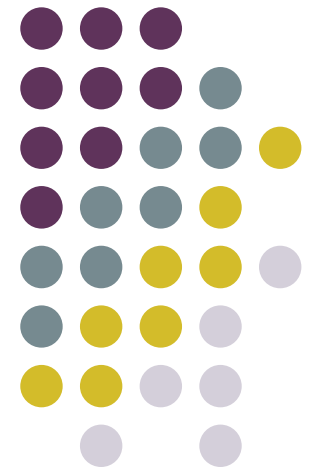


Incorporating Language Modeling into the Inference Network Retrieval Framework

Don Metzler





Motivation

- Great deal of information lost when forming queries
 - Example: “stemming information retrieval”
- InQuery
 - informal (*tf.idf* observation estimates)
 - structured queries via inference network framework
- Language Modeling
 - formal (probabilistic model of documents)
 - unstructured
- InQuery + Language modeling
 - formal
 - structured



Motivation

- Simple idea:
 - Replace *tf.idf* estimates in inference network framework with language modeling estimates
 - Result is a system based on ideas from language modeling that allows powerful structured queries
- Overall goal:
 - Do as well as, or better than, InQuery within this more formal framework



Outline

- Review
 - Inference Network Framework
 - Language Modeling
- Combined Approach
- Results

Review of Inference Networks

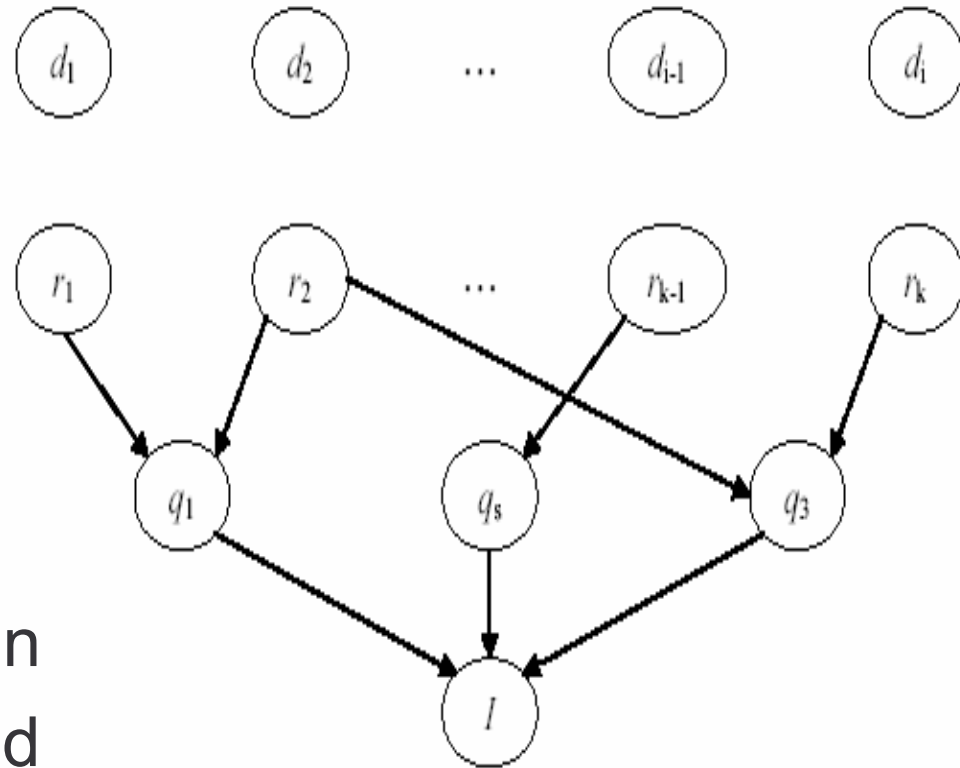


- Directed acyclic graph
- Compactly represents joint probability distribution over a set of continuous and/or discrete random variables
- Each node has a conditional probability table associated with it
- Network topology defines conditional independence assumptions among nodes
- In general, inference is NP-hard



