The Impact of Linkage Methods in Hierarchical Clustering for Active Learning to Rank

Ziming Li
University of Amsterdam
Amsterdam, The Netherlands
z.li@uva.nl

Maarten de Rijke
University of Amsterdam
Amsterdam, The Netherlands
derijke@uva.nl

ABSTRACT

Document ranking is a central problem in many areas, including information retrieval and recommendation. The goal of learning to rank is to automatically create ranking models from training data. The performance of ranking models is strongly affected by the quality and quantity of training data. Collecting large scale training samples with relevance labels involves human labor which is time-consuming and expensive. Selective sampling and active learning techniques have been developed and proven effective in addressing this problem. However, most active methods do not scale well and need to rebuild the model after selected samples are added to the previous training set. We propose a sampling method which selects a set of instances and labels the full set only once before training the ranking model. Our method is based on hierarchical agglomerative clustering (average linkage) and we also report the performance of other linkage criteria that measure the distance between two clusters of query-document pairs. Another difference from previous hierarchical clustering is that we cluster the instances belonging to the same query, which usually outperforms the baselines.

CCS CONCEPTS

•Information systems →Learning to rank;

ACM Reference format:
Ziming Li and Maarten de Rijke. 2017. The Impact of Linkage Methods in Hierarchical Clustering for Active Learning to Rank. In *Proceedings of SIGIR '17, Shinjuku, Tokyo, Japan, August 07-11, 2017,* 4 pages.
DOI: http://dx.doi.org/10.1145/3077136.3080684

1 INTRODUCTION

Document ranking is an essential feature in many information retrieval applications. How to sort the returned results according to their degree of relevance has given birth to the area of learning to rank [6], which aims to automatically create ranking models from a training dataset; the learned models are then used to rank the results of new queries. Many learning to rank algorithms have been proposed; they can be categorized into three types of approach: pointwise [2], pairwise [5] and listwise [1].

SIGIR '17, Shinjuku, Tokyo, Japan

978-1-4503-5022-8/17/08…$15.00
DOI: http://dx.doi.org/10.1145/3077136.3080684

Like other supervised machine learning methods, the performance of a ranking model highly depends on the quantity and quality of training datasets. To create training datasets, experts are hired to manually provide relevance labels for training data, which is expensive and time-consuming. Human labeling, whether by experts or crowds, can be noisy and biased [11]. Active learning is a paradigm to reduce the labeling effort [12]; it has mostly been studied in the context of classification tasks [9, 10, 14].

In this paper, we present an active learning method to select the most informative query-document pairs to be labeled for learning to rank. Our method relies on hierarchical clustering. Unlike traditional active learning methods, our method is unsupervised and the selected training sets can be used to train different learning to rank models. We build on the hypothesis that the information contained in an instance is highly correlated to the instance position in the feature space. Hierarchical clustering has the ability to group instances with similar information into the same cluster and each cluster can be represented by its centroid. While most active learning methods need to rebuild the training models each time new labeled documents are added to the training set, our method labels the instances only once before the training process. We first evaluate our method on three datasets from Letor 3.0 and find that the performance of our method is similar or superior to the baselines while we can achieve full training performance with fewer instances. We also analyze the limitations of our method and find that the effectiveness of our sampling method is closely related to the features and structures each dataset has.

2 RELATED WORK

In order to address the lack of labeled data, active learning has proven to be a promising direction that aims to achieve high accuracy using as few labeled instances as possible [12]. A number of active learning methods have been proposed for classification. Most methods start with only a small set of labeled instances and sequentially select the most informative instances to be labeled by an oracle. The trained model will be updated when new labeled instances are added to the training set. Different strategies are proposed to choose the most informative instances that can maximize the information value to the current model [9, 10, 14].

