1 Introduction

When given a set of documents and a query we want to rank these documents in terms of relevance to the query. For example, given the query ”cute kittens” we would expect that documents pertaining to kittens which are cute to be considered important and those pertaining to dogs to be ranked low. Traditional techniques for ranking involve hand crafted heuristcs or probabilitic approaches.

A modern technique for ranking documents is called learning to rank. Learning to rank is a technique in which we use machine learning tools to rank documents when compared to a query.

2 Method

Our method is composed of two parts: (1) construct our ranking algorithm (2) evaluating new documents in relation to a query.

- Pre-process Data
- Create clusters
  - Create cluster pairs
- For each cluster, and cluster pairs, create a learner
- Given a set of documents for a query we create pairs of each document
- For each document pair we choose our learner based on which cluster
- Using our pair rankings we construct a graph where those ranked higher have out edges to those ranked less
- From our graph we rank documents based on the number of out edges and in edges of a node.
  - We use the ratio of out degrees to end degrees to select the current top ranking document
  - once we have selected a document we remove it from the graph and repeat until there are no more documents

3 Results

In figure 1 we present each result for each underlying learner along with the best result using our method with each learn. For RankBoost and RankNet we acheve the best results with 5 clusters, for AdaRank and LambdaRank we acheive the best result with 2 clusters. RankBoost is the only underlying learner which results in better performance when we apply our method for K values of 2, 3, 4, 5, and 8. When using 8 clusters we only acheve better results when looking at the first document in the ranked list.

3.1 RankBoost

When using RankBoost as our training algorithm our results see an improvement up to 2%. In figure **??**