Inference Network Framework

- Node types
 - document (d_i)
 - concept (r_i)
 - query (q_i)
 - information need (I)
- Set evidence at document nodes
- Run belief propagation
- Documents are scored by $P(I = \text{true} \mid d_i = \text{true})$





Network Semantics

- All events in network are binary
- Events associated with each node:
 - d_i – document i is observed
 - r_i – representation concept i is observed
 - q_i – query representation i is observed
 - l – information need is satisfied



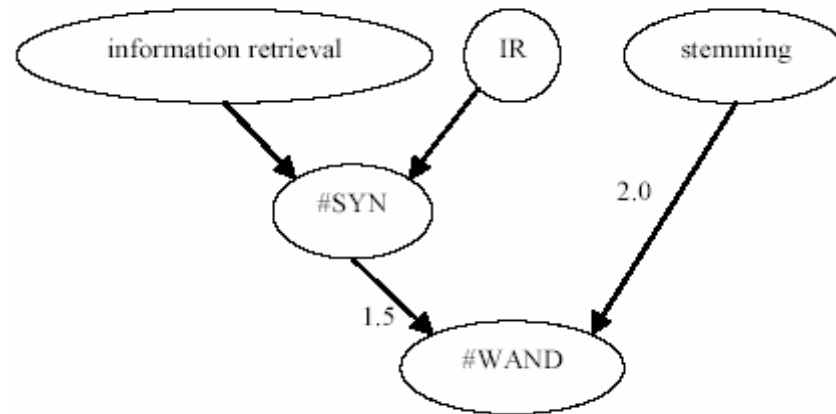
Query Language

Operator	Name	Description
#ODN ($q_1 \dots q_n$)	ordered window	A match occurs if the q_i 's appear in order with no more than N words between adjacent terms.
#UWN ($q_1 \dots q_n$)	unordered window	Similar to #ODN , except terms may appear unordered.
#PHRASE ($q_1 \dots q_n$)	phrase	Equivalent to #OD3($q_1 \dots q_n$).
#SYN ($q_1 \dots q_n$)	synonym	The q_i 's are to be treated as synonyms. Each q_i is treated as a match.
#PASSAGEN (q)	passage	Evaluates q 's belief for every passage of length N within a document and returns the highest belief.

Belief Operators	Weighted Belief Operators
#NOT (q_1)	#WSUM ($w_1 q_1 \dots w_n q_n$)
#AND ($q_1 \dots q_n$)	#WAND ($w_1 q_1 \dots w_n q_n$)
#OR ($q_1 \dots q_n$)	
#MAX ($q_1 \dots q_n$)	
#SUM ($q_1 \dots q_n$)	



Example Query



Unstructured:

stemming information retrieval

Structured:

```
#wand(1.5 #syn(#phrase(information retrieval) IR)  
2.0 stemming)
```



Belief Propagation

- Want to compute $\text{bel}(n)$ for each node n in the network ($\text{bel}(n) = P(n = \text{true} \mid d_i = \text{true})$)
- Term/proximity node beliefs (InQuery)

$$\text{bel}(r) = db + (1 - db) \overline{tf}_{r,d_i} idf_r$$

db = default belief

$$\overline{tf}_{r,d_i} = \frac{tf_{r,d_i}}{tf_{r,d_i} + 0.5 + 1.5 \frac{|d_i|}{|D|_{avg}}}$$

tf_{r,d_i} = number of times representation r is matched in document d_i

$$idf_r = \frac{\log\left(\frac{|C| + 0.5}{tf_{r,d_i}}\right)}{\log(|C| + 1)}$$

$|d_i|$ = length of document i

$|D|_{avg}$ = average doc. length

$|C|$ = collection length



Belief Nodes

- In general, marginalization is very costly
- Assuming a nice functional form, via link matrices, marginalization becomes easy
- p_1, \dots, p_n are the beliefs at the parent nodes of q
- $W = w_1 + \dots + w_n$

