Future directions in learning to rank

Olivier Chapelle CHAP@YAHOO-INC.COM
Yi Chang YICHANG@YAHOO-INC.COM
Yahoo! Labs
Sunnyvale, CA

Tie-Yan Liu TIE-YAN.LIU@MICROSOFT.COM
Microsoft Research
Beijing, China

Abstract

The results of the learning to rank challenge showed that the quality of the predictions from the top competitors are very close from each other. This raises a question: is learning to rank a solved problem? On the on hand, it is likely that only small incremental progress can be made in the "core" and traditional problematics of learning to rank. The challenge was set in this standard learning to rank scenario: optimize a ranking measure on a test set. But on the other hand, there are a lot of related questions and settings in learning to rank that have not been yet fully explored. We review some of them in this paper and hope that researchers interested in learning to rank will try to answer these challenging and exciting research questions.

1. Learning Theory for Ranking

Many learning to rank algorithms have been shown effective through benchmark experiments. However, sometimes benchmark experiments are not as reliable as expected due to the small scales of the training and test data. In this situation, a theory is needed to guarantee the performance of an algorithm on infinite unseen data.

Statistical learning theory, specifically the generalization theory, investigates the bound between the risk on the finite training data and the risk on infinite test data. In learning to rank, the risk on the training data is defined with a surrogate loss function (e.g., the pairwise losses in Ranking SVM (Joachims, 2002) and RankBoost (Freund et al., 2003)), while the risk on the test set is measured by a ranking measure (e.g., 1-NDCG or 1-MAP). Therefore, to obtain a generalization bound in this setting, we need to address the following issues: (i) a reasonable assumption on the data generation (e.g., queries and documents), (ii) a generalization bound regarding the surrogate loss function; (iii) the relationship between the surrogate loss function and the ranking measure; (iv) and the existence of the limit of the ranking measure when the number of documents approaches infinity.

As for these issues, there have been a number of attempts but still a large open space to explore.

- *Assumption on data generation.* In (Agarwal and Niyogi, 2005; Clemencon and Vayatis, 2007) it is assumed that the documents in the training data are sampled in an i.i.d. manner, no matter which queries they are associated with. However, it is

widely accepted that documents associated with different queries should have different distributions. In (Lan et al., 2008; Lan and Liu, 2009), it is assumed that queries are sampled in an i.i.d. manner, while documents associated with each query are generated in a deterministic manner. However, it is clear that there is sampling of documents when one constructs the training set. Recently, in (Chen et al., 2010b), a new assumption is adopted, which involves sampling at both query and document layers. This assumption seems to better describe the data generation for learning to rank. However, its problem lies in that the resultant data is non-i.i.d. and the theoretical analysis becomes difficult. It is desirable to find a reasonable yet easy way of describing the data generation, which will help the corresponding theoretical analysis a lot.

- *Generalization bound regarding surrogate loss functions.* There have been a number of works discussing this problem, however, most of them cannot be used to explain real learning to rank algorithms, mainly due to the unreasonable data assumptions they used (see the discussions in the previous item). In (Chen et al., 2010b), a two-layer generalization bound is derived for the pairwise surrogate loss functions given the assumption of two-layer sampling. However, it seems that the results cannot be easily extended to other types of loss functions (e.g., pointwise and listwise loss functions).

- *Relationship between surrogate loss functions and ranking measures.* There have been some attempts on this in (Cossock and Zhang, 2006) and (Chen et al., 2010a), which show that several pointwise, pairwise, and listwise surrogate loss functions are upper bounds of the widely-used ranking measures. However, there are still many other surrogate loss functions not covered. Furthermore, the bounds obtained in these works are the worst-case bounds, and thus very loose. Especially when the number of documents increases, the bounds will become almost meaningless because the weight before the surrogate loss functions will eventually approach infinity. It is highly desirable to obtain tighter bounds, probably in the sense of expectation.

