A Markov Random Field Model for Term Dependencies

Donald Metzler and W. Bruce Croft

University of Massachusetts, Amherst Center for Intelligent Information Retrieval
Overview

- Model
  - Markov Random Field model
  - Types of dependence
  - Features

- Training
  - Metric-based training

- Results
  - Newswire data
  - Web data
Past Work

- **Co-occurrence**
  - Boolean queries [Croft et al.]
  - Linked-dependence model [van Rijsbergen]

- **Sequential / Structural**
  - Phrases [Fagan]
  - N-gram language models [Song et al.]

- **Recent work**
  - Dependence language model [Gao et. al.]
  - Query operations [Mishne et. al.]
Motivation

- Terms are less likely to co-occur ‘by chance’ in small collections
- Need to enforce stricter dependencies in larger collections to filter out noise
- Query term order contains great deal of information
  - “white house rose garden”
  - “white rose house garden”
- Want a model that generalizes co-occurrence and phrase dependencies
Broad Overview

Three steps:

- Create dependencies between query terms based on their order
- Define set of term and phrase features over query terms / document pairs
- Train model parameters
Markov Random Field Model

- Undirected graphical model representing joint probability over $Q$, $D$

$$P_{\Lambda}(Q, D) = \frac{1}{Z_{\Lambda}} \prod_{c \in C(G)} \psi(c; \Lambda)$$

$$\psi(c; \Lambda) = \exp[\lambda_c f(c)]$$

- $f(\cdot)$ are feature functions over cliques
- Node types
  - Document node $D$
  - Query term nodes (i.e. $Q = q_1 q_2 \ldots q_N$)
- Rank documents by $P(D \mid Q)$
Types of Dependence

- **Independence (a)**
  - All terms are independent

- **Sequential dependence (b)**
  - Terms independent of all non-adjacent terms given adjacent terms

- **Full dependence (c)**
  - No independence assumptions
Features

- Model allows us to define arbitrary features over the graph cliques
- Aspects we want to capture
  - Term occurrences
  - Query sub-phrases
    - Example: prostate cancer treatment
    - “prostate cancer”, “cancer treatment”
  - Query term proximity
    - Want query terms to occur within some proximity to each other in documents
prostate cancer treatment
Features over cliques containing document and single query term node

How compatible is this term with this document?

\[
f_T(q_i, D) = \log \left[ (1 - \alpha_D) \frac{tf_{q_i,D}}{|D|} + \alpha_D \frac{cf_{q_i}}{|C|} \right]
\]
Term Occurrence Features

- Features over cliques containing document and single query term node
- How compatible is this term with this document?

\[ f_T(q_i, D) = \log \left( (1 - \alpha_D) \frac{tf_{q_i,D}}{|D|} + \alpha_D \frac{cf_{q_i}}{|C|} \right) \]
Term Occurrence Features

- Features over cliques containing document and single query term node
- How compatible is this term with this document?

\[ f_T(q_i, D) = \log \left[ (1 - \alpha_D) \frac{tf_{q_i, D}}{|D|} + \alpha_D \frac{cf_{q_i}}{|C|} \right] \]
**Query Sub-phrase Features**

- Features over cliques containing document and contiguous set of (one or more) query terms
- How compatible is this query sub-phrase with this document?

\[
f_{O}(q_{i}, \ldots, q_{i+k}, D) = \log \left( (1 - \alpha_D) \frac{tf_{\#1(q_{i}, \ldots, q_{i+k}, D)}}{|D|} + \alpha_D \frac{cf_{\#1(q_{i}, \ldots, q_{i+k})}}{|C|} \right)
\]
Query Sub-phrase Features

- Features over cliques containing document and contiguous set of (one or more) query terms
- How compatible is this query sub-phrase with this document?

\[ f_O(q_i, \ldots, q_{i+k}, D) = \log \left( (1 - \alpha_D) \frac{tf_{#1(q_i, \ldots, q_{i+k}), D}}{|D|} + \alpha_D \frac{cf_{#1(q_i, \ldots, q_{i+k})}}{|C|} \right) \]
Query Sub-phrase Features

- Features over cliques containing document and contiguous set of (one or more) query terms
- How compatible is this query sub-phrase with this document?

\[
f_q(q_i, \ldots, q_{i+k}, D) = \log \left( 1 - \alpha_D \right) \frac{tf_{#1(q_i, \ldots, q_{i+k}), D}}{|D|} + \alpha_D \frac{cf_{#1(q_i, \ldots, q_{i+k})}}{|C|}
\]
Query Term Proximity Features

- Features over cliques containing document and any non-empty, non-singleton set of query terms
- How proximally compatible is this set of query terms?

\[ f_U(q_i, \ldots, q_j, D) = \log \left( (1 - \alpha_D) \frac{tf_{\#uwN(q_i, \ldots, q_j), D}}{|D|} + \alpha_D \frac{cf_{\#uwN(q_i, \ldots, q_j)}}{|C|} \right) \]
Query Term Proximity Features

- Features over cliques containing document and any non-empty, non-singleton set of query terms
- How proximally compatible is this set of query terms?

