LIU Xi

xiliu1

Homework 5

Collaboration and Originality

1. Did you receive help of any kind from anyone in developing your software for this assignment (Yes or No)? It is not necessary to describe discussions with the instructor or TAs.
   No.

   If you answered Yes, provide the name(s) of anyone who provided help, and describe the type of help that you received.

2. Did you give help of any kind to anyone in developing their software for this assignment (Yes or No)?
   No.

   If you answered Yes, provide the name(s) of anyone that you helped, and describe the type of help that you provided.

3. Are you the author of every line of source code submitted for this assignment (Yes or No)? It is not necessary to mention software provided by the instructor.
   Yes.
   If you answered No:
   a. identify the software that you did not write,
   b. explain where it came from, and
   c. explain why you used it.

4. Are you the author of every word of your report (Yes or No)?
   Yes.
   If you answered No:
   a. identify the text that you did not write,
   b. explain where it came from, and
   c. explain why you used it.

LIU Xi

xiliu1

Homework 5

1 Experiment: Baselines

Provide information about the effectiveness of your system in three baseline configurations.

	BM25	Indri BOW	Indri SDM
P@10	0.3000	0.2300	0.3700
P@20	0.2950	0.2800	0.3750
P@30	0.2967	0.2900	0.3633
MAP	0.1304	0.1277	0.1878

BM25: k_1=1.2, b=0.75, k_3=0.0

Indri:mu=2500, lambda=0.4

SDM: 0.2 AND, 0.4 NEAR, 0.4 WINDOW

2 Custom Features

Feature 1: Body Length

Information: this feature is the number of tokens in the body field, in other words, the length of major part of the document.

Computational Complexity: information is stored in TermVector already so it takes O(1) time to retrieve.

Intuitions: in my intuition, documents with more contents could possibly have more information to satisfy user's query intentions, that is, longer documents could be more relevant. Since it is very convenient and computationally inexpensive to retrieve, I chose it as my feature.

Feature 2: Sum of Matching Term Frequencies in Body

Information: this feature is sum of all matching term frequencies in the body field, in other words, the total number of matching terms in the major part of the document.

Computational Complexity: tf for each term is stored in TermVector already so for a document contains n terms, it takes at most O(n) time to retrieve all tfs.

Intuitions: in my intuition, documents with more matching terms which appears more frequently could possibly match user's query intentions better, that is, documents of larger sum of matching term frequencies could be more relevant. Since it is convenient and not very computationally expensive to retrieve, I chose it as my feature.

3 Experiment: Learning to Rank

Use your learning-to-rank software to train four models that use different groups of features.

	Baseline			**LeToRank**			
	BM25	**Indri BOW**	**Indri SDM**	**IR Fusion**	**Content-Based**	**Base**	**All**
P@10	0.3000	0.2300	0.3700	0.4520	0.4720	0.4160	0.3960
P@20	0.2950	0.2800	0.3750	0.4480	0.4460	0.4020	0.4100
P@30	0.2967	0.2900	0.3633	0.4240	0.4280	0.4027	0.4227
MAP	0.1304	0.1277	0.1878	0.2500	0.2494	0.2335	0.2254

Discuss the trends that you observe

IR fusion and content-based achieved best performance, and outperforms using all features and feature 1 – 16.

Whether the learned retrieval models behaved as you expected.

Yes. Because IR fusion combines BM25 and Indri scores in different fields, could best illustrates the degree of matching. Content-based features include term overlap ratios, and it shows how much the query intentions are satisfied. So this strategy is expected to perform well too. As for the other two approaches, I think it should perform less well because feature selections have not been applied to them.

How the learned retrieval models compare to the baseline methods

Compared with the baseline, generally speaking, LeToRank outperforms BM25 BOW, Indri BOW, and Indri SDM, in terms of top document precision as well as recall, even without feature selection.

Any other observations that you may have.

First of all, LeToRank could significantly improve the precision and recall. This is because it could found most influential characteristics for relevance with machine learning approaches. However, it does not means more features will lead to better results. For LeToRank, good feature selections could improve the learning result, in order to achieve better performance in predicting relevant documents and further more improve the precision and recall for retrieval.

Also, discuss the effectiveness of your custom features. This should be a separate discussion, and it should be more insightful than "They improved P@10 by 5%". Discuss the effect on your retrieval

experiments, and if there is variation in the metrics that are affected (e.g., P@k, MAP), how those variations compared to your expectations.

Comparing the model using all features with the model using feature 1-16, we could notice that the top document precision and recall are lower, and could conclude that at this stage, my custom features do not bring improvement to ranking predictions.

This is probably because feature selection has not been applied, and it is still potential to help if we combine it with some other features.

4 Experiment: Features

Experiment with four different combinations of features.

	All (Baseline)	**$Comb_1$**	**$Comb_2$**	**$Comb_3$**	**$Comb_4$**
P@10	0.3960	0.4520	0.4120	0.3920	0.4480
P@20	0.4100	0.4460	0.4220	0.4000	0.4360
P@30	0.4227	0.4253	0.4293	0.4267	0.4147
MAP	0.2254	0.2498	0.2288	0.2248	0.2438

Describe each of your feature combinations, including its computational complexity. Explain the intuitions behind your choices. This does not need to be a lengthy discussion, but you need to convince us that your combinations are investigating interesting hypotheses about what delivers good search accuracy.

