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 \((l)\)
- Set evidence at document nodes
- Run belief propagation
- Documents are scored by \(P(l = true \mid d_i = true)\)
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

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

<table>
<thead>
<tr>
<th>Belief Operators</th>
<th>Weighted Belief Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>#NOT (q_1)</td>
<td>#WSUM ((w_1 q_1 \ldots w_n q_n))</td>
</tr>
<tr>
<td>#AND ((q_1 \ldots q_n))</td>
<td>#WAND ((w_1 q_1 \ldots w_n q_n))</td>
</tr>
<tr>
<td>#OR ((q_1 \ldots q_n))</td>
<td></td>
</tr>
<tr>
<td>#MAX ((q_1 \ldots q_n))</td>
<td></td>
</tr>
<tr>
<td>#SUM ((q_1 \ldots q_n))</td>
<td></td>
</tr>
</tbody>
</table>
Example Query

Unstructured:
stemming information retrieval

Structured:
#wand(1.5 #syn(#phrase(information retrieval) IR) 2.0 stemming)
Belief Propagation

- Want to compute bel(n) for each node n in the network (bel(n) = P(n = true | d_i = true))

- Term/proximity node beliefs (InQuery)

\[
bel(r) = db + (1 - db) \bar{tf}_{r,d_i} idf_r
\]

\[
\bar{tf}_{r,d_i} = \frac{tf_{r,d_i}}{tf_{r,d_i} + 0.5 + 1.5 \frac{|d_i|}{|D|_{avg}}}
\]

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

\[
db = \text{default belief}
\]

\[
|d_i| = \text{length of document } i
\]

\[
|D|_{avg} = \text{average doc. length}
\]

\[
|C| = \text{collection length}
\]
Belief Nodes

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

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

\[
\begin{align*}
bel_{\text{not}}(q) &= 1 - p_1 \\
bel_{\text{or}}(q) &= 1 - \prod_i (1 - p_i) \\
bel_{\text{max}}(q) &= \max(p_1, \ldots, p_n) \\
bel_{\text{sum}}(q) &= \sum_i p_i \\
bel_{\text{wsum}}(q) &= \frac{\sum_i w_i p_i}{W} \\
bel_{\text{and}}(q) &= \prod_i p_i \\
bel_{\text{wand}}(q) &= \prod_i p_i^{(w_i/W)}
\end{align*}
\]
Link Matrix Example

\[ L_{\text{and}} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \]

\[ P_{\text{and}}(q = \text{true}) = p_{00} P(a = \text{false}) P(b = \text{false}) + p_{01} P(a = \text{false}) P(b = \text{true}) + p_{10} P(a = \text{true}) P(b = \text{false}) + p_{11} P(a = \text{true}) P(b = \text{true}) \\
= 0(1 - p_a)(1 - p_b) + 0(1 - p_a)p_b + 0p_a(1 - p_b) + 1p_a p_b \\
= p_a p_b \]
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:
  \[ P(w \mid \theta_d) = \lambda \frac{tf_{w,d}}{|d|} + (1 - \lambda) \frac{cf_w}{|C|} \]
- Query likelihood model:
  \[ P(Q = q_1 \ldots q_n \mid \theta_d) = \prod_{q \in Q} P(q \mid \theta_d) \]
Inference Network + LM

- Rather than use *tf.idf* estimates for bel(r), use smoothed language modeling estimates:

\[
bel(r) = P(r \mid d_i)
\]

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

- Use Jelinek-Mercer smoothing throughout for simplicity
Combining Evidence

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

\[
P_{\text{wsum}}(I = \text{true}) = \frac{\sum_i w_i p_i}{\sum_i w_i} \\
= \frac{\sum_i w_i (\lambda \frac{t_{f_{q_i,d_j}}}{|d_j|} + (1 - \lambda) \frac{c_{f_{q_i,d_j}}}{|C|})}{\sum_i w_i} \\
\propto \sum_i w_i \frac{t_{f_{q_i,d_j}}}{|d_j|}
\]

\[
P_{\text{wand}}(I = \text{true}) = \prod_i p_i^{w_i} \\
= \prod_i (\lambda \frac{t_{f_{q_i,d_j}}}{|d_j|} + (1 - \lambda) \frac{c_{f_{q_i,d_j}}}{|C|})^{w_i}
\]
Relation to Query Likelihood