$$L_{and} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$bel_{not}(q) = 1 - p_1$$

$$bel_{or}(q) = 1 - \prod_i (1 - p_i)$$

$$bel_{max}(q) = \max(p_1, \dots, p_n)$$

$$bel_{sum}(q) = \frac{\sum_i p_i}{n}$$

$$bel_{wsum}(q) = \frac{\sum_i w_i p_i}{W}$$

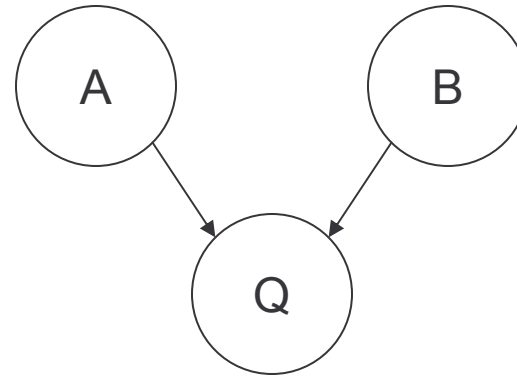
$$bel_{and}(q) = \prod_i p_i$$

$$bel_{wand}(q) = \prod_i p_i^{(w_i/W)}$$



Link Matrix Example

$$L_{and} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



$$\begin{aligned} P_{and}(q = true) &= p_{00}P(a = false)P(b = false) \\ &+ p_{01}P(a = false)P(b = true) \\ &+ p_{10}P(a = true)P(b = false) \\ &+ p_{11}P(a = true)P(b = true) \\ &= 0(1 - p_a)(1 - p_b) + 0(1 - p_a)p_b + 0p_a(1 - p_b) + 1p_ap_b \\ &= p_ap_b \end{aligned}$$



Language Modeling

- Models document generation as a stochastic process
- Assume words are drawn i.i.d. from an underlying multinomial distribution
- Use smoothed maximum likelihood estimate:

$$P(w | \theta_d) = \lambda \frac{tf_{w,d}}{|d|} + (1 - \lambda) \frac{cf_w}{|C|}$$

- Query likelihood model:

$$P(Q = q_1 \dots q_n | \theta_d) = \prod_{q \in Q} P(q | \theta_d)$$



Inference Network + LM

- Rather than use *tf.idf* estimates for $\text{bel}(r)$, use smoothed language modeling estimates:

$$\text{bel}(r) = P(r | d_i)$$

$$P(r | d_i) = \lambda \frac{tf_{r,d_i}}{|d_i|} + (1 - \lambda) \frac{cf_r}{|C|}$$

- Use Jelinek-Mercer smoothing throughout for simplicity



Combining Evidence

- InQuery combines query evidence via `#wsum` operator – i.e. all queries are of the form `#wsum(...)`
- `#wsum` does not work for combined model
 - resulting scoring function lacks *idf* component
- Must use `#wand` instead
- Can be interpreted as normalized weighted averages
 - arithmetic (InQuery)
 - geometric (combined model)



#wsum VS. #wand

$$\begin{aligned}P_{wsum}(I = true) &= \frac{\sum_i w_i p_i}{\sum_i w_i} \\&= \frac{\sum_i w_i (\lambda \frac{tf_{q_i, d_j}}{|d_j|} + (1 - \lambda) \frac{cf_{q_i}}{|C|})}{\sum_i w_i} \\&\propto \sum_i w_i \frac{tf_{q_i, d_j}}{|d_j|}\end{aligned}$$

$$\begin{aligned}P_{wand}(I = true) &= \prod_i p_i^{w_i} \\&= \prod_i (\lambda \frac{tf_{q_i, d_j}}{|d_j|} + (1 - \lambda) \frac{cf_{q_i}}{|C|})^{w_i}\end{aligned}$$



