Data Mining

Optimizing Search Engines using Clickthrough Data

(Thorsten Joachims)

Dimitrios Milios
Anastasios Polymeros

Data Mining in the service of Information Retrieval

Goal: Optimize retrieval quality of Search Engines

Exploit user preferences as recorded in the logfiles of search engines

Train a Ranking SVM algorithm

Data Mining in the service of Information Retrieval

- Training data can be generated by relevance judgement by experts
- Difficult and expensive procedure
- Instead, use logs of links that the users clicked on
- Such data is available in abundance, at very low cost

Clickthrough Data

- Triplets (**q**, **r**, **c**)
  - **q**: query
  - **r**: ranking presented to the user
  - **c**: set of links that the user clicked on
- Can be recorded with little overhead

Recording Clickthrough Data

- Clicks recorded in a proxy-system's log file
- A unique ID is assigned to each query
- Links on the results page point to the proxy-server
- The proxy-server records the clicked URL and query ID
- Finally, the proxy forwards the user to the target URL
- The whole process is transparent to the user

Information that can be elicited

- Ranking **r** is dependent on query **q**
- Set of links **c** is dependent on
  - **q**: it depends on the relevance to the query
  - **r**: it is unlikely to click on a link low in the ranking, no matter its relevance
- The users click on the relatively most promising links, among the top ones, independent on their absolute relevance

Information that can be elicited: Example

1. **Kernel Machines**
   $http://svm.first.gmd.de/$
2. Support Vector Machine
   $http://jbolivar.freeservers.com/$
3. **SVM-Light Support Vector Machine**
   $http://ais.gmd.de/\sim thorsten/svm\_light/$
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
   $http://www.jiscmail.ac.uk/lists/SUPPORT\text{–}VECTOR\text{–}MACHINES.html$
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
    $http://www.cs.wisc.edu/dmi/lsvm$

Information that can be elicited: Example

- Links 1, 3, 7 are relevant on an absolute scale
- Link 3 is more relevant than link 2
  - $link_3 <_{r*} link_2$
- Link 7 is more relevant than 2, 4, 5, 6
  - $link_7 <_{r*} link_2$
  - $link_7 <_{r*} link_4$
  - $link_7 <_{r*} link_5$
  - $link_7 <_{r*} link_6$

Information that can be elicited

- Clickthrough data does not convey absolute relevance judgements
- Instead, *partial relative relevance* judgements are conveyed for the links the user browsed through.
  - *relative*: some link are better than others
  - *partial*: there is no information for all of the links

Extracting Preference Feedback from Clickthrough data

- For ranking ($link_1$, $link_2$, $link_3$, ...) and a set C containing the ranks of clicked-on links, extract a preference example:
  - $link_i <_{r*} link_j$
  - $for\ all\ pairs\ 1 \leq j < i,\ with\ i \in C$ and $j \notin C$

The function to be optimized

- Optimum ordering $r* \subset D \times D$
  - Documents **D** are ordered according to their relevance to the query
- Ordering $r_{f(q)} \subset D \times D$
  - given by a function **f**, for a query **q**
- Maximization of similarity between

  r* and $\mathbf{r}_{f(q)}$

Definition of similarity

- Assume orderings
  - $r_a \subset D \times D$
  - $r_b \subset D \times D$

- $(d_i, d_j) \in D \times D, d_i \neq d_j$
- Concordant pairs **P**: if both $\mathbf{r}_a$ and $\mathbf{r}_b$ agree in how they order $\mathbf{d}_i$, $\mathbf{d}_j$
- Discordant pairs **Q**: if $\mathbf{r}_a$ and $\mathbf{r}_b$ disagree in how they order $\mathbf{d}_i$, $\mathbf{d}_j$

Definition of similarity

- Kentall's τ:
  - $$\tau(r_a, r_b) = \frac{P-Q}{P+Q} = 1 - \frac{2Q}{\binom{m}{2}}$$
  - where **m**, the number of documents in the collection **D**

