Learning Concept Importance Using a Weighted Dependence Model

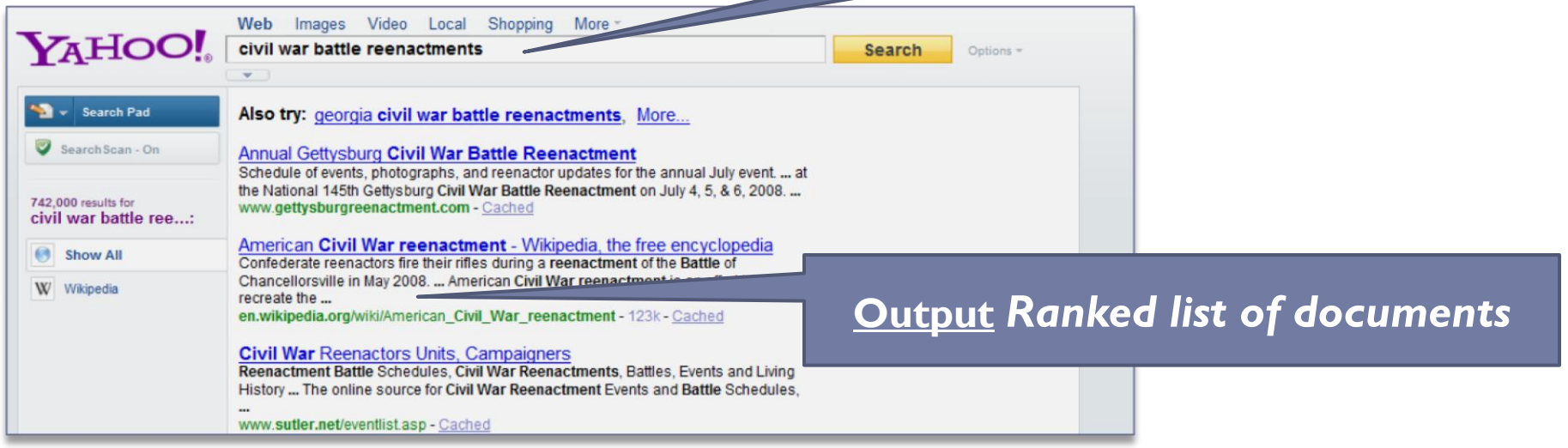
Michael Bendersky UMass Amherst

Donald Metzler Yahoo! Research

W. Bruce Croft UMass Amherst

WSDM 2010, New York

Ranked Retrieval



- Search engine must ***accurately*** interpret query intent
  - Detect phrases
    - ***new york times*** ≠ ***time new york***
  - Detect relative term/phrase importance
    - ***CONTINENTAL airline BOOKING***

Term-Based Models

- Term-based retrieval models treat the user's query as a "***bag-of-words***"
  - BM25 *(Robertson et al., 2000)*
  - Query Likelihood *(Ponte & Croft, 1998)*
  - DFR *(Amati, 2003)*

- A simple query model
  - Term order is interchangeable
  - Simple collection-based heuristics to weight query terms
    - e.g., ***IDF***
  - Term weights do not vary based on their context

Concept-Based Models

- Recently, researchers focused on incorporating term dependence into the term-based models
  - Markov Random Fields for IR *(Metzler & Croft, 2005)*
  - BM25 with term proximities *(Song et al., 2008)*
  - DFR-SD, DFR-FD *(Peng et al., 2007)*
- A more realistic query model
  - Term order is important
  - Captures concepts - dependencies between query terms
- However, concept weighting is still ***rigid and ad-hoc***
  - e.g., ***IDF***

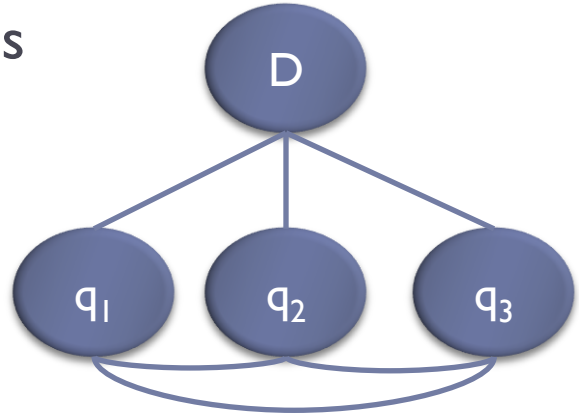
What is a Concept?