3.2 AdaRank

3.3 LambdaRank

3.4 RankNet

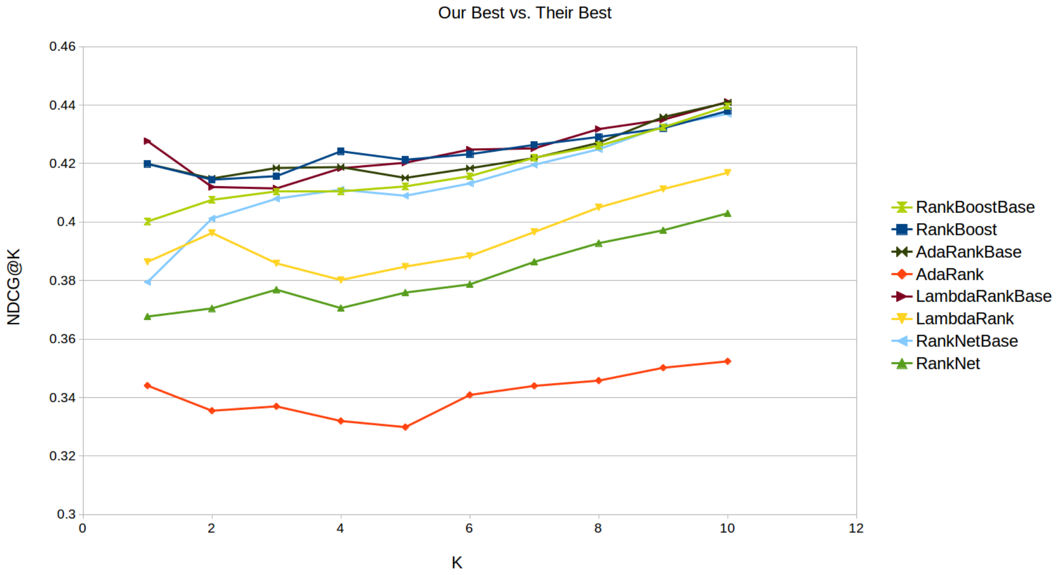
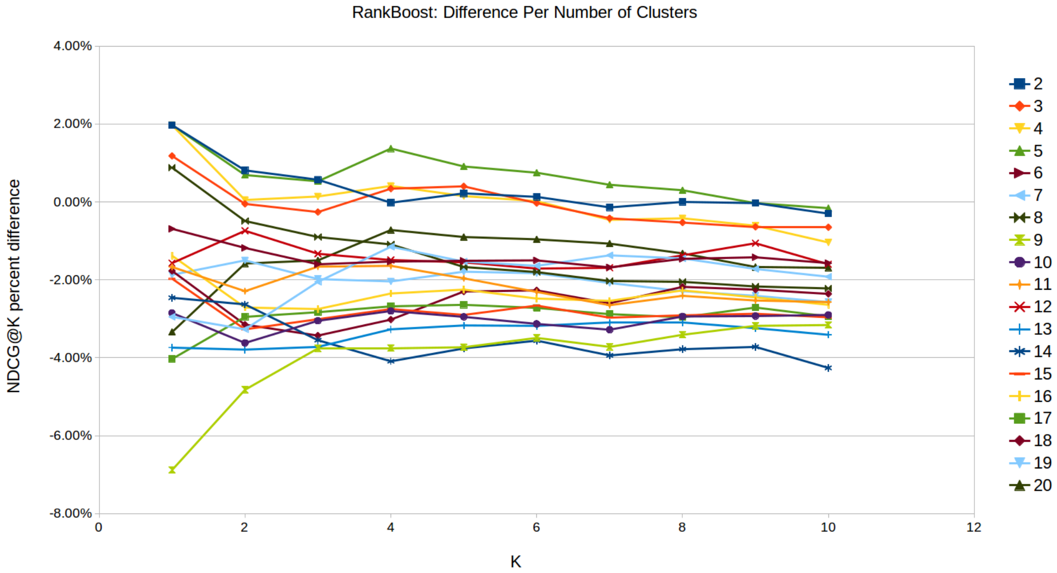
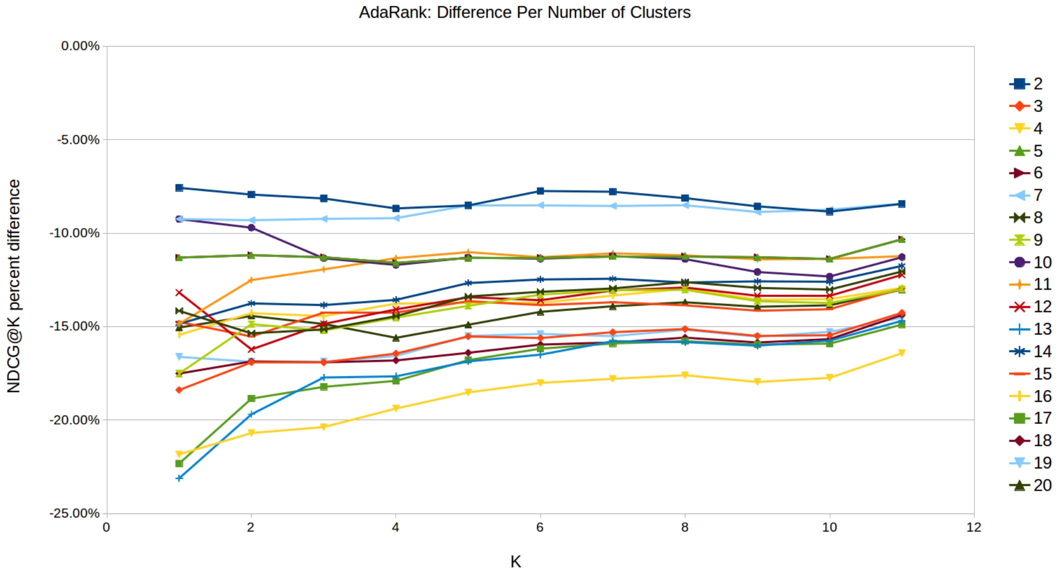
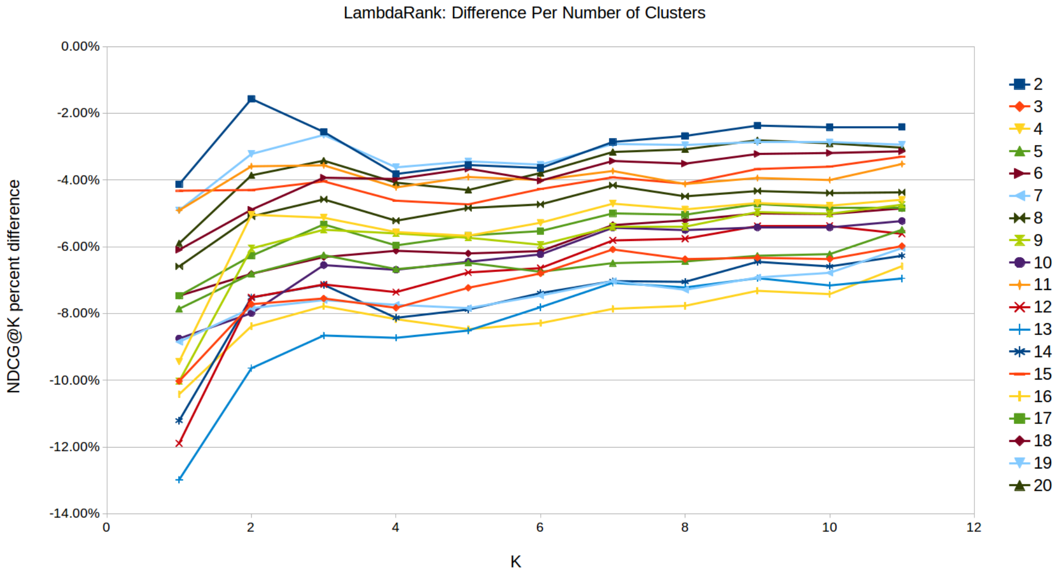
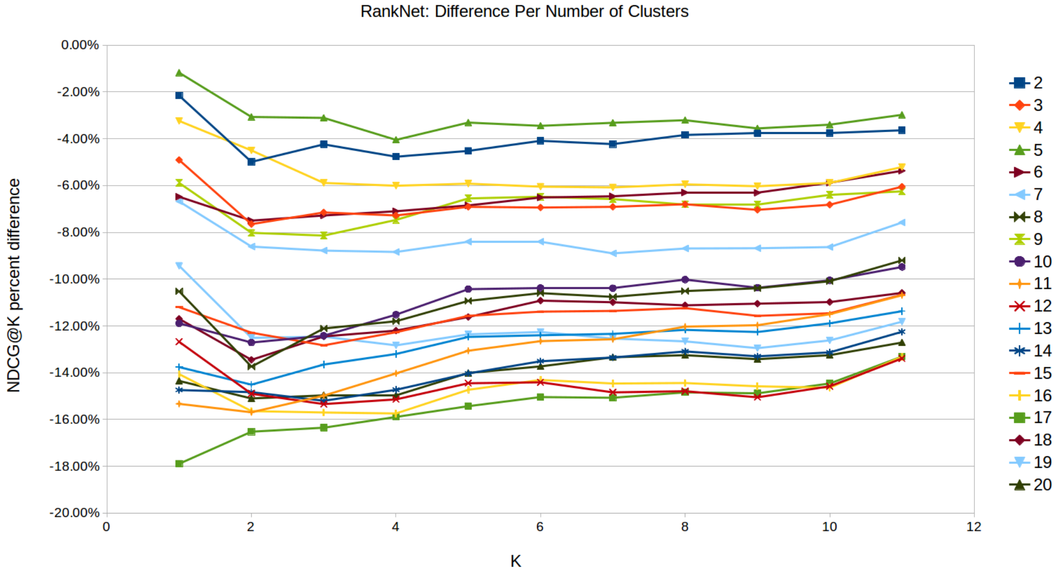


Figure 1: Best of each base algorithm along with best results we can get using our method.




AdaRank: Difference Per Number of Clusters
NDCG@K percent difference
0.00%
-5.00%
-10.00%
-15.00%
-20.00%
-25.00%
0
2
4
6
8
10
12
K
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20


LambdaRank: Difference Per Number of Clusters
NDCG@K percent difference
0.00%
-2.00%
-4.00%
-6.00%
-8.00%
-10.00%
-12.00%
-14.00%
0
2
4
6
8
10
12
K
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20


RankNet: Difference Per Number of Clusters
NDCG@K percent difference
0.00%
-2.00%
-4.00%
-6.00%
-8.00%
-10.00%
-12.00%
-14.00%
-16.00%
-18.00%
-20.00%
0
2
4
6
8
10
12
K
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20