The function to be optimized

- Learn a ranking function **f** so as to maximize:
  - $$\tau_P(f) = \int \tau(r_{f(q)}, r^*)\, d\Pr(q, r^*)$$
  - for a fixed but unknown distribution Pr(q, r*) of queries and target rankings
  - training set is a sample of Pr(q, r*)

Ranking SVM Algorithm

- **Φ(q, d)** is a mapping onto features that describe the match between query **q** and document **d**
- Consider the class of linear ranking functions

$$(d_i, d_j) \in f_{\vec{w}}(q) \Leftrightarrow \vec{w}\,\Phi(q, d_i) > \vec{w}\,\Phi(q, d_j)$$

  - For any weight vector **w**, the documents are ordered by the projection onto **w**

Ranking SVM Algorithm

- Instead of directly maximizing

$$\tau_P(f) = \int \tau(r_{f(q)}, r^*)\, d\,Pr(q, r^*)$$

- Minimize discordant pairs **Q**
- Find the weight vector so that the maximum number of the following inequalities is fulfilled

$$\forall (d_i, d_j) \in r_k^* \Leftrightarrow \vec{w}\Phi(q_k, d_i) > \vec{w}\Phi(q_k, d_j)$$

  - where $1 \leq k \leq n$

Ranking SVM Algorithm

- NP-hard optimization problem
- Introduce slack variables $\boldsymbol{\xi}_{i,j,k}$
- Minimize the upper bound $\sum \xi_{i,j,k}$
- $V(\vec{w}, \vec{\xi}) = \frac{1}{2}\vec{w}\cdot\vec{w} + C\sum \xi_{i,j,k}$
  - Subject to:

$$\forall (d_i, d_j) \in r_k^* \Leftrightarrow \vec{w}\,\Phi(q_k, d_i) \geq \vec{w}\,\Phi(q_k, d_j) + 1 - \xi_{i,j,k}$$

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

  - **C** allows trading-off margin size against training error

Relation to Classification SVM Algorithm

- The inequalities in the previous slide can be rearranged as well:
  - $\vec{w}(\Phi(q, d_i) - \Phi(q, d_j)) \geq 1 - \xi_{i,j,k}$
- The optimization problem is equivalent to that of classification SVM on pairwise difference vectors $\Phi(q, d_i) - \Phi(q, d_j)$
- *SVM*^{light} is used for training

Ranking SVM Algorithm

- Learned retrieval function $\mathbf{f}_{w^*}$ can be shown as linear combination of feature vectors
  - $(d_i, d_j) \in f_{\vec{w}*}(q)$

    $$\Leftrightarrow \vec{w}* \ \Phi(q, d_i) > \vec{w}* \ \Phi(q, d_j)$$
  - where: $\vec{w}* = \sum a_{k,l} * \ \Phi(q_k, d_l)$
- Kernels could be used, and extend the algorithm to non-linear functions

Ranking SVM Algorithm

- To produce a ranking using $\mathbf{f}_{w^*}$, according to a new query **q**:
  - sort the documents by their value of:

$$rsv(q, d_i) = \sum a_{k,l} * \Phi(q_k, d_l) \Phi(q, d_i)$$

Experiment setup

- *Striver* Meta-Search Engine
  - *Google*
  - *MSNSearch*
  - *Excite*
  - *Altavista*
  - *Hotbot*
- Striver ranks the union of the results according to the learned $\mathbf{f}_{w^*}$

Experiment setup

- In order to compare two rankings **A** and **B**
  - Combine into a single ranking **C**
  - **C** contains the top $\mathbf{k_a}$ links from **A**, and the top $\mathbf{k_b}$ links from **B**, where $|k_a - k_b| \leq 1$
  - The user should not be able to tell which retrieval method proposed each link
  - Assume that the user probably clicks on the most relevant links
- If the user clicks on significantly more links from **A** than from **B**, then **A** must contain more relevant links

Combination into single ranking - Example

Ranking A:

