] Jin Ha Lee. Analysis of User Needs and Information Features in Natural Language Queries seeking Music Information. Journal of the American Society for Information Science and Technology (JASIST), 61(5):1025-1045, 2010.

[LH99] Tessa Lau and Eric Horvitz. Patterns of Search: Analyzing and Modeling Web Query Refinement. In Proc. of the 7th Intl. Conf. on User Modeling (UM), pages 119-128, 1999.

[LI12] Li et al., QRU-1: A Public Dataset for Promoting Query Representation and Understanding Research, In proceedings of the Workshop on Web Search Click Data at WSDM 2012

[LIU08] Tie-Yan Liu Learning to Rank for Information Retrieval, Tutorial at SIGIR 2008

References

[LLCC09] Chia-Jung Lee, Yi-Chun Lin, Ruey-Cheng Chen, and Pu-Jen Cheng. Selecting Effective Terms for Query Formulation. In Proc. of the 5th Asia Information Retrieval Symposium on Information Retrieval Technology (AIRS), pages 168-180, 2009.

[LO08] Christina Lioma and Iadh Ounis. A Syntactically-based Query Reformulation Technique for Information Retrieval. Information processing & management (IPM), 44(1):143-162, 2008.

[LU2009] Lu et al., Improving Web Search Relevance with Semantic Features, In proceedings of EMNLP, 2009

[LYJ06] Yunyao Li, Huahai Yang, and HV Jagadish. Constructing a Generic Natural Language Interface for an XML Database. In Proc. of the 2006 Conf. on Advances in Database Technology (EDBT), pages 737-754, 2006.

[MC05] Donald Metzler and W. Bruce Croft. A Markov Random Field Model for Term Dependencies. In Proc. of the 28th Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 472-479, 2005.

[MC07] Donald Metzler and W Bruce Croft. Latent Concept Expansion using Markov Random Fields. In Proc. of the 30th Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 311-318, 2007.

[MC13] K Tamsin Maxwell and W Bruce Croft. Compact Query Term Selection using Topically Related Text. In Proc. of the 36th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 583-592, 2013.

[MSRG+11] Nikita Mishra, Rishiraj Saha Roy, Niloy Ganguly, Srivatsan Laxman, and Monojit Choudhury. Unsupervised Query Segmentation using Only Query Logs. In Proc. of the 20th Intl. Conf. on World Wide Web (WWW), pages 91-92, 2011.

[PALL07] Fuchun Peng, Nawaaz Ahmed, Xin Li, and Yumao Lu. Context Sensitive Stemming for Web Search. In Proc. of the 30th Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 639-646, 2007.

[PBW07] Nina Phan, Peter Bailey, and Ross Wilkinson. Understanding the Relationship of Information Need Specificity to Search Query Length. In Proc. of the 30th Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 709-710, 2007.

[PC98] Jay M Ponte and W Bruce Croft. A Language Modeling Approach to Information Retrieval. In Proc. of the 21st Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 275-281, 1998.

References

[PC10] Jae Hyun Park and W Bruce Croft. Query Term Ranking based on Dependency Parsing of Verbose Queries. In Proc. of the 33rd Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 829-830, 2010.

[PCS11] Jae Hyun Park, W Bruce Croft, and David A Smith. A Quasi-Synchronous Dependence Model for Information Retrieval. In Proc. of the 20th ACM Intl. Conf. on Information and Knowledge Management (CIKM), pages 17-26, 2011.

[PO14] Jiaul H. Paik and Douglas W. Oard. A Fixed-Point Method for Weighting Terms in Verbose Informational Queries. In Proc. of the 23rd ACM Conf. on Information and Knowledge Management (CIKM), pages 131-140, 2014.

[PSAH13] Nish Parikh, Prasad Sriram, and Mohammad Al Hasan. On Segmentation of E-Commerce Queries. In Proc. of the 22nd ACM Intl. Conf. on Information and Knowledge Management (CIKM), pages 1137-1146, 2013.