- *Existence of limits for ranking measures.* There is no result yet on whether the limits exist when the number of documents approaches infinity. The challenge lies in that the ranking measures do not have a sum-i.i.d. form with respect to documents and the law of large numbers cannot be used to prove their convergence and limits. However, if we cannot guarantee the existence of their limits, then we even do not know whether the risk on the infinite test data makes sense. We need to understand the convergence of existing ranking measures and/or to design new measures that can converge with respect to the increasing number of documents.

If the above four issues are solved, we will be able to move a significant step forward towards the generalization ability for learning to rank. However, this is not yet the end of the story. Usually the uniform generalization bound is too loose to guide real applications. In order to derive a tighter bound, we may need some additional assumptions on the data, instead of just a weak i.i.d. one. This is in the light of PAC-Bayes, and should be another important future research direction to explore.

2. Online Complexity v.s. Accuracy

Most of the research on learning to rank has focused on optimizing the relevance of search results. This quest for accuracy can lead to very complex models. Indeed, as datasets grow, learning theory suggests that the model complexity (in terms of VC dimension) should also grow. And complex models are often computationally expensive. For instance, the model of Eric Gottschalk and David Vogel – the runners-up in the first track of the challenge – has more than 10,000 trees some of the trees have more than 1,000 leaves. The brings the question on wether this kind of models can be deployed in a real world ranking system where latency is an important factor. In web search for instance there are stringent requirements on the execution time: the document scoring phase should typically not exceed 50 milliseconds.

For this reason, there is a recent effort in trying to build models which have a reduced execution time. Recently, (Wang et al., 2010) explicitly addressed this accuracy / complexity trade-off for linear ranking functions. A linear function is of course very fast to evaluate, but the feature computation cost should also be taken into account. By performing feature selection, the authors of the aforementioned paper were able to substantially reduce the online complexity without much loss in accuracy.

In the context of decision trees ensembles, execution time can be reduced using *early exits* (Cambazoglu et al., 2010): if, based on a partial evaluation of the trees, a document does not appear to be relevant, this document is not further evaluated. The ideas in these two papers could be combined using a *cascade* architecture pioneered in (Viola and Jones, 2004). An ensemble of trees would be learned such that early trees use only cheap features while later trees are allowed to use more expensive features. But combined with early exits, these expensive trees would not be evaluated for a large number of documents. This architecture would effectively considerably reduce the only complexity of decision trees ensembles.

This line of research bears some resemblance with large scale learning (see section 4). In both cases, the goal is to reduce the execution time due to large datasets, but large scaling learning is concerned in reducing the *training* time, while the focus of this section is on the *testing* time. Nevertheless, it might be possible to transfer techniques and ideas between these two domains.

3. Sample Selection Bias

Training sets for learning to rank are typically constructed using the so-called *pooling* strategy: the top documents for one of several systems are retrieved, merged and judged. These documents are thus, by construction, more relevant than the vast majority of other documents. But what about test documents? Most learning to rank paper consider an offline *reranking* scenario: for a given query, the test documents are also retrieved by pooling and are then reranked according to the learned model. In that scenario, the training and test sets follow the same distribution.

But in a search engine, the situation is different. A web search engine typically uses a scheme with two phases (or more) to retrieve the relevant documents. The first phase is a filtering one in which the potentially relevant documents – according to a basic ranking function – are selected from the entire search engine index. Then these documents are scored in a second phase by the learned ranking function. But there is still a large number of documents in this second phase: tens of thousand. And most of these documents have

little relevance to the query. There is a thus striking difference in the document distribution between training and test. This problem is called the *sample selection bias* (Zadrozny, 2004): the documents in the training set have not been drawn at random from the test distribution; they are biased toward relevant documents.

This selection bias causes serious problems in practice. An illustration of this could be that the model has not learned to demote spam documents because it has not seen enough of them in the training set. On the other hand, because most of the documents in the training set have high text match scores, the model might not have realized that text match features are very important features.