1. Kernel Machines
   http://svm.first.gmd.de/
2. SVM-Light Support Vector Machine
   http://ais.gmd.de/ ~ thorsten/svm_light/
3. Support Vector Machine and Kernel ... References
   http://svm.....com/SVMrefs.html
4. Lucent Technologies: SVM demo applet
   http://svm.....com/SVT/SVMsvt.html
5. Royal Holloway Support Vector Machine
   http://svm.dcs.rhbnc.ac.uk/
6. Support Vector Machine - The Software
   http://www.support–vector.net/software.html
7. Support Vector Machine - Tutorial
   http://www.support–vector.net/tutorial.html
8. Support Vector Machine
   http://jbolivar.freeservers.com/

Ranking B:

1. Kernel Machines
   http://svm.first.gmd.de/
2. Support Vector Machine
   http://jbolivar.freeservers.com/
3. An Introduction to Support Vector Machines
   http://www.support – vector.net/
4. Archives of SUPPORT-VECTOR-MACHINES ...
   http://www.jiscmail.ac.uk/lists/SUPPORT...
5. SVM-Light Support Vector Machine
   http://ais.gmd.de/ ~ thorsten/svm_light/
6. Support Vector Machine - The Software
   http://www.support–vector.net/software.html
7. Lagrangian Support Vector Machine Home Page
   http://www.cs.wisc.edu/dmi/lsvm
8. A Support ... - Bennett, Blue (ResearchIndex)
   http://citeseer.../bennett97support.html

Combined Results:

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

Offline experiment

- Compare Against
  - Google, MSNSearch
- 112 queries with non-empty sets of clicks
- Design a feature mapping Φ(q, d)
  - The set of features is not optimal

Offline experiment – Feature Mapping

1. Rank in other search engines (38 features total):

rank_X: 100 minus rank in X ∈ {Google, MSN-Search, Altavista, Hotbot, Excite} divided by 100 (minimum 0)
top1_X: ranked #1 in X ∈ {Google, MSNSearch, Altavista, Hotbot, Excite} (binary {0, 1})
top10_X: ranked in top 10 in X ∈ {Google, MSNSearch, Altavista, Hotbot, Excite} (binary {0, 1})
top50_X: ranked in top 50 in X ∈ {Google, MSNSearch, Altavista, Hotbot, Excite} (binary {0, 1})
top1count_X: ranked #1 in X of the 5 search engines
top10count_X: ranked in top 10 in X of the 5 search engines
top50count_X: ranked in top 50 in X of the 5 search engines

2. Query/Content Match (3 features total):

query_url_cosine: cosine between URL-words and query (range [0, 1])
query_abstract_cosine: cosine between title-words and query (range [0, 1])
domain_name_in_query: query contains domain-name from URL (binary {0, 1})

