Optimizing Search Engines using Clickthrough Data

By Thorsten Joachims

R222 Presentation by Kaitlin Cunningham
23 January 2017

What is clickthrough data?

- Triplet: (q, r, c)

- Premise: set *c* conveys some information about user preferences

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.AC.UK
   http://www.jiscmail.ac.uk/lists/SUPPORT-VECTOR-MACHINES.html
7. **Lucent Technologies: SVM demo applet**
   http://svm.research.bell-labs.com/SVT/SVMsvt.html
8. Royal Holloway Support Vector Machine
   http://svm.dcs.rhbnc.ac.uk/
9. Support Vector Machine - The Software
   http://www.support-vector.net/software.html
10. Lagrangian Support Vector Machine Home Page
    http://www.cs.wisc.edu/dmi/lsvm

Pairwise preferences

$$link_3 <_{r^*} link_2 \qquad \begin{array}{l} link_7 <_{r^*} link_2 \\ link_7 <_{r^*} link_4 \\ link_7 <_{r^*} link_5 \\ link_7 <_{r^*} link_6 \end{array} \qquad (1)$$

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 $1 \leq j < i$*, with* $i \in C$ *and* $j \notin C$.

A new learning algorithm

- Optimal (target) ranking r* v. system ranking $r_{f(q)}$

- Kendall's τ:

$$\tau(\mathrm{r}_a, \mathrm{r}_b) = \frac{P - Q}{P + Q} = 1 - \frac{2Q}{\binom{m}{2}} \qquad (2)$$

- Expected Kendall's τ:

$$\tau_P(\mathrm{f}) = \int \tau(\mathrm{r}_{\mathrm{f(q)}}, \mathrm{r}^*) d\,\Pr(\mathrm{q}, \mathrm{r}^*) \qquad (6)$$

SVM algorithm for learning

- Empirical risk minimization approach:

$$\tau_S(\mathrm{f}) = \frac{1}{n}\sum_{i=1}^{n} \tau(\mathrm{r}_{\mathrm{f}(\mathrm{q}_i)}, \mathrm{r}_i^*). \tag{8}$$

$$\forall(\mathrm{d}_i, \mathrm{d}_j) \in \mathrm{r}_1^* : \quad \vec{w}\Phi(\mathrm{q}_1, \mathrm{d}_i) > \vec{w}\Phi(\mathrm{q}_1, \mathrm{d}_j) \tag{10}$$

$$\ldots$$

$$\forall(\mathrm{d}_i, \mathrm{d}_j) \in \mathrm{r}_n^* : \quad \vec{w}\Phi(\mathrm{q}_n, \mathrm{d}_i) > \vec{w}\Phi(\mathrm{q}_n, \mathrm{d}_j) \tag{11}$$

Optimization Problem 1. (Ranking SVM)

minimize:

$$V(\vec{w}, \vec{\xi}) = \frac{1}{2}\, \vec{w} \cdot \vec{w} + C \sum \xi_{i,j,k} \tag{12}$$

subject to:

$$\begin{aligned} &\forall(d_i, d_j) \in r_1^* : \vec{w}\Phi(q_1, d_i) \geq \vec{w}\Phi(q_1, d_j) + 1 - \xi_{i,j,1} \\ &\qquad\qquad \ldots \\ &\forall(d_i, d_j) \in r_n^* : \vec{w}\Phi(q_n, d_i) \geq \vec{w}\Phi(q_n, d_j) + 1 - \xi_{i,j,n} \end{aligned} \tag{13}$$

$$\forall i \forall j \forall k : \xi_{i,j,k} \geq 0 \tag{14}$$

Using partial feedback

- Replace r* with r':

OPTIMIZATION PROBLEM 2. (RANKING SVM (PARTIAL))

$$\textit{minimize:} \qquad V(\vec{w}, \vec{\xi}) = \frac{1}{2}\,\vec{w}\cdot\vec{w} + C\sum \xi_{i,j,k} \tag{21}$$

subject to:

$$\begin{aligned} &\forall (d_i, d_j) \in r'_1 : \vec{w}\Phi(q_1, d_i) > \vec{w}\Phi(q_1, d_j) + 1 - \xi_{i,j,1} \\ &\qquad\qquad\qquad \ldots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (22) \\ &\forall (d_i, d_j) \in r'_n : \vec{w}\Phi(q_n, d_i) > \vec{w}\Phi(q_n, d_j) + 1 - \xi_{i,j,n} \end{aligned}$$

$$\forall i \forall j \forall k : \xi_{i,j,k} \geq 0 \tag{23}$$

Experiment: Offline

- Training set: 112 queries over one month to Google and MSNSearch through “Striver”
- Feature mapping $\Phi(\mathrm{q}, \mathrm{d})$:
  - 38 rank-based features
  - 3 query/content features
  - ~20 000 popularity attribute features
- Extracted pairwise preferences using Alg 1
- 50 constraints added