A standard way of addressing the sample selection bias is to reweight the training samples such that reweighted training distribution matches the test distribution. These weights can be found through logistic regression (Bickel and Scheffer, 2007, Section 2.1). Once the weights have been estimated, they can readily be incorporated into a pointwise learning to rank algorithm. How to use the weights in a pointwise or listwise is algorithm is an interesting research question.

Note that the sample selection bias is related to transfer learning (see section 7) insofar as both deal with the case of different training and test distributions. But in the sample selection bias, even though the marginal distribution $P(x)$ changes between training and test, the conditional output distribution $P(y|x)$ is assumed to be fixed. In most transfer learning scenarios, this conditional output distribution shifts between training and test.

Another way of correcting this sample selection bias is to improve the scheme used for collecting training data. The pooling strategy could for instance be modified to include documents deeper in the ranking, thus reflecting more closely the test distribution. But judging more documents has a cost and this brings the question of how to select the training documents under a fixed labeling budget. This question is at the core of *active learning* (Long et al., 2010).

4. Large Scale Learning to Rank

In the literature of learning to rank, people have paid a lot of attention to the design of loss functions, but somehow overlooked the efficiency and scalability of algorithms. The latter, however, has become a more and more important issue nowadays, especially due to the availability of large-scale click-through data in Web search that can be used to train the learning to rank models.

While it is good to have more training data, it is challenging for many existing algorithms to handle such data. In order to tackle the challenge, we may want to consider one of the following approaches.

- *Parallel computing.* For example, we can use the MPI or MapReduce infrastructure to distribute the computations in the algorithms. There are a number of attempts on distributed machine learning in the literature (Chu et al., 2007; Chang et al., 2008), however, the efforts on learning to rank are still very limited.

- *Ensemble learning.* We can down-sample the data to make it easy to handle by a single-machine algorithm. After learning a ranking model based on this sample, we can repeat the sampling for multiple times and aggregate the ranking models obtained

from all the samples. It has been proved in (Ueda and Nakano, 1996) that such ensemble learning can effectively make use of large datasets and the resultant model can be more effective than the model learned from every single sample.

- *Approximate algorithms.* In some cases, we can derive an approximate version of an existing algorithm, whose complexity is much lower than the original algorithm while the accuracy remains to a certain extent. This kind of approaches have been well studied in the literature of computational theory (Sipser, 2006), but still not sufficiently investigated in learning to rank.

In addition to discussing how to handle large data from a purely computational perspective, we may want to investigate the problem from another angle. That is, it may not be always necessary to increase the data scale. With more and more data, the learning curve may get saturated, and the cost we pay for the extra computations may become wasteful. To gain more understanding on this, we need to jointly consider the learning theory and computational theory for ranking. This can also be an important piece of future work.

5. Robust Learning to Rank

In most of the existing works, multiple ranking functions are compared and evaluated, and the best ranking function is selected based on some relevance measurement, e.g. NDCG. For a commercial search engine company, the real scenario is more complex: the ranking function is required to be updated and improved periodically with more training data, newly developed ranking features, or more fancy ranking algorithms, however, the ranking results should not change dramatically. Such requirements would bring new challenge of robustness to the learning to rank area.

How to measure robustness? Multiple evaluations over time is one method but very costly. One practical solution is that robustness can be measured with the probability of switching neighboring pairs in a search result when ranking score turbulence happens (Li et al., 2009). From metrics perspective, if adding the robustness factors into the original relevance metrics, e.g. NDCG, the new metrics could be more suitable to measure relevance and robustness at the same time. However, the efforts on robustness measurement are still very preliminary.

In addition, the robustness over training data noise is another challenging task. Many of existing learning to rank algorithms are based on pairwise preference framework which is quite fragile to training data noise, since one mis-judged relevance label on a document would lead to a large number of mis-labeled document pairs. This would significantly jeopardize the robustness of the overall ranking performance. Carvalho et al. (2008) study the impacts of outlying pairs in learning to rank with pairwise preferences and introduce a new meta-learning algorithm capable of suppressing these undesirable effects. In terms of point-wise and list-wise learning algorithms, the corresponding efforts are still missing.