Relation to Query Likelihood

- Model subsumes query likelihood model
- Given a query $Q = q_1, q_2, \dots, q_n$ (q_i is a single term) convert it to the following structured query:

`#and(q1 q2 ... qn)`

- Result is query likelihood model



Smoothing

- InQuery – crude smoothing via “default belief”
- Proximity node smoothing
 - Single term smoothing
 - Other proximity node smoothing
- Each type of proximity node can be smoothed differently



Experiments

- Data sets
 - TREC 4 *ad hoc* (manual & automatic queries)
 - TREC 6, 7, and 8 *ad hoc*
- Comparison
 - Query likelihood (QL)
 - InQuery
 - Combined approach (StructLM)
- Single term node smoothing $\lambda = 0.6$
- Other proximity node smoothing $\lambda = 0.1$



Example Query

- **Topic:** “Is there data available to suggest that capital punishment is a deterrent to crime?”
- **Manual structured query:**

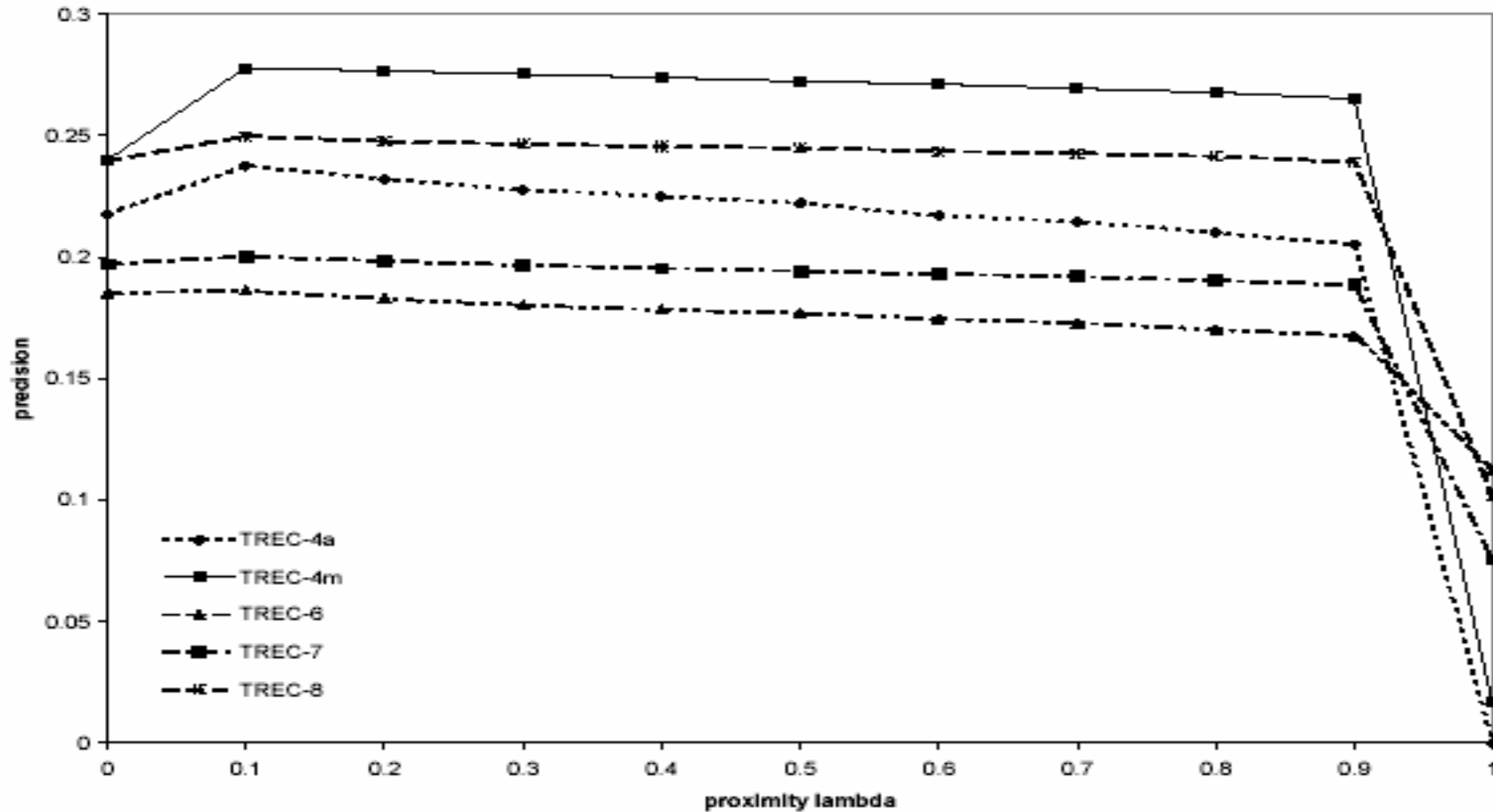
```
#wsum(1.0 #wsum(1.0 capital 1.0 punishment
          1.0 deterrent 1.0 crime
          2.0 #uw20(capital punishment deterrent)
          1.0 #phrase(capital punishment)
1.0 #passage200 (1.0 capital 1.0 punishment
                 1.0 deterrent 1.0 crime
                 1.0 #phrase(capital punishment)))
```