\[ f_U(q_i, ..., q_j, D) = \log \left( 1 - \alpha_D \right) \left( \frac{tf_{\#uwN(q_i, ..., q_j), D}}{|D|} + \alpha_D \frac{cf_{\#uwN(q_i, ..., q_j)}}{|C|} \right) \]
Query Term Proximity Features

- Features over cliques containing document and any non-empty, non-singleton set of query terms
- How proximally compatible is this set of query terms?

\[ f_U (q_i, ..., q_j, D) = \log \left( (1 - \alpha_D) \frac{tf_{\#uwN(q_i, ..., q_j), D}}{|D|} + \alpha_D \frac{cf_{\#uwN(q_i, ..., q_j)}}{|C|} \right) \]
Query Term Proximity Features

- Features over cliques containing document and any non-empty, non-singleton set of query terms
- How proximally compatible is this set of query terms?

\[
f_U(q_i, \ldots, q_j, D) = \log \left[ (1 - \alpha_D) \frac{tf_{\#uwN(q_i, \ldots, q_j), D}}{|D|} + \alpha_D \frac{cf_{\#uwN(q_i, \ldots, q_j)}}{|C|} \right]
\]
Ranking Function

- Infeasible to have a parameter for every distinct feature
- Tie weights for each feature ‘type’
- Final form of (log) ranking function:

\[
\log P_\Lambda(D \mid Q) = \sum_{c \in T} \lambda_T f_T(c) + \sum_{c \in O} \lambda_O f_O(c) + \sum_{c \in O \cup U} \lambda_U f_U(c)
\]

- Need to tune \( \lambda_T, \lambda_O, \lambda_U \)
Likelihood-Based Training

- Maximum likelihood estimate
  - Parameter point estimate that maximizes the likelihood of the model generating the sample

- Downfalls
  - Small sample size
    - TREC relevance judgments
  - Unbalanced data
  - “Metric divergence” [Morgan et. al]

- Need an alternative training method
Metric-Based Training

- Since systems are evaluated using mean average precision, why not directly maximize it?
- Feasible because of the small number of parameters
- Simple hill climbing
Alternate views of the model

- Full independence model, using our features, is exactly LM query likelihood
- Indri structured query

```plaintext
#weight( 0.8 #combine( prostate cancer treatment )
    0.1 #combine( #1( prostate cancer )
                   #1( cancer treatment )
                   #1( prostate cancer treatment )
    0.1 #combine( #uw8( prostate cancer )
                   #uw8( cancer treatment )
                   #uw8( prostate treatment )
                   #uw12( prostate cancer treatment )
                )
```

- Linear combination of features

\[
P_\Lambda(Q, D) \propto \sum_{c \in C(G)} \lambda_c f(c)
\]
Experimental Setup

- Search engine: Indri
- Stemming: Porter
- Stopping: query-time

### Collection Statistics

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Size</th>
<th># Docs</th>
<th>TREC Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>CACM</td>
<td>Communications of the ACM abstracts</td>
<td>1.4MB</td>
<td>3,204</td>
<td>N/A</td>
</tr>
<tr>
<td>WSJ</td>
<td>Wall Street Journal '87-'92</td>
<td>510MB (350x)</td>
<td>173,252 (50x)</td>
<td>50-200</td>
</tr>
<tr>
<td>AP</td>
<td>Associated Press '88-'90</td>
<td>730MB (500x)</td>
<td>242,918 (75x)</td>
<td>50-200</td>
</tr>
<tr>
<td>WT10g</td>
<td>TREC Web Track collection</td>
<td>11GB (8,000x)</td>
<td>1,692,096 (500x)</td>
<td>451-550</td>
</tr>
<tr>
<td>GOV2</td>
<td>TREC Terabyte Track collection</td>
<td>427GB (300,000x)</td>
<td>25,205,179 (8,000x)</td>
<td>701-750</td>
</tr>
</tbody>
</table>
## Sequential Dependence Results

<table>
<thead>
<tr>
<th>Length</th>
<th>AP</th>
<th>WSJ</th>
<th>WT10g</th>
<th>GOV2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.1861</td>
<td>0.2776</td>
<td>0.2148</td>
<td>0.2697</td>
</tr>
<tr>
<td>8</td>
<td>0.1867</td>
<td>0.2763</td>
<td>0.2167</td>
<td>0.2832</td>
</tr>
<tr>
<td>50</td>
<td>0.1858</td>
<td>0.2766</td>
<td>0.2154</td>
<td>0.2817</td>
</tr>
<tr>
<td>Unlimited</td>
<td>0.1857</td>
<td>0.2759</td>
<td>0.2138</td>
<td>0.2714</td>
</tr>
</tbody>
</table>

- Mean average precision results for various unordered window lengths
  - 2 – ‘unordered bigrams’
  - 8 – sentence
  - 50 – passage/paragraph
  - Unlimited – co-occurrence