Offline: Results

- Test error decreases to ~10% with 80 training queries (out of 112)

✓Proof of concept

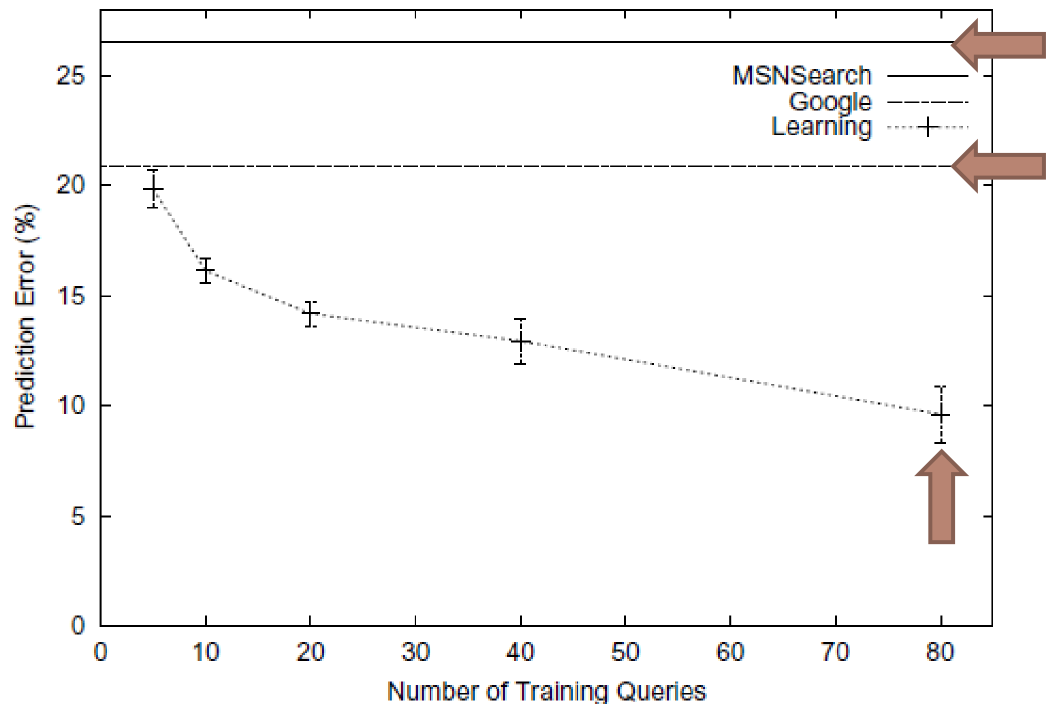

Figure 4: Generalization error of the Ranking SVM depending on the size of the training set. The error bars show one standard error.

Experiment: Interactive Online

- Training set: 260 queries from 20 users over less than a month
- Evaluation period of ~2 weeks
- Compared against Google, MSNSearch and Toprank

Interactive Online: Results

Comparison	more clicks on learned	less clicks on learned	tie (with clicks)	no clicks	total
Learned vs. Google	29	13	27	19	88
Learned vs. MSNSearch	18	4	7	11	40
Learned vs. Toprank	21	9	11	11	52

Table 2: Pairwise comparison of the learned retrieval function with Google, MSNSearch, and the non-learning meta-search ranking. The counts indicate for how many queries a user clicked on more links from the top of the ranking returned by the respective retrieval function.

- Users clicked on more links from the learned retrieval function than the other search engines

✓ Learned function improves retrieval

Discussion

- Personalised retrieval functions which can be tailored to small homogenous groups or individual users
- Function doesn’t rely on explicit relevance judgements
- Question: What are the computational demands of training using clickthrough data?

Critique

- Theory well-placed in context of other measures and research

- Well-reasoned explanations throughout

Critique

- Little to no discussion about the constraints
- No discussion about the relevance/influence of the tied clicks or no clicks in the online experiment
- Experiments based on homogenous user base:
  - How diverse were the queries in the training and testing periods
- Hypothesise the effect of scaling up the number of queries