How to learn a robust ranking function is another interesting topic. Intuitively, if an algorithm could learn the parameters which control the metric sensitivity to the score turbulence, the generated ranking functions would be more robust. Another possible solution is related to incremental learning, which guarantees that the new model is largely similar to previous models.

6. Learning to Rank for Diversity

Most learning to rank approaches implicitly assume independence in the documents relevance. This often implies redundancy in the search results. A recent trend in learning to rank research is thus to optimize not only for relevancy, but also for diversity (Radlinski et al., 2009): the search result set should ideally cover various information *nuggets*, *facets* or *topics*.[1] This prompted a diversity task as part of the 2009 and 2010 Web tracks of TREC.[2].

An explicit diversification strategy involves 3 components: 1. A method for automatically discovering what are the information nuggets for a given query; 2. A system for assessing the relevance of a document to a particular nugget; 3. A strategy for ordering the results. This last component has been well studied recently, and in particular the optimization of an objective function with a *diminishing return* property sounds appealing (Agrawal et al., 2009). But the first two components still need to be worked out.

There also approaches to diversification which bypass these two difficult steps: instead of explicitly controlling the diversification by enumerating the different nuggets of information, the diversification is implicit. A rather common technique in this category is the minimization of textual similarities across documents. This was first proposed by Carbonell and Goldstein (1998) in their *Maximal Marginal Relevance* framework. An other approach is to use implicit user feedback in a bandit-style algorithm (Radlinski et al., 2008).

Until recently, two of the difficulties in this line of research were the lack of benchmark datasets and of a consensus on the evaluation metric. The recent diversity tasks at TREC are a commendable step in addressing these issues.

The problem of learning to rank for diversity can be cast into the more general framework of learning with structured prediction models (or relational learning) (Bakir et al., 2007): instead of predicting the relevance of the documents independently, an entire ranking is predicted. Qin et al. (2009) have proposed methods for this type of structured prediction in the context of two information retrieval tasks: Pseudo Relevance Feedback and Topic Distillation. A unified framework to handle all these different kinds of structured prediction could facilitate the development of new learning to rank algorithms where the relation between documents is important.

7. Transfer Learning to Rank

Most learning to rank algorithms are heavily dependent on a large amount of editorial labeled training data, which is time consuming and costly to obtain. Furthermore, collecting specific labeled data for different domains or different ranking applications is not scalable.

The idea of *transfer learning* – and of the related concepts of *multi-task learning* and *domain adaption* – is that if the tasks or domains share some similarities, then the training data or model from one task can be helpful in building a model for another task. The key issue in applying a transfer learning algorithm for a ranking task is to identify and understand what are the differences and similarities between the tasks: Is there only a

1. These various terms are used in the literature depending at which scale the diversity is considered.
2. See http://plg.uwaterloo.ca/~trecweb/

change in $P(x)$ as in the selection bias issue identified in section 3? Are the conditional distributions different? Do the features need to be transformed between the tasks?

As the nature of the differences between tasks can be quite diverse, there are also various types of transfer learning algorithms to cover these various cases. They can be grouped in three categories:

- *algorithm level transfer learning.* The motivation of transfer learning to rank on algorithm level is to learn different ranking tasks with a joint model, which addresses the specifics of each learning task with task-specific parameters and the commonalities between them through shared parameters, and most of existing works are related to algorithm level (Chapelle et al., 2011), (Gao et al., 2009).