	QL		InQuery		StructLM	
	Auto	Manual	Auto	Manual	Auto	Manual
Rel	6086	6086	6086	6086	6086	6086
Rret	3190	3371	3306	3679	3355	3737
0.0	0.6761	0.7166	0.7188	0.7896	0.6888	0.7893
0.1	0.4796	0.5082	0.4944	0.5601	0.4983	0.5479
0.2	0.3801	0.4156	0.3942	0.4581	0.3926	0.4486
0.3	0.3220	0.3465	0.3310	0.3883	0.3300	0.3997
0.4	0.2672	0.2960	0.2844	0.3317	0.2769	0.3329
0.5	0.2087	0.2315	0.2241	0.2552	0.2268	0.2631
0.6	0.1546	0.1708	0.1622	0.1849	0.1713	0.2079
0.7	0.0903	0.1033	0.0975	0.1236	0.1331	0.1520
0.8	0.0480	0.0567	0.0544	0.0727	0.0763	0.0894
0.9	0.0056	0.0175	0.0194	0.0300	0.0246	0.0422
1.0	0.0021	0.0038	0.0016	0.0014	0.0048	0.0024
Avg	0.2179	0.2397	0.2312	0.2688	0.2376	0.2779
5	0.5020	0.5102	0.5265	0.6082	0.5020	0.5796
10	0.4510	0.4714	0.4735	0.5551	0.4531	0.5490
15	0.4204	0.4422	0.4190	0.5034	0.4190	0.5007
20	0.3959	0.4235	0.4071	0.4694	0.3969	0.4602
30	0.3544	0.3748	0.3578	0.4211	0.3571	0.4156
100	0.2376	0.2516	0.2412	0.2794	0.2380	0.2843
200	0.1735	0.1856	0.1771	0.2039	0.1747	0.2064
500	0.1030	0.1114	0.1071	0.1192	0.1067	0.1227
1000	0.0651	0.0688	0.0675	0.0751	0.0685	0.0763
RPr	0.2761	0.2904	0.2763	0.3174	0.2872	0.3282

	QL	InQuery	StructLM
TREC-6	0.1854	0.1622	0.1863
TREC-7	0.1972	0.1803	0.2004
TREC-8	0.2396	0.2343	0.2498

Proximity Node Smoothing





Conclusions

- Good structured queries help
- Combines inference network's structured query language with formal language modeling probability estimates
- Performs competitively against InQuery
- Subsumes query likelihood model



Future Work

- Smoothing
 - Try other smoothing techniques
 - Find optimal parameters for each node type
- Combine LM and *tf.idf* document representations
- Other estimates for $\text{bel}(r)$
- Theoretical considerations