- A concept is any syntactic expression that can be **matched** within a document
  - Practical definition for IR

- Examples
  - Unigrams
    - Match each of the terms "white", "house"
  - Exact phrases
    - match exact phrase "white house"
  - Proximities
    - match unordered phrase "white house" in a window of ***K*** terms

MRFs for IR in a Nutshell

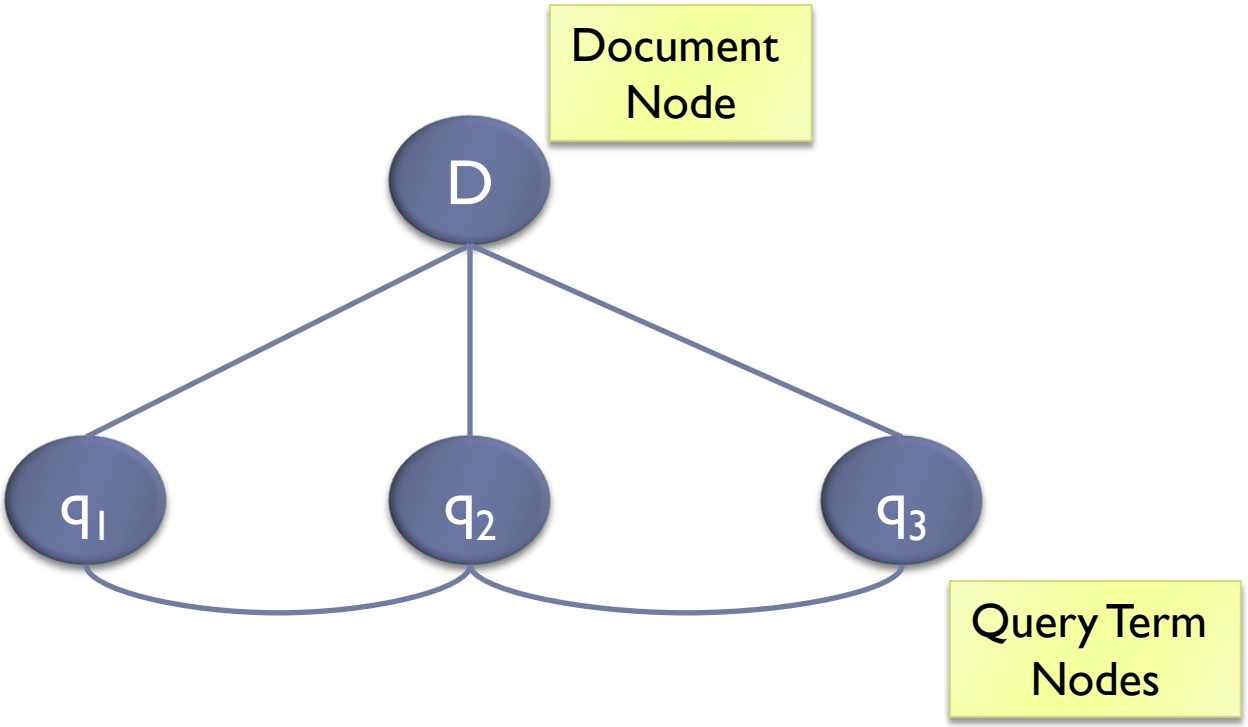
- Encode document and query terms in a graph ***G***
  - **vertices** represent document/query nodes
  - **edges** encode dependence semantics



- Potentials over the cliques of ***G***
  - Non-negative functions over clique configurations
  - Measure query-document "match"

- Score the document using the joint probability mass function over ***G***

MRF - Sequential Dependence Model (SD)



- Assume dependence between adjacent terms
- Effectiveness/Efficiency tradeoff
- Empirically proven retrieval performance