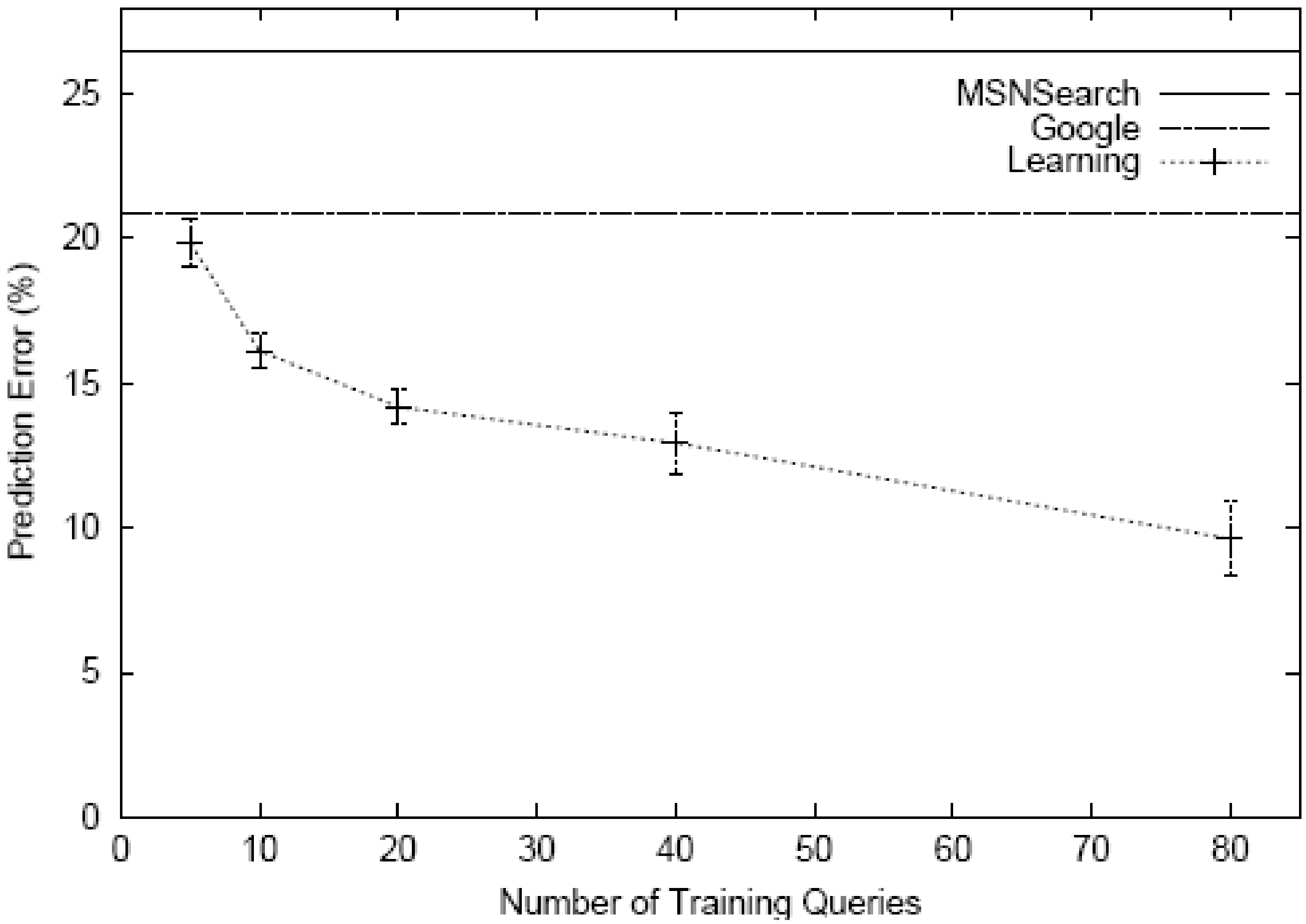
3. Popularity-Attributes (∼ 20.000 features total):

url_length: length of URL in characters divided by 30
country_X: country code X of URL (binary attribute {0, 1} for each country code)

Offline experiment

- Split data into a training and a test set
- Test ranking SVM for a different number of training queries
  - The more the queries, the better the performance
- Trade-off between training error and margin was selected from $C \in \{0.001, 0.003, 0.005, 0.01\}$
  - Minimizing leave-one-out error on the training set

Offline experiment



Online experiment

- Striver was made available to a group of 20 users
- Compare Striver's learned function $\mathbf{f}_{w^*}$ against:
  - Google, MSNSearch, Toprank
- The comparison is based on the number of links clicked from each one of the strategies

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

Analysis of the Learnt Function

- High positive weights indicate that documents with these features should be higher in ranking
- High negative weights indicate that documents with these features should be lower in ranking
- Most training queries were for scientific material, which is reflected in the weighting
  - e.g. URLs for domain “citeseer” received positive weight

Analysis of the Learnt 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

Conclusions

- Data mining logfiles of WWW search engines
- It is verified that Ranking SVM can learn an improved retrieval function from clickthrough data
- Adapting the retrieval function to the preferences of a group of users

Data Mining

Thank You

Dimitrios Milios
Anastasios Polymeros