NIEK TAX

METHODS FOR LARGE SCALE LEARNING-TO-RANK

METHODS FOR LARGE SCALE LEARNING-TO-RANK

A study concerning parallelization of Learning-to-Rank algorithms using Hadoop

NIEK TAX BSC.
Databases
Electrical Engineering, Mathematics and Computer Science (EEMCS)
University of Twente

UNIVERSITY OF TWENTE.

avanade®
Results Realized

Februari 2014 – version 0.1

Ohana means family.
Family means nobody gets left behind, or forgotten.

— Lilo & Stitch

ABSTRACT

Short summary of the contents in English…

SAMENVATTING

Een korte samenvatting in het Nederlands…

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

Part I

INTRODUCTION

1 MOTIVATION AND PROBLEM STATEMENT

Ranking is a core problem in the field of information retrieval. The ranking task in information retrieval entails the ranking of candidate documents according to their relevance for a query. Heuristic ranking models have long been around in the information retrieval field. The increasing amounts of potential training data have recently made it possible to leverage machine learning methods to obtain more effective models. Learning-to-Rank is the relatively new research area covering the use of machine learning models for the ranking task.

In recent years several Learning-to-Rank benchmark datasets have been proposed that enable comparison of the performance of different Learning-to-Rank methods. Well-known benchmark datasets include the *Yahoo! Learning to Rank Challenge* dataset[5], the Yandex Internet Mathematics competition[1], and the LETOR dataset[11] that was build by Microsoft Research. One of the concluding observations of the *Yahoo! Learning to Rank Challenge* was that almost all work in the Learning-to-Rank field focuses on ranking accuracy, while efficiency and scalability of Learning-to-Rank algorithms is still an underexposed research area that is likely to become more important in the near future as training sets are becoming larger and larger[6]. Liu[10] confirms the observation that efficiency and scalability of Learning-to-Rank methods has so far been an overlooked research area in his influential book on Learning-to-Rank.

Some research has been done in the area of on parallel or distributed machine learning [7, 4]. However, almost none of these studies include the Learning-to-Rank subfield of machine learning. The field of efficient Learning-to-Rank has been getting some attention lately [1, 2, 3, 13, 12], since Liu [10] first stated its growing importance back in 2007. Only several of these studies [13, 12] have explored the possibilities of efficient Learning-to-Rank through the use of parallel programming paradigms.

MapReduce[8] is a parallel programming framework that is inspired by the *Map* and *Reduce* functions commonly used in functional programming. Since Google developed the MapReduce parallel programming framework back in 2004 it has since grown to be the industry standard model for parallel programming. Lin [9] observed that algorithms that are of iterative nature, which most Learning-to-

1 http://imat-relpred.yandex.ru/en/

Rank algorithms are, are not amenable to the MapReduce framework. Lin argued that as a solution to the non-amenability of iterative algorithms to the MapReduce framework, iterative algorithms can often be replaced with non-iterative alternatives or can still be optimized in such a way that its performance in a MapReduce setting is good enough.

The appearance of benchmark datasets gave insight in the performance of different Learning-to-Rank approaches, which resulted in increasing popularity of those methods that showed to perform well on one or more benchmark datasets. Up to now it remains unknown whether popular existing Learning-to-Rank methods scale well when they are used in a parallel manner using the MapReduce framework. This thesis aims to be an explorative start in this little researched area of parallel Learning-to-Rank. A more extensive overview of my research goals and questions are described in chapter 2.

2 RESEARCH GOALS

The objective of this thesis is to explore the speed-up in execution time of Learning-to-Rank algorithms through parallelization using the MapReduce framework. This work focuses on those Learning-to-Rank algorithms that have shown leading performance on relevant benchmark datasets. The research questions raised and answered in this work include:

- What is the speed-up of existing Learning-to-Rank algorithms when executed using the MapReduce framework?
- Can we adjust those Learning-to-Rank algorithms such that the parallel execution speed-up increases without decreasing accuracy?

3 APPROACH

To answer the first research question I will implement Learning-to-Rank methods in the MapReduce framework and measuring the run-time as a factor of the number of cluster nodes used to complete the computation.

To implement the Learning-to-Rank algorithms I will use cloud based MapReduce implementation from Microsoft was used that is called HDInsight. It which is based on the popular MapReduce open source implementation Hadoop[1]. The algorithms that we include in the measurements will be determined based on experimental results on the *Yahoo! Learning to Rank Challenge*[5], the Yandex Internet Mathematics competition[2], the LETOR[11] dataset and the LETOR successor MSLR-WEB30k.