SD Ranking Function

- Associate each clique in the graph with one or more potential function ***f***

$$P(D|Q) \overset{rank}{=} \lambda_T \sum_{q \in Q} f_T(q, D) + \lambda_O \sum_{q_i, q_{i+1} \in Q} f_O(q_i, q_{i+1}, D) + \lambda_U \sum_{q_i, q_{i+1} \in Q} f_U(q_i, q_{i+1}, D)$$

how well does q match D?
*[**bag of words** score]*

how well does "q_i q_{i+1}" match D?
*[**exact phrase** score]*

how well does prox(q_i q_{i+1}) match D?
*[**proximity** score]*

Limitations of SD

$$P(D|Q) \overset{rank}{=} \lambda_T \sum_{q \in Q} f_T(q, D) + \lambda_O \sum_{q_i, q_{i+1} \in Q} f_O(q_i, q_{i+1}, D) + \lambda_U \sum_{q_i, q_{i+1} \in Q} f_U(q_i, q_{i+1}, D)$$

- Parameter tying
  - All matches of the same type are equally weighted
  - Especially detrimental for verbose queries
- Instead, we'd like query concept weights to vary

Weighted Sequential Dependence Model (WSD)

- 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

Defining Concept Importance

- Features **g** define the concept importance
- Depend on the concept (term/bigram)

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

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

- Independent of a specific document/document corpus
- Combine several sources for more accurate weighting
  - ***Endogenous Features*** – collection dependent features
  - ***Exogenous Features*** – collection independent features

Concept Importance Features

	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*

- **Unigram concepts**: *all features (7)*
- **Bigram concepts**: *all features (7) + PMI for each data source (4)*
- **Total features**: ***18***
- *All features are log-scaled and normalized*

WSD Ranking Function

- **Score document D by:**

$$P(D|Q) \overset{rank}{=} \sum_{i=1}^{k_u} w_i^u \sum_{q \in Q} g_i^u(q) f_T(q, D) +$$

$$\sum_{i=1}^{k_b} w_i^b \sum_{q_j, q_{j+1} \in Q} g_i^b(q_j, q_{j+1}) f_O(q_j, q_{j+1}, D) +$$

$$\sum_{i=1}^{k_b} w_i^b \sum_{q_j, q_{j+1} \in Q} g_i^b(q_j, q_{j+1}) f_U(q_j, q_{j+1}, D)$$

- Note that ***WSD*** model is also linear (with respect to ***w***)

Direct Optimization

- Learn the weights ***w*** to directly optimize a retrieval performance metric
  - *MAP*
  - *NDCG*

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

- We use a coordinate-level ascent algorithm
  - Efficient for a small number of parameters
  - Empirically good performance
- However, most other LR4IR methods can be easily adopted for optimization

Query "civil war battle reenactments"

Concept	Importance Features			Weight
	GF	...	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*

Concept weights may vary even if concept DF is similar

Query "civil war battle reenactments"

