Optimizing Search Engines Using Clickthrough Data

Juozas Gordevičius Tadas Makčinskas

December 5, 2007

Outline

Clickthrough data

- Clickthrough data in search engines can be thought of as triplets (q, r, c).
- Users do not click on links at random, but make a (somewhat) informed choice .
- Even though clickthrough data is typically noisy, the clicks are likely to convey some information.

Collecting clickthrough data

- No overhead for user
- Little overhead for the system
- Easy to collect data

Kind of Information Clickthrough Data Convey

- User is more likely to click on a link, if it is relevant to q
- User is less likely to click on a link low in the ranking, independent of how relevant it is
- It is necessary to consider and model the dependencies of c on q and r appropriately.
  - Click on particular link can't be interpreted as an absolute relevance judgment.
  - User must have observed all $n-1$ links before clicking on link n.
  - Clicked on links gets higher rank then not clicked ones, and keeps the order between themselves as in r.

Algorithm 1. (Extracting Preference Feedback from Clickthrough)

For a ranking $(link_1, link_2, link_3, \ldots)$ *and a set C containing the ranks of the clicked-on links, extract a preference example*

$$link_i <_{r^*} link_j$$

for all pairs $1 <= j < i$, *with* $i \in C$ *and* $j \notin C$.

1. **Kernel Machines**
   http://svm.first.gmd.de/
2. Support Vector Machine
   http://jbolivar.freeservers.com/
3. **SVM-Light Support Vector Machine**
   http://ais.gmd.de/~thorsten/svm_light/
4. An Introduction to Support Vector Machines
   http://www.support-vector.net/
5. Support Vector Machine and Kernel Methods References
   http://svm.research.bell-labs.com/SVMrefs.html
6. Archives of SUPPORT-VECTOR-MACHINES@JISCMAIL..
   http://www.jiscmail.ac.uk/lists/SUPPORT-VECTOR-
7. **Lucent Technologies: SVM demo applet**
   http://svm.research.bell-labs.com/SVT/SVMsvt.html

1. **Kernel Machines**
   http://svm.first.gmd.de/
2. **SVM-Light Support Vector Machine**
   http://ais.gmd.de/~thorsten/svm_light/
3. **Lucent Technologies: SVM demo applet**
   http://svm.research.bell-labs.com/SVT/SVMsvt.html
4. Support Vector Machine
   http://jbolivar.freeservers.com/
5. An Introduction to Support Vector Machines
   http://www.support-vector.net/
6. Support Vector Machine and Kernel Methods References
   http://svm.research.bell-labs.com/SVMrefs.html
7. Archives of SUPPORT-VECTOR-MACHINES@JISCMAIL.
   http://www.jiscmail.ac.uk/lists/SUPPORT-VECTOR-

Outline

Clickthrough data in search engines

A framework for learning of retrieval functions

An SVM Algorithm for Learning of Ranking Functions

Experiments

Discussion and Conclusions

Optimal retrieval function

Problem definition: For query q and document collection $D = \{d_1, \ldots, d_m\}$, optimal retrieval function should return a ranking r^* that ranks the documents in D according to their relevance to the query.

- r^* optimal ordering, $r_{f(q)}$ is ordering retrieved by operational retrieval function f.
- Both r^* and $r_{f(q)}$ are binary relations over $D \times D$.
- $r^* \subset D \times D, r_{f(q)} \subset D \times D$ are asymmetric, negatively transitive matrices.
- $\{r : (d_i, d_j) \in D \times D | d_i <_r d_j\}$ is are strict ordering.

Similarity measure

- Average Precision

$$AvgPrec(r_{sys}, r_{rel}) = \frac{1}{R} \sum_{i=1}^{R} \frac{i}{p_i}$$

- Kendall's τ

$$\tau(r_a, r_b) = \frac{P - Q}{P + Q} = 1 - \frac{2Q}{\binom{m}{2}}$$

- The number of inversions Q gives o lower bound on Average Precision

Kendall's τ example

$$d_1 <_{r_a} d_2 <_{r_a} d_3 <_{r_a} d_4 <_{r_a} d_5$$

$$d_3 <_{r_b} d_2 <_{r_b} d_1 <_{r_b} d_4 <_{r_b} d_5$$

Concordant pairs $P = 7$:
$(d_1, d_4), (d_1, d_5), (d_2, d_4), (d_2, d_5), (d_3, d_4), (d_3, d_5), (d_4, d_5)$.
Discordant pairs $Q = 3$:
$(d_2, d_3), (d_1, d_2), (d_1, d_3)$.

$$\tau(r_a, r_b) = \frac{P - Q}{P + Q} = \frac{7 - 3}{7 + 3} = 0,4$$

Outline

Clickthrough data in search engines

A framework for learning of retrieval functions

An SVM Algorithm for Learning of Ranking Functions

Experiments

Discussion and Conclusions

An SVM Algorithm for Learning of Ranking Functions

- Training sample S of size n
  - Independently and identically distributed
  - Contains queries with their target rankings

$$(q_1, r_1^*), \ldots, (q_n, r_n^*)$$

- Learner L
  - Selects a ranking function $f \in F$ that maximizes the average τ.