1 http://hadoop.apache.org/
2 http://imat-relpred.yandex.ru/en/

4 THESIS OVERVIEW

PART II: BACKGROUND introduces the reader to the basic principles and recent work in the fields of Learning-to-Rank.

PART III: RELATED WORK concisely describes existing work in the field of parallel machine learning and parallel Learning-to-Rank.

PART IV: BENCHMARK RESULTS sketches the performance of existing Learning-to-Rank methods on several benchmark datasets and describes the selection of Learning-to-Rank methods for the parallelization experiments.

PART V: SELECTED LEARNING-TO-RANK METHODS describes the algorithms and details of the selected Learning-to-Rank methods.

PART VI: IMPLEMENTATION describes implementation details of the Learning-to-Rank algorithms in the Hadoop framework.

PART VII: RESULTS & DISCUSSION presents and discusses speed-up results for the implemented Learning-to-Rank methods.

PART VIII: CONCLUSION summarizes the results and answers our research questions based on the results. The limitations of our research as well as future research directions in the field are mentioned here.

Part II

BACKGROUND

This part provides a background in the Learning-to-Rank field with the goal of making the subsequential parts of this thesis understandable for non-experts in the field.

5 INTRODUCTION TO LEARNING-TO-RANK

Different definitions of Learning-to-Rank exist. In general, all ranking methods that use machine learning technologies to solve the problem of ranking are called Learning-to-Rank methods. Figure 1 describes the general process of machine learning. It shows training elements from an input space that are mapped to an output space using a model such that the difference between the actual labels of the training elements and the labels predicted with with the model are minimal in terms of a loss function.

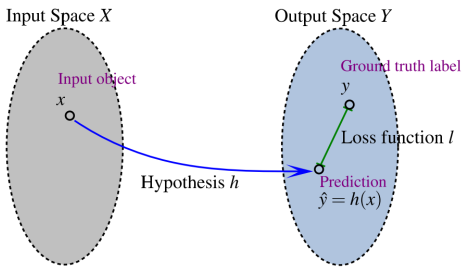

Figure 1: Machine learning framework for Learning-to-Rank, from Liu[10]

Liu [10] proposes a more narrow definition and only considers ranking methods to be a Learning-to-Rank method when it is *feature based* and uses *discriminative training*, which are itself defined as follows:

FEATURE BASED *Feature based* means that all documents under investigation are represented by feature vectors that reflect the relevance of the documents to the query.

DISCRIMINATIVE TRAINING *Discriminative training* means that the learning process can be well described by the four components of discriminative learning. That is, a Learning-to-Rank method has its own *input space*, *output space*, *hypothesis space*, and *loss function*, like the machine learning process described by Figure 1. *Input space*, *output space*, *hypothesis space*, and *loss function* are hereby defined as follows:

INPUT SPACE contains the objects under investigation. Usually objects are represented by feature vectors, extracted according to different applications.

OUTPUT SPACE contains the learning target with respect to the input objects.

HYPOTHESIS SPACE defines the class of functions mapping the input space to the output space. The functions operate on the feature vectors of the input object, and make predictions according to the format of the output space.

LOSS FUNCTION in order to learn the optimal hypothesis, a training set is usually used, which contains a number of objects and their ground truth labels, sampled from the product of the input and output spaces.

Figure 2 shows how the machine learning process as described in Figure 1 typically takes place in a ranking scenario. A set of queries q_i with $n > i > 1$, the documents associated with these queries which are represented by feature vector x_i, and the relevant judgments of those documents y_i are used together to train a model h, that can predict a ranking of the documents y_i, such the difference between the document rankings predicted by h and the actual optimal rankings based on y_i is are minimal in terms of a loss function.

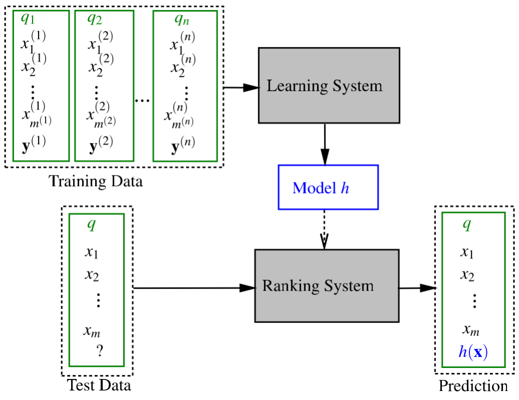

Figure 2: A typical Learning-to-Rank setting, from Liu[10]

The predictions and the loss function might either be defined for:

1. the relevance of a single document
2. the classification of the most relevant document out of a document-pair
3. the ranking of documents directly

These three approaches are in literature respectively called the pointwise approach, the pairwise approach and the listwise approach. In the following sections we will further describe the three different approaches in Learning-to-Rank.