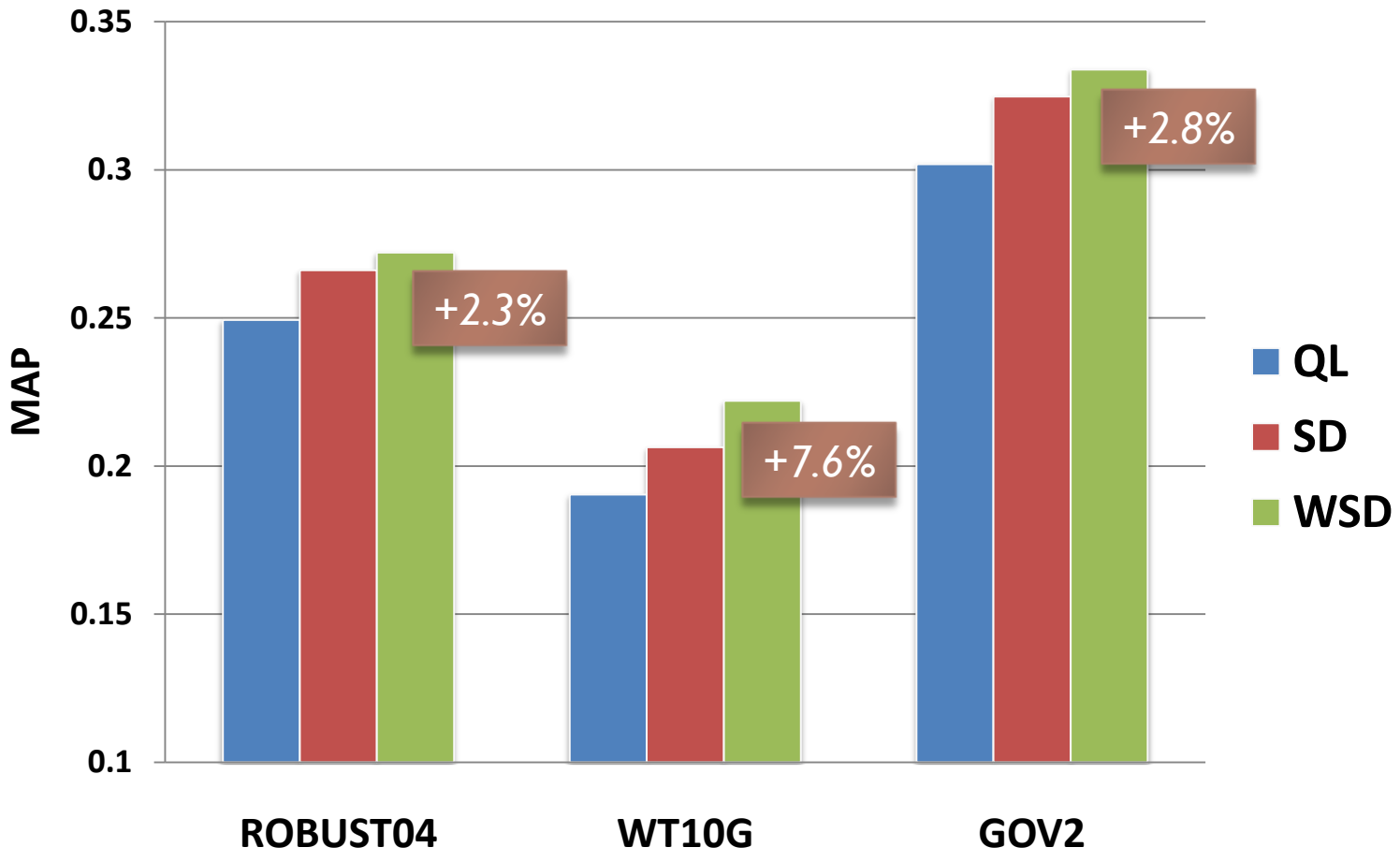
Concept	Importance Features			Weight
	GF	...	DF	
civil	16.9		14.1	*0.0619*
war	17.9		12.8	*0.1947*
battle	16.6		12.6	*0.0913*
reenactments	10.8		9.7	*0.3487*
civil war	**14.5**		**10.8**	***0.1959***
war battle	**9.5**		**7.4**	***0.2458***
battle reenactments	7.6		4.7	*0.0540*