It is not straightforward to extend these methods to ranking problems. On the one hand, these methods try to minimize the classification error and do not take into account the rank order while position-based measures are usually non-continuous and non-differentiable. On the other, each instance in most supervised classification tasks can be treated as independent of each other while the instances in learning to rank are conditionally independent. Compared with traditional classification problems, there is

limited work about active learning in ranking. Long et al. [8] adopt a two-stage active learning strategy schema to integrate query level and document level data selection. They select samples minimizing the expected loss as the most informative ones and achieves good results in the case of web search ranking. But this method requires a relative big seed set and the ranking models are restricted to pointwise models. Donmez and Carbonell [3] select the documents that wolud impart the greatest change to the current model. While all these methods sequentially select instances to be labeled, Silva et al. [13] adopt hierarchical clustering to "compress" the original training set, which is the state-of-the-art selection method in overall performance and efficiency. The method we propose can be viewed as an extension of [13]. The difference are that we cluster the query-document pairs belonging to the same query separately and average linkage is used, which achieves better performance.

3 METHOD

3.1 Hierarchical agglomerative clustering

In hierarchical agglomerative clustering, each instance is regarded as a singleton cluster and then merged based on the distance or similarity between clusters until there is only one cluster that contains all instances. The structure of the final cluster is a tree or dendrogram and each level of the resulting tree is a segmentation of the data. For two given clusters, C_1 and C_2, and a non-negative real value λ, if distance $f(C_1, C_2) < \lambda$, then C_1 and C_2 will be merged.

According to the merging rule, the number of clusters is associated with the value of λ, which is the indistinguishability threshold [13]. Different linkage criterions have been proposed to measure the distance or dissimilarity between two clusters. We use *average* as our linkage criterion and we also report the performance of minimum linkage, maximum linkage and ward linkage.

3.1.1 Minimum linkage clustering. In minimum linkage clustering (also called single linkage clustering), one of the simplest agglomerative hierarchical clustering methods, the value of the shortest link from any member of one cluster to the member of another cluster denotes the distance of these two clusters. For two clusters C_1 and C_2, the distance between C_1 and C_2 is:

$$f(C_1, C_2) = \min_{u \in C_1, v \in C_2} d(u, v),$$

In this paper, d is the Euclidean distance. This method is also known as the nearest technique and used in [13]. In this case, minimum linkage clustering can group the instances in "stringy" clusters and be converted to find the Minimum Spanning Tree (MST) of query-document pairs [4]. By deleting all edges longer than a specified indistinguishability threshold in the MST, the remaining connected instances form a hierarchical cluster.

3.1.2 Average linkage clustering. Average linkage clustering uses the average of distances between all pairs of instances, where each pair consists of two points from two different clusters, as the distance between two clusters:

$$f(C_1, C_2) = \frac{1}{N_1 * N_2} \sum_{u \in C_1, v \in C_2} d(u, v),$$

where N_1 and N_2 are the sizes of clusters C_1 and C_2, respectively. We use average linkage as our linkage criterion to measure the cluster distance and we perform clustering on each subset which contains all the query-document pairs belonging to the same query.

3.1.3 Maximum linkage clustering. Different from single linkage clustering, the distance of two clusters in maximum linkage clustering is the value of the largest link from one cluster to another cluster. For two clusters C_1 and C_2, the distance is computed as follows:

$$f(C_1, C_2) = \max_{u \in C_1, v \in C_2} d(u, v),$$

where d is the Euclidean distance.

3.1.4 Ward linkage clustering. In Ward's linkage method, the distance between two clusters is the sum of the squares of the distances between all objects in the cluster and the centroid of the cluster. For two clusters C_1 and C_2, the distance is computed as follows:

$$f(C_1, C_2) = \sum_{x \in C_1 \cup C_2} d(x, \mu_{C_1 \cup C_2})^2,$$

where μ is the centroid of the new cluster merged from C_1 and C_2.

3.2 Hierarchical clustering for learning to rank

In this paper, we apply different linkage criteria to hierarchical clustering. As shown in Algorithm 1, we first cut all the query-documents pairs into different subsets which have different query ids and then adopt hierarchical clustering on each subset. After the last two steps, we can get the clusters on each subset. In our sampling strategy, we use the instance closest to the geometric centroid of each cluster to represent all the query-document pairs in this cluster. In fact, the clustering distribution shows that there is only one single point in most clusters. The final dataset to be labeled is made up of all the selected instances.