$$\tau_S(f) = \frac{1}{n} \sum_{i=1}^{n} \tau(r_{f(q_i)}, r_i^*).$$

Linear Ranking Functions

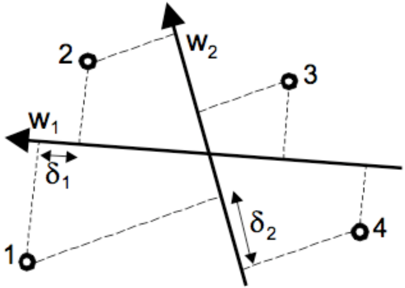
- Given a class of linear ranking functions

$$(d_i, d_j) \in f_{\overrightarrow{w}}(q) \iff \overrightarrow{w}\Phi(q, d_i) > \overrightarrow{w}\Phi(q, d_j)$$

  - $\overrightarrow{w}$ is a weight vector adjusted by learning
  - $\Phi(q, d_i)$ describe the match between the query q and document d_i.
    - e.g. the number of words that query and document share.
- Just need to find the weight vector that maximizes the average τ.

The Weight Vector

- For any vector $\vec{w}$, the points are ordered by their projection onto $\vec{w}$.
- For each query we seek a vector that orders documents correctly.
- For the whole dataset we seek one vector that minimizes the number of discordant pairs.



Ranking SVM

- Finding weight vector $\overrightarrow{w}$ is NP-hard.
- The solution can be approximated like in classification SVMs.
- The learned retrieval function $f_{\overrightarrow{w}^*}$ is a linear combination of feature vectors.
- $f_{\overrightarrow{w}^*}$ will be used for ranking the set of documents according to a new query.

Using Partial Feedback

- Clickthrough logs are the source of training data
- Target ranking r^* is not known
- A subset $r' \subseteq r^*$ can be inferred from the log.
- Thus the training set is

$$(q_1, r'_1), \ldots, (q_n, r'_n)$$

- And the retrieval function is determined based on the partial feedback.

Outline

"Striver" a Meta-search Engine

- Meta-search combines results of several basic search engines
  - Easy to implement
  - Covers large document collection
  - Basic search engines provide basis for comparison
- "Striver"
  - Forward user query to Google and others
  - Extract top 100 from each search result
  - Rank the union of all documents according to learned function
  - Return top 50.

Blind Statistical Test

- How to compare the quality of different retrieval functions?
  - Present two rankings at the same time
- For two rankings A and B produce the combined one C, s.t.
  - for any l
  - The top l links of C contain top k_A and k_B of A and B respectively
  - $|k_A - k_B| \leq 1$
  - Such combined ranking always exists.

Example

- User clicked on links 1, 3, 7
- Therefore he saw the top 4 from each ranking
- All 3 clicked links were within top 4 of ranking A
- Only 1 clicked link was in ranking B

$\Rightarrow$ Ranking A is significantly better.

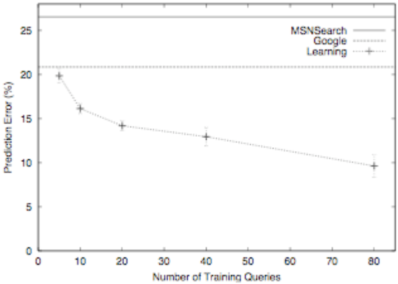
Offline Experiment

Does the Ranking SVM learn regularities using partial feedback from clickthrough data?

- Recorded 112 queries with non-empty set of clicks from "Striver"
- Constructed the feature mapping $\Phi(q, d)$ to learn the retrieval function. e.g.:
  - top1_X - ranked #1 in Google, MSN or any other
  - query_url_cosine - cosine between URL words and query
  - url_contains_tilde ...
- Additional 50 constraints to stabilize the result

Offline Experiment

- x - number of training queries
- y - percentage of pairwise preference constraints that are not fullfiles



Interactive Online Experiment

- "Striver" made available for a group of 20
- Ranking SVM applied on collected 260 queries
- The learned function was implemented in "Striver" and used subsequently
- "Striver" vs Google
  - For 29 queries the learned function was prefered
  - For 13 queries Google result prevailed
  - For 27+19 queries equal number or no links clicked
  - Therefore, the learned retrieval function is better than the one of Google with 95% confidence.

The Learned Function

weight	feature
0.60	query_abstract_cosine
0.48	top10_google
0.24	query_url_cosine
0.24	top1count_1
0.24	top10_msnsearch
0.22	host_citeseer
0.21	domain_nec
0.19	top10count_3
0.17	top1_google
0.17	country_de
...	
0.16	abstract_contains_home
0.16	top1_hotbot
...	
0.14	domain_name_in_query
...	
-0.13	domain_tu-bs
-0.15	country_fi
-0.16	top50count_4
-0.17	url_length
-0.32	top10count_0
-0.38	top1count_0

Table 3: Features with largest and smallest weights as learned from the training data in the online experiment.

Outline

Ranking SVM is Good

- Successfully learned retrieval function from clickthrough data
- Automatically adapted to the particular preferences of a group of about 20
- No manual parameter tuning

Therefore, ML techniques improve the retrieval by tailoring the retrieval function to small homogenous groups.

New Questions

- What is a good size of a user group and how can those be determined?
- Can we use clickthrough data to tailor search of particular topics?
- Is there an incremental online algorithm for learning?
- How sensitive is the approach to spamming?

Ranking SVM in Recommender Systems

- The approach is not limited to meta-search engines
- Observing the channel surfing behaviour one could infer user's favorite programmes.

End of presentation.