Your Name Chao-Hung Chen

Your Andrew ID chaohunc

Report

1 Experiment: Baselines

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

	BM25	Indri BOW	Indri SDM
P@10	0.4600	0.2720	0.2880
P@20	0.4160	0.3420	0.3520
P@30	0.4160	0.3573	0.3627
MAP	0.2417	0.1964	0.2010

Document the parameter settings that were used to obtain these results.

Indri:mu=2500
Indri:lambda=0.4
BM25:k_1=1.2
BM25:b=0.75
BM25:k_3=1

2 Custom Features

Describe each of your custom features, including what information it uses and 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 features are reasonable hypotheses about what improves search accuracy, and not too computationally expensive to be practical.

To find the proper features, I tried to search some ideas from previous homework, and finally find that the performance of Ranked Boolean model is not that bad, especially with AND operators. Although it seems Ranked Boolean model is worse than BM25 and Indri from homework 2, however, it's because of one of the disadvantage of Boolean model, which only finds documents containing "all" query terms. That is, if we had a query with five terms, the relevant document could still be discarded even if the relevant document contains four query terms. Therefore, my hypothesis is that we might get bad result if we only used Ranked Boolean model to retrieve relevant documents, nevertheless, if we just consider it as one of the feature in Learning To Rank Model (LETOR), it may be more helpful. Besides, the other reason to use this one is that computational complexity is very convenient. It just needs to match terms, and it's

faster than BM25 and Indri. Therefore, the first customer feature I used is the AND operator in Ranked Boolean model.

For second features, I tried to use TF-IDF divides the document length as the other customer feature. One of the reason I used TF-IDF is because it's also a very popular feature in information retrieval. Although we also used BM25 in LETOR, which also shared some similar concepts of TF-IDF, one of the issues in BM25 is that it had many parameters to be determined. Therefore, I would like to try using a pure TF-IDF to see if we might also have good results without tuning parameters. However, if the document length is longer, it is also had number of terms appear in documents. Therefore, finally I decided to keep the idea of TF-IDF and tried just divided them to the document length directly for normalization.

3 Experiment: Learning to Rank

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

	IR Fusion	Content-Based	Base	All
P@10	0.4560	0.4680	0.4880	0.4840
P@20	0.4280	0.4420	0.4620	0.4620
P@30	0.4307	0.4173	0.4560	0.4613
MAP	0.2615	0.2628	0.2685	0.2662

Discuss the trends that you observe; whether the learned retrieval models behaved as you expected; how the learned retrieval models compare to the baseline methods; and any other observations that you may have.

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.

By comparing the learned retrieval models with baseline model, what we could observe is that the model with LETOR is better than the baseline model without LETOR. As we could see, even if we use model with lowest MAP (IR Fusion) in LETOR for comparison, it is still better than the best model (BM25) in baseline model (MAP: 0.2615 > 0.2417). This result is just as we expected since LETOR is an ensemble model using the scores from several features.

For the comparison of different LETOR models, at first I guess that when we increase the numbers of features, basically we could get higher MAP. My assumption behind this is that since LETOR could decide whether to use the feature or not, it might be better to put more features in the LETOR model. From the experiment result, at first it seems reasonable. However, one of the interesting result out of my

expectation is that the model in Baseline (with 16 features) had higher MAP, P@10, P@30 than the model in All (with same 16 features and 2 extra customer feature)!

I think one of the good explanations for the result is that my machine learning model (LETOR) is overfitting on training data, which causes that even though we use two customize feature to have a better fit with training data, we still didn't get better result. This other possible explanation is that my two new customer features might not have good performance, or maybe the two features are useless. To discuss further information about this situation, I ran the model separately for two custom features. Surprisingly, the performance of my first custom feature (Ranked Boolean Model with AND) is pretty good (MAP:0.2433) than most features, however, the second custom feature is bad (MAP:0.2171). The weight in the model file also shows similar result, which states that Ranked Boolean Model with AND is very useful (3rd place in useful features list), whereas the second custom feature is useless (2rd place in useless features list). From this experiment result, we are pretty sure that the model uses the first feature with high weights. Therefore, we could conclude that it's highly possible that the LETOR overfits the result because LETOR does really use the weight of the first custom feature a lot.