[RMB03] Knut Magne Risvik, Tomasz Mikolajewski, and Peter Boros. Query Segmentation for Web Search. In Proc. of the 12th Intl. Conf. on World Wide Web (WWW), 2003.

[SKK10] Krysta M. Svore, Pallika H. Kanani, and Nazan Khan. How Good is a Span of Terms?: Exploiting Proximity to Improve Web Retrieval. In Proc. of the 33rd Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 154-161, 2010.

[SPS12] Gyanit Singh, Nish Parikh, and Neel Sundaresan. Rewriting Null E-Commerce Queries to Recommend Products. In Proc. of the 21st Intl. Conf. Companion on World Wide Web (WWW), pages 73-82, 2012.

[SSSC11] Daniel Sheldon, Milad Shokouhi, Martin Szummer, and Nick Craswell. LambdaMerge: Merging the Results of Query Reformulations. In Proc. of the 4th ACM Intl. Conf. on Web Search and Data Mining (WSDM), pages 795-804, 2011.

[TP08] Bin Tan and Fuchun Peng. Unsupervised Query Segmentation using Generative Language Models and Wikipedia. In Proc. of the 17th Intl. Conf. on World Wide Web (WWW), pages 347-356, 2008.

[WZ08] Xuanhui Wang and ChengXiang Zhai. Mining Term Association Patterns from Search Logs for Effective Query Reformulation. In Proc. of the 17th ACM Conf. on Information and Knowledge Management (CIKM), pages 479-488, 2008.

References

[XC11] Xiaobing Xue and W Bruce Croft. Modeling Subset Distributions for Verbose Queries. In Proc. of the 34th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 1133-1134, 2011.

[XC12] Xiaobing Xue and W Bruce Croft. Generating Reformulation Trees for Complex Queries. In Proc. of the 35th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 525-534, 2012.

[XHC10] Xiaobing Xue, Samuel Huston, and W Bruce Croft. Improving Verbose Queries using Subset Distribution. In Proc. of the 19th ACM Intl. Conf. on Information and Knowledge Management (CIKM), pages 1059-1068, 2010.

[XJC08] Xiaobing Xue, Jiwoon Jeon, and W. Bruce Croft. Retrieval Models for Question and Answer Archives. In Proc. of the 31st Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 475-482, 2008.

[XTJL12] Xiaobing Xue, Yu Tao, Daxin Jiang, and Hang Li. Automatically Mining Question Reformulation Patterns from Search Log Data. In Proc. of the 50th Annual Meeting of the Association for Computational Linguistics (ACL), pages 187-192, 2012.

[XU09] Xu et al., Query Dependent Pseudo-Relevance Feedback based on Wikipedia, In Proc. of SIGIR, 2009

[YM11] Jeonghe Yi and Farzin Maghoul. Mobile Search Pattern Evolution: The Trend and the Impact of Voice Queries. In Proc. of the 20th Intl. Conf. Companion on World Wide Web (WWW), pages 165-166, 2011.

[YPSS14] Bishan Yang, Nish Parikh, Gyanit Singh, and Neel Sundaresan. A Study of Query Term Deletion Using Large-Scale E-commerce Search Logs. In Proc. of the 36th European Conf. on Information Retrieval (ECIR), pages 235-246, 2014.

[ZC10] Le Zhao and Jamie Callan. Term Necessity Prediction. In Proc. of the 19th ACM Intl. Conf. on Information and Knowledge Management (CIKM), pages 259-268, 2010.

[ZL01] Chengxiang Zhai and John Lafferty. A Study of Smoothing Methods for Language Models applied to Ad hoc Information Retrieval. In Proc. of the 24th Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 334-342, 2001.

[ZL04] Chengxiang Zhai and John Lafferty. A Study of Smoothing Methods for Language Models applied to Information Retrieval. ACM Transactions on Information Systems (TOIS), 22(2):179-214, 2004.