Algorithm 1 Hierarchical Clustering on Each Query (HCEQ)

Require: Unlabeled dataset D with m queries, desired sampling size n, linkage criterion $linkage$
Ensure: The subset S to be labeled

```
D_1, D_2, ..., D_i, ..., D_m ← DividingDataset(D).
S ← ∅
for all i ∈ {1, 2, ..., m} do
    n_i ← n * |D_i| / |D|
    C_1, C_2, ..., C_j, ..., C_{n_i} ← AggClustering(D_i, n_i, linkage)
    for all j ∈ {1, 2, ..., n_i} do
        g_j ← GeometricCentroid(C_j)
        p_j ← NearestNeighbour(C_j, g_j)
        S ← S ∪ {p_j}
    end for
end for
return S
```

4 EXPERIMENTAL SETUP

4.1 Data Sets and Evaluation Measure

To compare the performance of different linkage criteria, we apply hierarchical clustering to a well-known L2R benchmarking collection, Letor 3.0 [7]. In Letor 3.0, there are 7 datasets, based

on two document collections: Gov and OHSUMED. We focus on topic distillation (TD2003, TD2004) from GOV and OHSUMED from OHSUMED. In the sets from GOV, each instance is represented by 64 features extracted from the corresponding query-document pairs and has a label indicating its relevance level. The datasets from GOV have binary relevance labels (*relevant*, *not relevant*) while OHSUMED has 3 levels (*definitely relevant*, *possibly relevant*, *not relevant*) and each instance is represented by a 45-dimensional feature vector. Different datasets have different numbers of instances; there are 50, 75 and 106 queries, each with 50K, 74K and 16K instances in TD2003, TD2004 and OHSUMED, respectively.

The metric we use is Normalized Discounted Cumulative Gain (NDCG). We run 5-fold cross-validation on all datasets which are query-level normalized and report the average NDCG@10 on 5 folds as final performance.

4.2 Baselines

We compare the results obtained using our methods with the approach in [13], which is also based on hierarchical clustering; the results of random sampling and the full training set are also reported for reference:

Cover. The method proposed in [13] is an unsupervised and compression-based selection mechanism that tries to create a small and highly informative set to represent the original training dataset. Hierarchical clustering (single linkage) is employed to group query-document pairs into different clusters with required number of clusters. The authors also use the instance closest to the geometric centroid of each cluster to represent all the query-document pairs in this cluster and form the final training set to be labeled. Unlike our method, they perform clustering globally and instances belonging to different queries could be grouped into the same cluster.

Random. The instances to be labeled are selected randomly and no active learning methods are used here. For the same dataset, we run random sampling 10 times and report the average performance.

Full Training. We use the labeled original training datasets to train the learning to rank models.

To be able to show the difference between our methods and [13], we select SVMRank and Random Forests as our ranking models which are also used in [13]. The parameters of SVMRank and Random Forests are tuned using a small validation set.

We run all sampling methods until the fraction of selected instances reaches 50% of the original set.

5 RESULTS AND ANALYSIS

5.1 Experimental Results

Fig. 1 shows the NDCG@10 of different linkage criteria and baselines (denoted by *Average*, *Max*, *Min*, *Ward*, *Cover*, *Random* and *Full* respectively) on the TD2003, TD2004 and OHSUMED datasets. As we use the instance closest to the centroid to represent the corresponding cluster, the instance selected before may not be selected in later sampling rounds; accuracy is not monotonically increasing.

On TD2003, in terms of SVMRank (Fig. 1(a)), the accuracy of *Average* first exceeds the accuracy of *Full* at size 2%. Before 24% of the original training set have been selected, all the curves fluctuate around 0.35 except *Random* and *Ward*. When more and more instances are added to the training set, the performance of *Cover* goes down and becomes worse than *Full*. For Random Forests (Fig. 1(d)), *Average* achieves the highest accuracy before 4% of the original training set has been selected and is the first one to reach the same accuracy as *Full*. The accuracies of *Ward*, *Max* and *Cover* start from relatively low points. All methods achieve the performance of *Full* at around 12% except *Random*. After 18%, *Cover* stays below *Full* (close to 0.36) and fluctuates around 0.35.

