Learning to Rank and Data Labeling from Clickthrough Data

ABSTRACT
Learning to rank is the application of machine learning algorithms in solving ranking problems through model training. It has many applications in information retrieval, natural language processing and data mining.

LEARNING TO RANK
In document retrieval, given a query, search engine will retrieve a series of relevant documents (through term/keyword/semantic mapping), rank these retrieved documents, and output the top N results. To solve the ranking problem, we use a ranking model, which can either be heuristically designed or trained from machine learning algorithms. The second approach is becoming more and more popular, especially in Web Search. This is because in Web Search, lots of information can be used to identify the relevance between a query-document pair. On the other hand, due to the existence of large amount of search logs, using users' clickthrough logs as training data makes it possible for obtaining a ranking model from machine learning.

Learning to rank is supervised learning, and is composed of training stage and testing stage, as shown in fig. 2

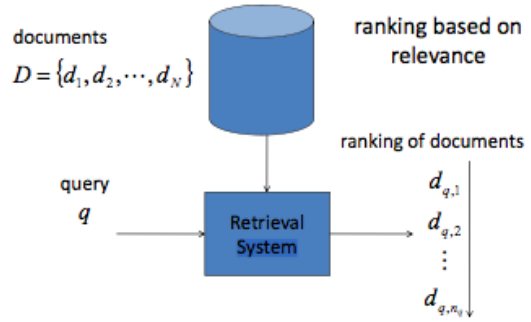

Fig. 1 Document Retrieval

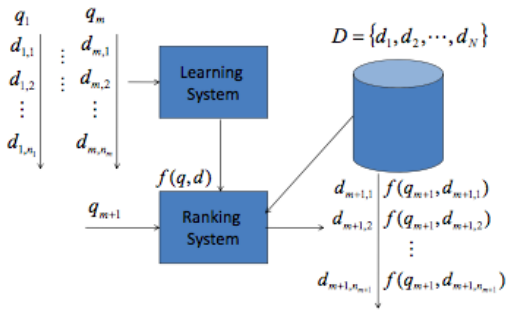

Fig. 2 Learning to Rank for Document Retrieval

TRAINING DATA
Search logs have recorded users' daily searching and clicking actions in real life. Clicking action implicitly implies the relevance of each query-document pair, and can be used to determine the level of relevance of query-document pair. A simplest way is to use the number of clicks on different documents as their relevance level to the same query.

One thing to notice is that user clicking actions are position biased, which means that users tend to click on documents that are presented higher in the list, even though the document is not relevant or does not have a high relevance level. There are many methods that can be used to remove this position bias:

1. When document ranked lower has more clicks than document ranked higher, the lower-ranked document has a higher relevance than the higher-ranked document
2. Joachims has come up with a few ways to determine relevance level without bias: Click > Skip Above, Last Click > Skip Above, Click > Earlier Click, Click > Skip Previous, Click > No Click Next, etc.
3. Previously, we mentioned that a higher clicked number does not necessarily mean higher relevance. However, if the gap becomes large enough: when a document has 5-10 times of clicks than another document, we believe that the more clicked documents

has higher relevance level than the less clicked document. But this judgement depends on the context.

4. Position bias exists because in the search action, user is unable to see all the results, but only the top ranked results. Click Model can make use of a user's clickthrough data to determine the list of documents that the user actually viewed and the list of documents that the user did not view. From this information, we will be able to determine the relative relevance level of documents from more accurate clickthrough-rates.

Even though explicit judgement is more reliable than implicit judgement, the real users are usually unwilling to give the relevance feedbacks. It is usually done by another group of people who did not conduct the actual search using the query. This introduces inaccuracy and makes it difficult to know the real intention of the search, and is costly and time-consuming at the same time. On the other hand, implicit judgement which is retrieved from search logs is affected by noisy clicking actions, especially in long-tail query. Furthermore, queries with clickthrough data is a subset of all queries, and it is impossible to get the labels of documents for all queries.

FORMULATION

To represent Learning to Rank algorithm, the loss function is $L(F(x), y)$, and risk function is:

$$R(F) = \int_{x \times y} L(F(x), y) dP(x, y)$$

With the training data, we are able to get the empirical risk function:

$$\mathcal{R}(F) = \frac{1}{m} \sum_{i=1}^{m} L(F(x_i), y_i)$$

Thus, the learning problem becomes how to minimize this empirical risk function. This optimization problem is difficult to solve because loss function is not continuous. We can use another surrogate function which is easier to solve to replace the original loss function, and optimize the surrogate function:

$$\mathcal{R}'(F) = \frac{1}{m} \sum_{i=1}^{m} L'(F(x_i), y_i)$$

There are many different choices of surrogate functions:

1. Pointwise loss: e.g. squared loss

$$L'(F(x), y) = \sum_{i=1}^{n} (f(x_i) - y_i)^2$$

2. Pairwise loss: e.g. hinge loss, exponential loss, logistic loss

$$L'(F(x), y) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \emptyset(sign(y_i - y_j), f(x_i) - f(x_j))$$

3. Listwise loss:

$$L'(F(x), y) = \exp(-NDCG)$$

CONCLUSION

Learning to Rank methods can be categorized into 3 types: pointwise, pairwise and listwise. Pointwise and pairwise methods transform ranking problem to problem of classification, regression, and ordinal classification, etc. The advantage is to utilize existing classification and regression algorithms, but the disadvantage is that group structure is being ignored: relative rank of all the documents for each query is not considered. This makes the learning target not

the same as the evaluation target. On the other hand, listwise method trans a ranking list as one instance, and considers the relative rank of all the documents for each query. Below are different algorithms for each of the 3 approaches:

1. Pointwise: Subset Ranking, McRank, Prank, OC SVM
2. Pairwise: Ranking SVM, RankBoost, RankNet, GBRank, IR SVM, Lambda Rank, LambdaMart
3. Listwise: ListNet, ListMLE, AdaRank, SVM MAP, Soft Rank