- *feature level transfer learning.* How to learn some common features which are effective for all domains is a fundamental problem in the area of multi-task learning. In learning to rank scenario, a common feature could be either a meta feature learned with multiple element features, or a weak learner in the boosting framework (Chen et al., 2009). However, due to the heterogenous features across different domains, some common features may not be sufficient to train a good ranking function on target domains;

- *data level transfer learning.* On data level, the basic idea is how to adjust weight of different sources of labeled data to build a robust training superset, which is applicable for all domains. For example, (Long et al., 2009) proposed a learning framework based on the concept of label-relation function to transfer knowledge among different domains without explicitly formulating the data distribution differences;

In fact, these different levels of transfer learning are complementary, how to effectively combine them into one generic framework is an open question. Finally learning theory might be able to guide us in conceptualizing the notation of task relatedness and in providing us some generalization error analysis for multi-task learning (Ben-David and Borbely, 2008; Baxter, 2000).

8. Online Learning to Rank

Traditional learning to rank algorithms for web search are trained in a batch mode, to capture stationary relevance of documents to queries, which has limited ability to track dynamic user intention in a timely manner. For those time sensitive queries, the relevance of documents to a query on breaking news often changes over time, which indicates the batch-learned ranking functions do have limitations. User real-time click feedback could be a better and timely proxy for the varying relevance of documents rather than the editorial judgments provided by human editors. In the other word, an online learning to rank algorithm can quickly learn the best re-ranking of the top portion of the original ranked list based on real-time user click feedback.

Existing work (Moon et al., 2010) is heavily dependent on the data collected with an exploration random shuffled bucket which removes positional biases on clicks. However,

such a random shuffled bucket is costly and not practical. How to leverage ordinary real-time click information to build an online learning re-ranker on the top of batch mode ranker to improve the relevance and freshness of time sensitive queries is a still challenge task.

As the same time, using real-time click feedback would also benefit the recency ranking issue (Dong et al., 2010), which is to balance relevance and freshness of the top ranking results. Comparing the real-time click feedback over the click history, we could observe some signals to capture the buzzness of click, which can be leverage as an important feature for both ranking and time-sensitive query classification (Inagaki et al., 2010).

We have to admit that the research on online learning and recency ranking are still relatively preliminary. How to effectively combine the time sensitive features into the online learning framework is still an open question for the research community. Furthermore, as a ranking function is frequently updated according to real-time click feedback, how to keep the robustness and stability is another important challenge which can not be ignored.

References

S. Agarwal and P. Niyogi. Stability and generalization of bipartite ranking algorithms. In *Proceedings of the 18th Annual Conference on Learning Theory (COLT 2005)*, pages 32–47, 2005.

R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *WSDM '09: Proceedings of the 2nd international conference on Web search and web data mining*, pages 5–14. ACM, 2009.

G. Bakir, T. Hofmann, B. Schölkopf, A. Smola, B. Taskar, and S. V. N. Vishwanathan. *Predicting Structured Data*. The MIT Press, 2007. ISBN 0262026171.

J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

S. Ben-David and R.S. Borbely. A notion of task relatedness yielding provable multiple-task learning guarantees. *Machine learning*, 73(3):273–287, 2008. ISSN 0885-6125.

S. Bickel and T. Scheffer. Dirichlet-enhanced spam filtering based on biased samples. In *Advances in Neural Information Processing Systems 19*, 2007.

B.B. Cambazoglu, H. Zaragoza, O. Chapelle, J. Chen, C. Liao, Z. Zheng, and J. Degenhardt. Early exit optimizations for additive machine learned ranking systems. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 411–420, 2010.

J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336, 1998.

V. Carvalho, J. Elsas, W. Cohen, and J. Carbonell. A meta-learning approach for robust rank learning. In *Proceedings of SIGIR 2008 LR4IR - Workshop on Learning to Rank for Information Retrieval*, 2008.

E. Chang, K. Zhu, H. Wang, H. Bai, J. Li, Z. Qiu, and H. Cui. Parallelizing support vector machines on distributed computers. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 257–264. MIT Press, Cambridge, MA, 2008.

O. Chapelle, P. Shivaswamy, S. Vadrevu, K. Weinberger, Y. Zhang, and B. Tseng. Boosted multi-task learning. *Machine Learning Journal*, 2011. To appear.