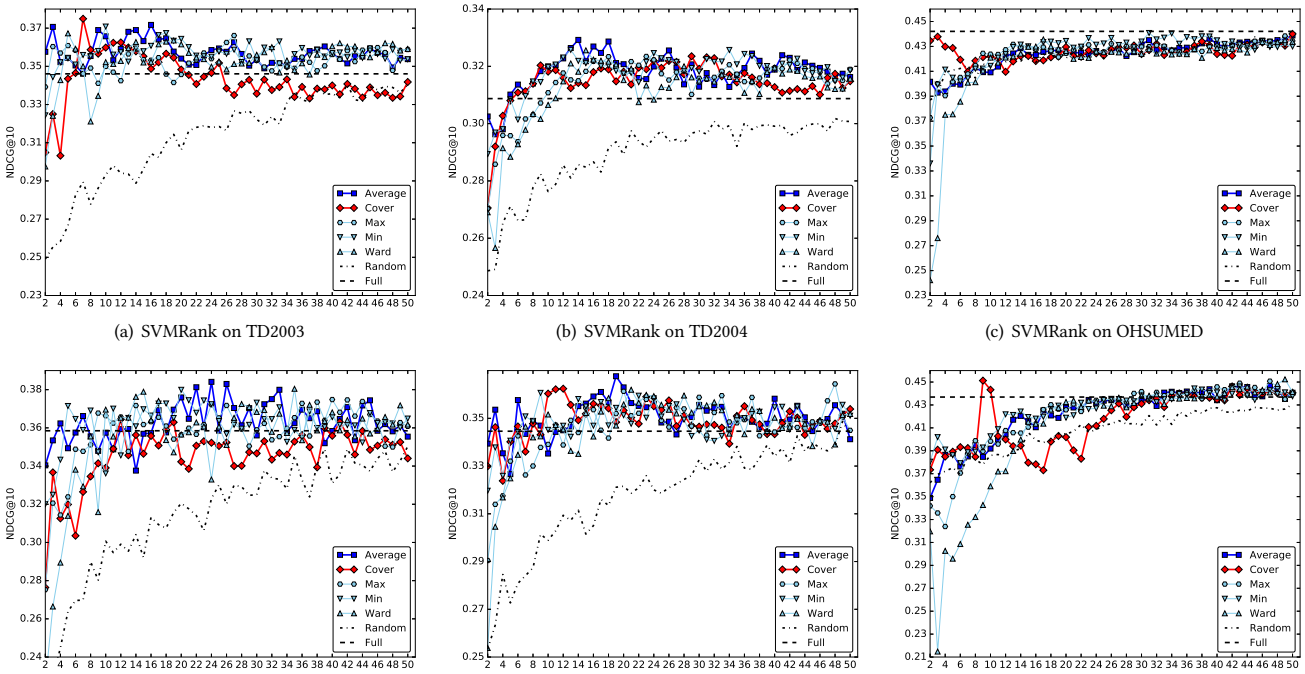
Fig. 1(b) and 1(e) describe the performance of different methods on the TD2004 dataset. In Fig. 1(b) we see that *Average* has the highest starting point when 2% of the original training set have been selected. However, after one more percent has been added to the training set, all methods have a very similar performance at around 0.295. *Average* and *Cover* reach the performance of *Full* with about 5% and continue to rise before they reach their peaks with 13% and 8%, respectively. Except *Random*, all methods reach the performance of *Full* with about 11% selected; their accuracy stays higher than that of *Full*. In terms of Random Forests (Fig. 1(e)), *Average* still has the highest starting accuracy and after the fluctuations before 11% of the training set has been selected, *Average* always performs better than *Full* and peaks around 19% of the training set, which is also the highest accuracy in all methods. *Cover* has a relatively high accuracy when the selected size is 7– 11%.

The OHSUMED dataset is based on another document collection, different from TD2003 and TD2004. The performance of SVMRank and Random Forests on OHSUMED are shown in Fig. 1(c) and 1(f), respectively. In terms of SVMRank, an interesting thing is that *Random* has similar performance as hierarchical clustering, which means that hierarchical clustering plays a small role when selecting informative instances in this case. With respect to Random Forests, *Average*, *Min*, *Ward* and *Max* have very similar performance after 16% of the training data has been selected and they reach the accuracy of *Full* at around 30%. Although *Cover* is the first method to achieve full training set performance with around 7% and reaches the peak at the same time, its accuracy is not stable and dramatically decreases until 17% of the training set has been selected. After 30% of the original training set has been selected, all methods achieve the same performance as *Full*.