5.1 POINTWISE APPROACH

5.2 PAIRWISE APPROACH

5.3 LISTWISE APPROACH

Part III

RELATED WORK

Part IV

BENCHMARK RESULTS

This part describes benchmark characteristics like collection size and features and gives an overview of the performance of different Learning-to-Rank methods. Performance differences for a given Learning-to-Rank method are explained in terms of differences in benchmark characteristics. This section is concluded with a selection of Learning-to-Rank methods to include in the experiments.

6 YAHOO! LEARNING TO RANK CHALLENGE

6.1 benchmark characteristics

6.2 results

7 YANDEX INTERNET MATHEMATICS COMPETITION

7.1 BENCHMARK CHARACTERISTICS

7.2 RESULTS

8 LETOR

8.1 BENCHMARK CHARACTERISTICS

8.2 RESULTS

9 MSLR-WEB30K

9.1 Benchmark characteristics

9.2 Results

Part V

SELECTED LEARNING-TO-RANK METHODS

Algorithms and details of the well-performing Learning-to-Rank methods as selected in the aforegoing part are presented and explained in this part.

Part VI

IMPLEMENTATION

Describes implementation details of the Learning-to-Rank algorithm implementations in HDInsight that are either Hadoop, Microsoft Azure, or HDInsight and are not part of the algorithm specification itself.

Part VII

RESULTS & DISCUSSION

Part VIII

CONCLUSIONS

Part IX

APPENDIX

BIBLIOGRAPHY

[1] Asadi, N., and Lin, J. Training Efficient Tree-Based Models for Document Ranking. In *Advances in Information Retrieval SE - 13*, P. Serdyukov, P. Braslavski, S. Kuznetsov, J. Kamps, S. Rüger, E. Agichtein, I. Segalovich, and E. Yilmaz, Eds., vol. 7814 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 146–157.

[2] Asadi, N., Lin, J., and de Vries, A. Runtime Optimizations for Prediction with Tree-Based Models. *IEEE Transactions on Knowledge and Data Engineering PP*, 99 (2013), 1.

[3] Busa-Fekete, R., Benbouzid, D., and Kégl, B. Fast classification using sparse decision dags. In *29th International Conference on Machine Learning (ICML 2012)* (2012).

[4] Chang, E. Y., Zhu, K., Wang, H., Bai, H., Li, J., Qiu, Z., and Cui, H. PSVM: Parallelizing support vector machines on distributed computers. In *Advances in Neural Information Processing Systems* (2007), pp. 257–264.

[5] Chapelle, O., and Chang, Y. Yahoo! Learning to Rank Challenge Overview. *Journal of Machine Learning Research-Proceedings Track 14* (2011), 1–24.

[6] Chapelle, O., Chang, Y., and Liu, T.-Y. Future directions in learning to rank. *JMLR Workshop and Conference Proceedings 14* (2011), 91–100.

[7] Chu, C., Kim, S. K., Lin, Y.-A., Yu, Y., Bradski, G., Ng, A. Y., and Olukotun, K. Map-reduce for machine learning on multicore. *Advances in neural information processing systems 19* (2007), 281.

[8] Dean, J., and Ghemawat, S. MapReduce: simplified data processing on large clusters. *Communications of the ACM 51*, 1 (2004), 107–113.

[9] Lin, J. Mapreduce is Good Enough? If All You Have is a Hammer, Throw Away Everything That's Not a Nail! *Big Data 1*, 1 (2013), 28–37.

[10] Liu, T.-Y. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval 3*, 3 (Mar. 2007), 225–331.

[11] Qin, T., Liu, T.-Y., Xu, J., and Li, H. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval 13*, 4 (2010), 346–374.

[12] Shukla, S., Lease, M., and Tewari, A. Parallelizing ListNet Training Using Spark. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2012), SIGIR '12, ACM, pp. 1127–1128.

[13] Sousa, D., Rosa, T., Martins, W., Silva, R., and Gonçalves, M. Improving On-Demand Learning to Rank through Parallelism. In *Web Information Systems Engineering - WISE 2012 SE - 38*, X. Wang, I. Cruz, A. Delis, and G. Huang, Eds., vol. 7651 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 526–537.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

http://code.google.com/p/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

DECLARATION

Put your declaration here.

Enschede, Februari 2014

Niek Tax