D. Chen, Y. Xiong, J. Yan, G.-R. Xue, G. Wang, and Z. Chen. Knowledge transfer for cross domain learning to rank. *Information Retrieval*, 13(3):236–253, 2009.

W. Chen, T.-Y. Liu, Y. Lan, Z. Ma, and H. Li. Ranking measures and loss functions in learning to rank. In *Advances in Neural Information Processing Systems 22 (NIPS 2009)*, pages 315–323, 2010a.

W. Chen, T.-Y. Liu, and Z. Ma. Two-layer generalization analysis for ranking using rademacher average. Technical report, Advances in Neural Information Processing Systems 23 (NIPS 2010), 2010b.

C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, A. Ng, and K. Olukotun. Map-reduce for machine learning on multicore. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 281–288. MIT Press, Cambridge, MA, 2007.

S. Clemencon and N. Vayatis. Ranking the best instances. *Journal of Machine Learning Research*, 8(Dec):2671–2699, 2007.

D. Cossock and T. Zhang. Subset ranking using regression. In *Proceedings of the 19th Annual Conference on Learning Theory (COLT 2006)*, pages 605–619, 2006.

A. Dong, Y. Chang, Z. Zheng, G. Mishne, J. Bai, R. Zhang, K. Buchner, C. Liao, and F. Diaz. Towards recency ranking in web search. In *Proceedings of WSDM 2010*, 2010.

Y. Freund, R. Iyer, R. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003. ISSN 1533-7928.

J. Gao, Q. Wu, C. Burges, K. Svore, Y. Su, N. Khan, S. Shah, and H. Zhou. Model adaptation via model interpolation and boosting for web search ranking. In *Proceedings of EMNLP 2009*, 2009.

Y. Inagaki, N. Sadagopan, G. Dupret, C. Liao, A. Dong, Y. Chang, and Z. Zheng. Session based click features for recency ranking. In *Proceedings of AAAI 2010*, 2010.

T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2002)*, pages 133–142, 2002. ISBN 1-58113-567-X.

Y. Lan and T.-Y. Liu. Generalization analysis of listwise learning-to-rank algorithms. In *Proceedings of the 26th International Conference on Machine Learning (ICML 2009)*, pages 577–584, 2009.

Y. Lan, T.-Y. Liu, T. Qin, Z. Ma, and H. Li. Query-level stability and generalization in learning to rank. In *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, pages 512–519, 2008.

X. Li, F. Li, S. Ji, Z. Zheng, A. Dong, and Y. Chang. Incorporating robustness into web ranking evaluation. In *Proceedings of the CIKM*, 2009.

B. Long, S. Lamkhede, S. Vadrevu, Y. Zhang, and B. Tseng. A risk minimization framework for domain adaptation. In *Proceedings of CIKM 2009*, 2009.

B. Long, O. Chapelle, Y. Zhang, Y. Chang, Z. Zheng, and B. Tseng. Active learning for ranking through expected loss optimization. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 267–274, 2010.

T. Moon, L. Li, W. Chu, C. Liao, Z. Zheng, and Y. Chang. Online learning for recency search ranking using real-time user feedback. In *Proceedings of CIKM 2010*, 2010.

T. Qin, T.-Y. Liu, X.-D. Zhang, D.-S. Wang, and H. Li. Global ranking using continuous conditional random fields. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1281–1288, 2009.

F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th international conference on Machine learning*, pages 784–791, 2008.

F. Radlinski, P. Bennett, B. Carterette, and T. Joachims. Redundancy, diversity and interdependent document relevance. *SIGIR Forum*, 43, 2009.

M. Sipser. *Introduction to the Theory of Computation(2nd ed.)*. PWS Publishing, 2006. ISBN 0-534-94728-X.

N. Ueda and R. Nakano. Generalization error of ensemble estimators. In *IEEE International Conference on Neural Networks*, pages 90–95, 1996.

P. A. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

L. Wang, J. Lin, and D. Metzler. Learning to efficiently rank. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 138–145, 2010.

B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the twenty-first international conference on Machine learning*, 2004.