Good segments do not necessarily predict important concepts

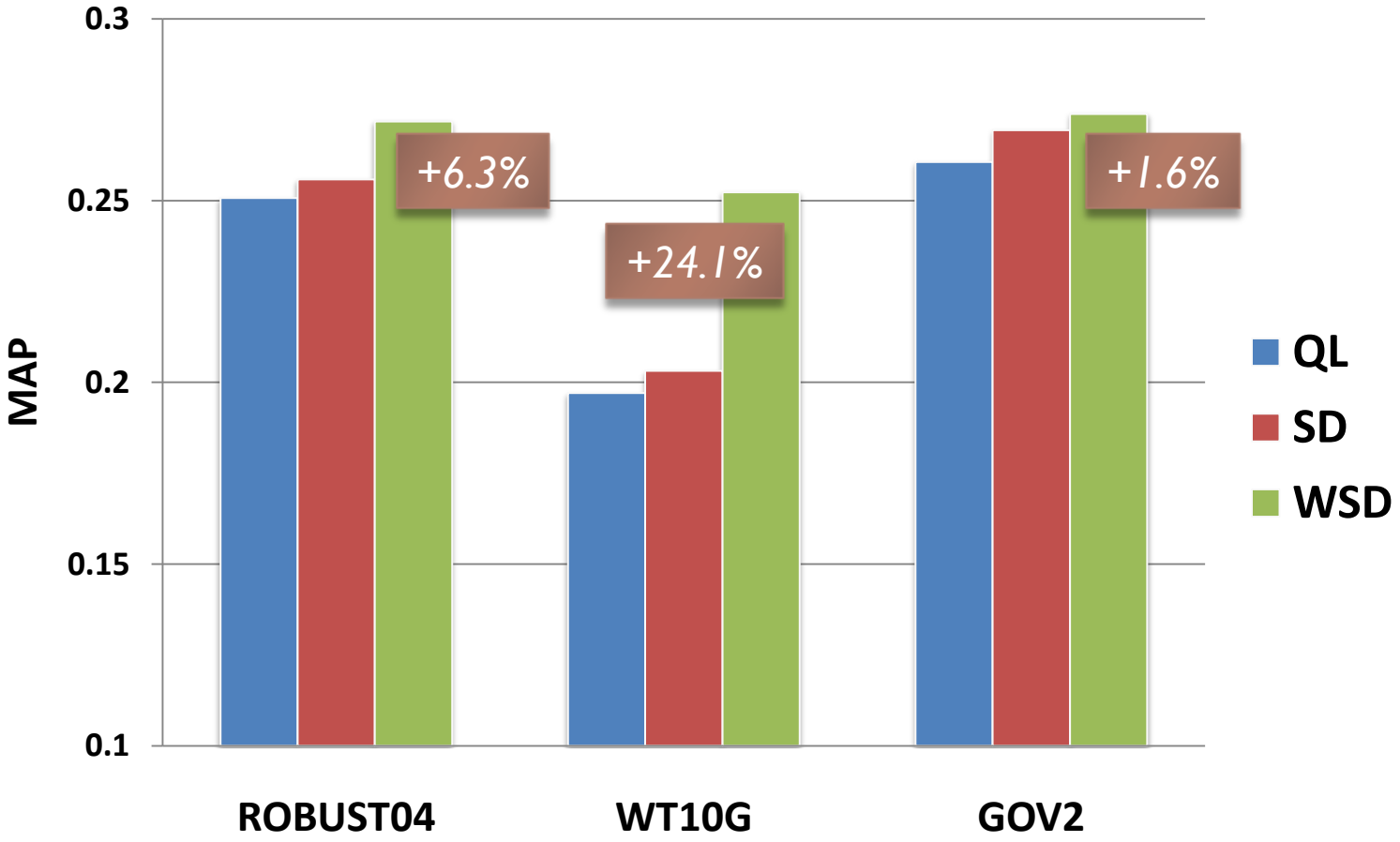
Experimental Results

- A detailed evaluation of our approach
  - ***TREC and web*** document collections
  - ***Short & Long*** queries
  - Contribution of *different* ***feature types***
  - Contribution of *different* ***concept types***

TREC Title (Short) Queries



TREC Description (Long) Queries



Endogenous & Exogenous Features

- Results with using ***either endogenous or exogenous*** features alone are comparable
- Using both types of features improves the performance over the unweighted sequential dependence model (SD)
- In most cases combining both types of features results in better performance

Term & Bigram Weights

- For short web queries (1-3 terms)
  - Bigram weights have more impact than term weights
- For TREC queries and longer web queries
  - Unigram weights have more impact than bigram weights
- In most cases combining both types of weights results in better performance, especially for longer queries

Web Queries

	DCG@1	DCG@5	DCG
QL	0.629	1.691	5.844
SD	0.864	2.383	6.681
WSD	0.884 ***(+2.3%)***	2.443 ***(+2.5%)***	6.741 ***(+0.9%)***

- Results using a large-scale commercial web search test collection
- A sample of long web search queries *(length 4+)*
- A total of 1,000 queries with 5-fold CV
- All improvements are stat. significant *(Wilcoxon sign test, $p < 0.05$)*

Conclusions

- Existing retrieval methods can be enhanced by
  - More accurate ***modeling*** of query concepts
  - More accurate ***weighting*** of query concepts
- Concept weight should be determined by a combination of both endogenous and exogenous features
- Dynamic concept weighting leads to significant improvements, especially for long queries

We would like to thank WSDM'10 Conference for the Student Travel Award!

Thank you!