Comb1: BM25-based f5, 8, 11, 14

Description: it calculated all BM25 scores for all field of each document.

Computational Complexity: it does not take additional computations, compared with the baseline.

Intuitions: I think since BM25 BOW works better than Indri BOW in the baseline, only selecting BM25 could possibly works better than IR Fusion. Also, it is computational inexpensive. Thus I chose this combination.

Comb2: Indri-based f6, 9, 12, 15

Description: it calculated all Indri scores for all field of each document.

Computational Complexity: it does not take additional computations, compared with the baseline.

Intuitions: I think only selecting Indri could possibly works better than IR Fusion because BM25 and Indri could possibly disagree with each other on the ranking for a certain document. Also, it is computational inexpensive. Thus I chose this combination.

Comb3: Body-based f1, 5, 6, 7, 17, 18

Description: it selects all features related to body field of each document.

Computational Complexity: it calculated the body length and sum of frequencies of matching terms. It takes at most O(n) time for a document containing n terms, but the increase in computation costs is not significant compared with the time spent on file I/O and SVM training.

Intuitions: I think body contains most information of a document and could provide sufficient evidences for ranking and avoid harms from meaningless terms in other fields. Also, it is computational inexpensive. Thus I chose this combination.

Comb4: URL-based f2, 3, 4, 11, 12, 13, 14, 15, 16

Description: it selects all features related to URL, PageRank and inlinks for each document.

Computational Complexity: it does not take additional computations, compared with the baseline.

Intuitions: I think URL and inlinks contains some keywords and provide strong evidences for ranking. Also, other information about its URL and links, such as URL depth and Wiki score could provide complementary information in addition to matching terms. Also, it is computational inexpensive. Thus I chose this combination.

Were you able to get good effectiveness from a smaller set of features, or is the best result obtained by using all of the features? Why?

Yes, all combinations except comb3 outperforms the baseline, in terms of top document precision and recall. The comb1 (BM25 scores) and comb4(URL-based) is as good as IR Fusion.

Using a selection of features could improve the performance. This is because selecting most representative features could produce a more accurate estimation in the training process, so that the prediction on ranking could be more accurate, thus the ranking result could be more precise. Also, features of low quality could bring in noises or negative effects on the training process, while features of high quality could help to estimate relevance better.

5 Analysis

Examine the model files produced by SVM^{rank}. Discuss which features appear to be more useful and which features appear to be less useful.

I looked into the weights of SVM model, trained by all features, and list the features in descending order according to their weights. I believe with a normalized dataset, a larger absolute weight indicates stronger influence on the result.

BM25-body > Spam Score> Term Overlap – title > Term Overlap – body> Indri – body > Wiki Score > Body Length>Term Overlap – URL > Term Overlap – inlink> BM25-title> BM25-inlink>Indri-URL>Indri-inlink>tfSum> PageRank>Indri-title>Indri-inlink*>URL depth**

**negative*

I think BM25 score on body field, which weights is the most useful feature, then the other 5 most important features are: Spam Score, Term Overlap in title, Term Overlap in body, Indri Score for body, and Wiki score.

The least helpful feature is Indri Score for inlink, then Indri Score for title and PageRank score.

Support your observations with evidence from your experiments. Keep in mind that some of the features are highly correlated, which may affect the weights that were learned for those features.

To experiment with my assumption, I disabled the top 5 useful features, and retrained the model and measured its performance, with the same queries and corpus.

	All (Baseline)	Remove all 5	Remove Top 1	Remove Top 2	Remove Top 3	Remove Top 4
P@10	0.3960	0.3600	0.3920	0.3560	0.3640	0.3600
P@20	0.4100	0.3940	0.4040	0.3860	0.3860	0.3860
P@30	0.4227	0.4133	0.4147	0.4120	0.4093	0.4093
MAP	0.2254	0.2251	0.2260	0.2225	0.2235	0.2244

Removing top 5 useful features harms the precision and recall. What is more, this group of experiments shows a general trend that removing useful features could introduce more harm than removing them together with less helpful features. This probably because some features are correlated, and leaving one or two alone could not help estimate the result.

Then I disabled the top 3 least useful features, and retrained the model and measured its performance, with the same queries and corpus.

	All (Baseline)	Remove all 3	Remove Top 1	Remove Top 2
P@10	0.3960	0.3840	0.3840	0.3840
P@20	0.4100	0.4160	0.4160	0.4160
P@30	0.4227	0.4187	0.4187	0.4173
MAP	0.2254	0.2250	0.2250	0.2251

Removing top 5 useful features did not bring large difference to the precision and recall, this is because the baseline model did not extract much information from them.

Some of this discussion may overlap with your discussion of your experiments. However, in this section we are primarily interested in what information, if anything, you can get from the SVM^{rank} model files.

I also assumed feature of negative weights could harm the prediction, and further more harms the precision and recall for the retrieval. So I experimented this idea by removing all features of

negative weights, to be more specific, URL depth, Indri for inlink, and Indri for title, and get exactly same performance as the baseline.

I also tried to experiment with a better selection of features, by selecting top 10 useful features but the result is worse than baseline. This is probably because some features are correlated, and leaving one or two alone could not help estimate the result. Due to limited time and resource, I could not find the correlations at current stage, but I believe by incrementally adding features to top useful features, we could gradually find the best combination to train a better model, in order to produce a more precise ranking.