5.2 Analysis

As we can see from the results presented above, most of the time, *Average* outperforms other methods on the TD2003 and TD2004 datasets. Although *Min* and *Cover* both use the shortest link to measure the distance between two clusters, they have different performance and *Min* performs better. One possible reason is that our selection mechanism can guarantee that the proportions of selected instances from each query are same and every query contributes to the final performance. Another reason might be *Cover* clusters query-document pairs globally, which causes that query-document pairs from different queries will be represented by the instances from one specific query. And this will limit the number of applicable instance pairs for pairwise ranking models.

On OHSUMED, *Average*, *Min* and *Max* have similar and stable performance while *Cover* fluctuates dramatically when relatively few instances are selected, especially with the Random Forests



(a) SVMRank on TD2003 (b) SVMRank on TD2004 (c) SVMRank on OHSUMED

(d) Random Forests on TD2003 (e) Random Forests on TD2004 (f) Random Forests on OHSUMED

Figure 1: Performance on TD2003, TD2004, and OHSUMED. The x-axis displays the percentage of selected instances.

learner. The difference between the proposed methods and *Random* is not significant. How datasets are constructed and what structures datasets have will influence the performance of clustering-based active learning methods. For example, an instance from the OHSUMED dataset is represented by 45 features which is fewer than for TD2003 and 2004, and some specific features could have greater impact on the clustering results. In addition, a query has only about 152 associated documents in OHSUMED. When we cluster query-document pairs with respect to each query, the number of selected instances from each query is small and every individual query will have very few associated documents which can influence the performance of ranking models.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we adopt hierarchical clustering to select the most informative instances for learning to rank and report the performance of different linkage criteria and baselines. On the Letor 3.0 dataset, the performance of average linkage is similar or superior to the baselines while fewer instances are needed. In the future, we will investigate how to make our method more stronger and robust on different datasets. One possible direction is to detect correlations between specific features and clustering performance. How to choose an optimal fraction of instances for each query while the total number of selected instances is fixed is another future direction worth exploring.

Acknowledgments. This research was supported by Ahold Delhaize, Amsterdam Data Science, the Bloomberg Research Grant program, the Dutch national program COMMIT, Elsevier, the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement nr 312827 (VOX-Pol), the Microsoft Research Ph.D. program, the Netherlands Institute for Sound and Vision, the Netherlands Organisation for Scientific Research (NWO) under project nrs 612.001.116, HOR-11-10, CI-14-25, 652.002.001, 612.001.551, 652.001.003, and Yandex. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *ICML '07*, pages 129–136. ACM, 2007.
[2] K. Crammer and Y. Singer. Pranking with ranking. In *NIPS '01*, pages 641–647, 2001.
[3] P. Donmez and J. G. Carbonell. Optimizing estimated loss reduction for active sampling in rank learning. In *ICML '08*, pages 248–255, 2008.
[4] J. C. Gower and G. Ross. Minimum spanning trees and single linkage cluster analysis. *Applied statistics*, pages 54–64, 1969.
[5] R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. In *ICANN '99*, pages 97–102, 1999.
[6] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.
[7] T.-Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, pages 3–10, 2007.
[8] B. Long, J. Bian, O. Chapelle, Y. Zhang, Y. Inagaki, and Y. Chang. Active learning for ranking through expected loss optimization. *IEEE Transactions on Knowledge and Data Engineering*, 27(5):1180–1191, 2015.
[9] A. K. McCallumzy and K. Nigamy. Employing em and pool-based active learning for text classification. In *ICML '98*, pages 359–367, 1998.
[10] H. T. Nguyen and A. Smeulders. Active learning using pre-clustering. In *ICML '04*, page 79. ACM, 2004.
[11] F. Radlinski and T. Joachims. Active exploration for learning rankings from clickthrough data. In *KDD '07*, pages 570–579. ACM, 2007.
[12] B. Settles. Active learning literature survey. Technical report, University of Wisconsin–Madison, 2009.
[13] R. M. Silva, G. Gomes, M. S. Alvim, and M. A. Gonçalves. Compression-based selective sampling for learning to rank. In *CIKM '16*, pages 247–256. ACM, 2016.
[14] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2(Nov):45–66, 2001.