Learning to Rank under Tight Budget Constraints

Christian Pölitz

MMCI

4th of September, 2012
Introduction and problem description

Framework and ranking model

Costs of parts of a ranking

Use as expected value of parts of a ranking

Optimal ranking with budgets
Problem

Given:
- Document corpus $D$ and index
- Query $q = (t_1, \cdots, t_n)$ of $n$ terms $t_i$
- Budget $B$ and costs $C$

Task:
- Find top $k$ documents for the query (Ranking model)
- Keep budget $B$, (i.e. all costs together are below $B$)
Costs

We assume that the calculation of the ranking costs a certain amount of effort:

- accessing and loading of the elements of an index
- processing the information from the index.

This costs

- access and processing time
- network traffic and energy consumption.

To reflect these efforts, we estimate costs for loading information from an index and processing the information by a ranking model.
Idea

- Load only partial information from the index and process only some parts of the ranking model.
- Estimate importance of parts of the index and ranking model.
- Try to use an optimal combination of parts of the index and ranking model that budget is kept.
- Parameterize the ranking model with respect to what needs to be loaded and what needs to be processed.
Loading

Index (term postings)

\[ L_t = \{\{pos_i\}, id_j | t = d_j[pos_i]\} \]

Additional information about documents length and corpus size

- Separate index list into blocks, that can be loaded independently
- Estimate use and costs of loading a block
Example of blocks

- Blocks sorted by impact (expected use for ranking)
- In the blocks, postings sorted by doc id (compression)
Processing

Ranking model as additive ensemble of base rankers $F_r$

$$\text{score}(q, d) = \sum_r F_r(S(q), d)$$

$$S(q) = \{ q' | q' \subseteq 2^q \}$$

- Combine features to base rankers $F_r$ to estimate the relevance of documents to a given query
- Calculate features from postings from index blocks and additional information about documents and corpus
Parameterization

Parameterized model with $X$ parameter vector

$$score(q, d, X) = \sum_r F_r(S(q, d, X), d) \cdot X_r$$

$$S(q, d, X) = \{ q' | q' \subseteq 2^q \land X_{q'}(d) = 1 \}$$

- $X_q(d) = X_{q,k}$ s.t. $d \in L_{q,k}$
- $X = (X_{t_i,k}, X_{t_i,t_{i+1},k}, X_r)$
Loading costs

\[
\begin{align*}
\text{costs}_l(t_i t_{i+1}, X) &= 0 \\
\text{costs}_l(t_i, X) &= \sum_k \text{costs}_l(t_i, k) \cdot X_{t_i,k} \\
\text{costs}_l(t_i,k) &= |L_{t_i,k}| \cdot k_l
\end{align*}
\]

- Costs for using term \( t_i \) depends on the blocks to be loaded.
- Since bigram \( t_i t_{i+1} \) can only be used when terms \( t_1 \) and \( t_{i+1} \) are already loaded no further costs occur.
- Costs for loading a block depends on its size and a constant.
Processing costs

\[ \text{costs}_p(F_r, X) = (k_p \cdot I(F_r \neq r) + k_r) \cdot \sum_{X_{t_i, t_{i+1}, k=1}} (|L_{t_i,k}| + |L_{t_{i+1},k}|) \cdot X_r \]

\[ + (k_p \cdot I(F_r \neq r) + k_r) \cdot \sum_{X_{t_i, k=1}} |L_{t_i,k}| \cdot X_r \]

- Costs of processing loaded blocks
- How many postings must be processed multiplied by a constant
- Iterate over postings from the blocks only once
Given a query, different posting lists, resp. blocks, and different base rankers have different expected value for the final ranking.

- Some terms or bigrams are more important
- Not all blocks have equal expected value for the ranking
- Value for applying base rankers inherent different and depends on order of application
Expected use of loading

\[ U(t_i, X) = \sum_k \delta_k \cdot X_{t_i,k} \]

\[ + \quad U(t_i t_{i+1}) + U(t_{i-1} t_i) \]

\[ U(t_i t_{i+1}, X) = \sum_{k,k'} (\delta_k + \delta_{k'}) \cdot X_{t_i,k} \cdot X_{t_{i+1},k'} \]

- Assume functional dependency \( \delta_{k+1} = f(\delta_k) \) with decreasing use for additionally load blocks
- Assume additive use when using two terms as bigram
Expected use of processing

\[ U(F_r) = \epsilon_r \cdot \rho_t \]  \hspace{1cm} (4)

- Each base ranker has expected use \( \epsilon_r \)
- Use depends also on the position \( t \) when the ranker is applied
- We expect decreasing use when applying more and more rankers \( \rho_t \)
Optimal parameterization of ranking model

\[ X = \arg\max_{X'} \sum_{q' \in S(q), r} U(q', F_r, X') \cdot X'_r \cdot X'_{q'} \]  

\[ \text{s.t.} \quad \sum_{q' \in S(q)} \text{costs}_l(q', X') + \sum_r \text{costs}_c(F_r, X') \leq B \]

- Knap-sack like approach
- Greedy optimization of benefit: \( \frac{\text{use}}{\text{costs}} \)
- \( U(q', F_r, X) = U(q', X) \cdot U(F_r) \)
Find optimal use parameters

$$\arg\max_{\delta, \epsilon, \rho} \frac{1}{|Q_{tr}|} \cdot \sum_{q \in Q_{tr}} \sum_{B} E(D, score_X(B)(q, .))$$  \hspace{1cm} (6)

- Optimize ranking quality $E$ over the parameters
- Use training data set $Q_{tr}$ with labeled queries
- Linear search over parameter values
Related Approaches

- Cambazoglu et al. WSDM’10
- Wang et al. SIGIR’11
Results on WT10g

**Figure:** NDCG for different budget on the WT10g data set.
Table: Mean NDCG@20 and Precision@20 over all tested budgets. Error notes how many test queries could not actually end before the budget was exceeded. Bold numbers show best results for the data sets. *Shows significant improvements.

<table>
<thead>
<tr>
<th>Data set</th>
<th>.Gov2: Topics 776 to 850</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Error</td>
</tr>
<tr>
<td>Early exit</td>
<td>4%</td>
</tr>
<tr>
<td>Our method</td>
<td>4%</td>
</tr>
</tbody>
</table>
Conclusion

- Estimated use and costs of applying (parts of) a ranking model
- Defined search for optimal loading and application strategy as knap-sack optimization problem
- Learned use of parts of the ranking model by optimizing ranking quality
- Evaluated on a large benchmark collection
Problems

- Too many parameters
- Not directly applicable to more complex ranking models
- Does not work with (gradient) boosting
Thanks for your attention

Questions?