- Sentence-length windows appear to be good choice
## Full Dependence Results

<table>
<thead>
<tr>
<th>Train \ Test</th>
<th>AP</th>
<th>WSJ</th>
<th>WT10g</th>
<th>GOV2</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP</td>
<td>0.1866</td>
<td>0.2716</td>
<td>0.2226</td>
<td>0.2839</td>
</tr>
<tr>
<td>WSJ</td>
<td>0.1841</td>
<td>0.2738</td>
<td>0.2195</td>
<td>0.2694</td>
</tr>
<tr>
<td>WT10g</td>
<td>0.1865</td>
<td>0.2719</td>
<td>0.2231</td>
<td>0.2783</td>
</tr>
<tr>
<td>GOV2</td>
<td>0.1852</td>
<td>0.2709</td>
<td>0.2201</td>
<td>0.2844</td>
</tr>
</tbody>
</table>

- Mean average precision using term, ordered, and unordered features
- Trained parameters generalize well across collections
# Summary of Results

## Mean Average Precision

<table>
<thead>
<tr>
<th>Collection</th>
<th>$F_I$</th>
<th>$SD$</th>
<th>$FD$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP</td>
<td>0.1775</td>
<td>0.1867* ( +5.2% )</td>
<td>0.1866* ( +5.1% )</td>
</tr>
<tr>
<td>WSJ</td>
<td>0.2592</td>
<td>0.2776† ( +7.1% )</td>
<td>0.2738* ( +5.6% )</td>
</tr>
<tr>
<td>WT10g</td>
<td>0.2032</td>
<td>0.2167* ( +6.6% )</td>
<td>0.2231** ( +9.8% )</td>
</tr>
<tr>
<td>GOV2</td>
<td>0.2502</td>
<td>0.2832* ( +13.2% )</td>
<td>0.2844* ( +13.7% )</td>
</tr>
</tbody>
</table>

## Precision @ 10

<table>
<thead>
<tr>
<th>Collection</th>
<th>$F_I$</th>
<th>$SD$</th>
<th>$FD$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP</td>
<td>0.2912</td>
<td>0.2980 ( +2.3% )</td>
<td>0.3068* ( +5.4% )</td>
</tr>
<tr>
<td>WSJ</td>
<td>0.4327</td>
<td>0.4427 ( +2.3% )</td>
<td>0.4413 ( +2.0% )</td>
</tr>
<tr>
<td>WT10g</td>
<td>0.2866</td>
<td>0.2948 ( +2.9% )</td>
<td>0.3031 ( +5.8% )</td>
</tr>
<tr>
<td>GOV2</td>
<td>0.4837</td>
<td>0.5714* ( +18.1% )</td>
<td>0.5837* ( +20.7% )</td>
</tr>
</tbody>
</table>

* stat. sig. better than $F_I$
** stat. sig. over $SD$ and $F_I$
† stat. sig. over $F_I$, but stat. sig. worse than $FD$
Conclusions

- Model can take into account wide range of dependencies and features over query terms
- Metric-based training may be more appropriate than likelihood or geometric training methods
- Incorporating simple dependencies yields significant improvements
- Past work may have failed to produce good results due to data sparsity
Questions?
References


Relevance Distribution

- Population:
  - Every imaginable user enters every imaginable query and produces a list of documents (from every possible document) they find relevant

- Intuition: the more “votes” a document has for a given query, the more relevant it is on average.
  - Relevance distribution

- Could be applied on a per-user basis
  - User-specific relevance distribution

- Given a large enough sample, we could directly estimate $P( D \mid Q )$
# Metric Divergence Example

Logistic regression model:  
\[ P_\Lambda (R = 1 \mid \mathbf{X} = \mathbf{x}) = \frac{\exp[\Lambda^T \mathbf{x}]}{1 + \exp[\Lambda^T \mathbf{x}]} \]

<table>
<thead>
<tr>
<th>Training Data</th>
<th>Parameters</th>
<th>Ranking</th>
<th>Likelihood / MAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathbf{x}_1 = \begin{bmatrix} 1 \ 1 \end{bmatrix} ) (relevant)</td>
<td>( \Lambda_A = \begin{bmatrix} 0.1 \ 0 \end{bmatrix} )</td>
<td>( \mathbf{x}_1(0.52) )</td>
<td>( P_{\Lambda_A}({\mathbf{x}_i}) = 0.088 )</td>
</tr>
<tr>
<td>( \mathbf{x}_2 = \begin{bmatrix} -5 \ 2 \end{bmatrix} ) (non-relevant)</td>
<td>( \Lambda_B = \begin{bmatrix} 0 \ 0.1 \end{bmatrix} )</td>
<td>( \mathbf{x}_2(0.55) )</td>
<td>( P_{\Lambda_B}({\mathbf{x}_i}) = 0.112 )</td>
</tr>
<tr>
<td>( \mathbf{x}_3 = \begin{bmatrix} -10 \ -1 \end{bmatrix} ) (relevant)</td>
<td></td>
<td>( \mathbf{x}_1(0.52) )</td>
<td>( MAP = 0.83 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \mathbf{x}_3(0.48) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>( MAP = 0.58 )</td>
</tr>
</tbody>
</table>