- Model subsumes query likelihood model
- Given a query $Q = q_1, q_2, \ldots, q_n$ ($q_i$ is a single term) convert it to the following structured query:
  \[ \text{and}(q_1, q_2, \ldots, q_n) \]
- 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)))
```
<table>
<thead>
<tr>
<th></th>
<th>QL</th>
<th>InQuery</th>
<th>StructLM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rel</td>
<td>6086 6086</td>
<td>6086 6086</td>
<td>6086 6086</td>
</tr>
<tr>
<td>Ret</td>
<td>3190 3371</td>
<td>3306 3679</td>
<td>3355 3737</td>
</tr>
<tr>
<td>0.0</td>
<td>0.6761 0.7156</td>
<td>0.7188 0.7896</td>
<td>0.6888 0.7893</td>
</tr>
<tr>
<td>0.1</td>
<td>0.4796 0.5082</td>
<td>0.4944 0.5601</td>
<td>0.4983 0.5479</td>
</tr>
<tr>
<td>0.2</td>
<td>0.3801 0.4156</td>
<td>0.3942 0.4581</td>
<td>0.3926 0.4486</td>
</tr>
<tr>
<td>0.3</td>
<td>0.3220 0.3465</td>
<td>0.3310 0.3883</td>
<td>0.3300 0.3997</td>
</tr>
<tr>
<td>0.4</td>
<td>0.2672 0.2960</td>
<td>0.2844 0.3317</td>
<td>0.2769 0.3329</td>
</tr>
<tr>
<td>0.5</td>
<td>0.2087 0.2315</td>
<td>0.2241 0.2552</td>
<td>0.2268 0.2631</td>
</tr>
<tr>
<td>0.6</td>
<td>0.1546 0.1708</td>
<td>0.1622 0.1849</td>
<td>0.1713 0.2079</td>
</tr>
<tr>
<td>0.7</td>
<td>0.0903 0.1033</td>
<td>0.0975 0.1236</td>
<td>0.1331 0.1520</td>
</tr>
<tr>
<td>0.8</td>
<td>0.0480 0.0567</td>
<td>0.0544 0.0727</td>
<td>0.0763 0.0894</td>
</tr>
<tr>
<td>0.9</td>
<td>0.0056 0.0175</td>
<td>0.0194 0.0300</td>
<td>0.0246 0.0422</td>
</tr>
<tr>
<td>1.0</td>
<td>0.0021 0.0038</td>
<td>0.0016 0.0014</td>
<td>0.0048 0.0024</td>
</tr>
<tr>
<td>Avg</td>
<td>0.2179 0.2397</td>
<td>0.2312 0.2688</td>
<td>0.2376 0.2779</td>
</tr>
<tr>
<td></td>
<td>5  0.5020 0.5102</td>
<td>0.5265 0.6082</td>
<td>0.5020 0.5796</td>
</tr>
<tr>
<td></td>
<td>10 0.4510 0.4714</td>
<td>0.4735 0.5551</td>
<td>0.4531 0.5490</td>
</tr>
<tr>
<td></td>
<td>15 0.4204 0.4422</td>
<td>0.4190 0.5034</td>
<td>0.4190 0.5007</td>
</tr>
<tr>
<td></td>
<td>20 0.3959 0.4235</td>
<td>0.4071 0.4694</td>
<td>0.3969 0.4602</td>
</tr>
<tr>
<td></td>
<td>30 0.3544 0.3748</td>
<td>0.3578 0.4211</td>
<td>0.3571 0.4156</td>
</tr>
<tr>
<td></td>
<td>100 0.2376 0.2516</td>
<td>0.2412 0.2794</td>
<td>0.2380 0.2843</td>
</tr>
<tr>
<td></td>
<td>200 0.1735 0.1856</td>
<td>0.1771 0.2039</td>
<td>0.1747 0.2064</td>
</tr>
<tr>
<td></td>
<td>500 0.1030 0.1114</td>
<td>0.1071 0.1192</td>
<td>0.1067 0.1227</td>
</tr>
<tr>
<td></td>
<td>1000 0.0651 0.0688</td>
<td>0.0675 0.0751</td>
<td>0.0685 0.0763</td>
</tr>
<tr>
<td>RPr</td>
<td>0.2761 0.2904</td>
<td>0.2763 0.3174</td>
<td>0.2872 0.3282</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>QL</th>
<th>InQuery</th>
<th>StructLM</th>
</tr>
</thead>
<tbody>
<tr>
<td>TREC-6</td>
<td>0.1854</td>
<td>0.1622</td>
<td>0.1863</td>
</tr>
<tr>
<td>TREC-7</td>
<td>0.1972</td>
<td>0.1803</td>
<td>0.2004</td>
</tr>
<tr>
<td>TREC-8</td>
<td>0.2396</td>
<td>0.2343</td>
<td>0.2498</td>
</tr>
</tbody>
</table>
Proximity Node Smoothing

![Graph showing precision against proximity lambda for different TREC datasets (4a, 4m, 6, 7, 8).]
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 bel(r)
- Theoretical considerations