4 Experiment: Features

Experiment with four different combinations of features.

	All (Baseline)	**$Comb_1$** (f1,f2,f,3,f4)	**$Comb_2$** (f5,f6,f7)	**$Comb_3$** (f6,f9,f12,f15)	**$Comb_4$** (f6,f7,f10,f11,f12, f13,f15,f16)
P@10	0.4840	0.4240	0.4600	0.4160	0.4520
P@20	0.4620	0.4240	0.4200	0.4260	0.4260
P@30	0.4613	0.4267	0.4200	0.4267	0.4293
MAP	0.2662	0.2384	0.2421	0.2510	0.2608

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. 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?

Computational Complexity (from slow to fast): $Comb_4 > Comb_3 > Comb_2 > Comb_1$

Intuitions of my choices and describe feature combinations:

For $Comb_1$, I want to compare external source features (f1,f,2,f3,f4) with content-based features (f5-f16) to see if we could use those external source features as the smaller candidate set of features to get good result. However, the result shows that the MAP is not pretty good comparing than those model using content-based features. Therefore, I decided to abandon those external source features (f1,f2,f3,f4) and pick some small other sample of content-based features.

At first I tried to use $Comb_2$ to build a small candidate sets, which using "body" attribute for documents with model of BM25, Indri, and Term overlaps (f5,f6,f7) , because I am guessing that the "body" attribute includes the most information of documents. The performance this time is a little bit better than $Comb_1$ with only use three features.

The concept of $Comb_3$ is from experiment 1, where I found that BM25 had a very good performance. As a result, I tried to only use the four features related to BM25 model in different attributes (f6,f9,f12,f15). The experiment result is really impressive, which gets around 0.2510 of MAP, which is better than $Comb_2$ especially with only using four features.

For $Comb_4$, after finding BM25 model is very useful, and then I tried to increase number of features based on $Comb_3$. I found that since BM25 models also had good performance, another popular model Indri may also had good results. Therefore, I tried to increase the number of features from original features sets with the four other features with Indri models. The experiment result of show that $Comb_4$ has the best MAP among the other 3 Comb and it's more close to the Baseline models.

From the experiment result, we could still find that Baseline model, which used 16 features, is still getting better result (effectiveness) than the other Combinations, which used 4 ~ 8 features. The trend we could find is that as long as the model is not overfitting, we could get more accurate result with more features in most cases. The reason of the result is that with more feature sets, our machine learning models could learn whether to pick the feature or not. On the other hand, if we want to use only small sets of features due to computational cost, we should also focus on feature selections. From the experiment, we could find that $Comb_2$ with 3 features had better result than $Comb_1$ with 4 features.

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. 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.

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.

For this experiment, I used all features (including custom features) to learn models for SVM^{rank}. The result shows that the top 3 useful features are BM25 for "body" attribute, spam scores for documents, and RankedBoolean model with "AND" operator, which is the custom feature. On the other hand, the top 3 less useful features are Indri with "inlink", TF-IDF divides document length, and PageRank for a document.

Discussion:

I think it' not surprised that BM25 with "body" attribute could be very useful since it already had nice performance (MAP=0.2417) by itself. BM25 is a pretty nice probabilistic model using the concept of TF-IDF, and reformulate it by tuning with more parameters (k1,k3,mu) . With this pretty good result for BM25, no wonders that BM25 is still a very popular model in information retrieval.

The result also shows that spam scores of document also had high weights. I think this feature is very important because it's provided us an external source to know whether the document is spam. In most cases, most relevant documents are not spams. However, if we only used content data in documents, sometimes it's very difficult to know if the document is spam or not because in real world some spams document will fake themselves as not spam by making themselves had similar terms structure with non-spam documents. If we had external resources outside of documents, this will be easier for us to know whether the document is spam.

RankedBoolean model with "AND" operator is at the third place for top 3 useful feature. I believed one of the reasons is that it could classify whether a document contain all query terms. For BM25 and Indri, sometimes they will retrieve the documents only contain part of query terms because those appearing terms had super high scores, higher than some documents with sum of scores containing all query terms. However, in most cases, it's better for relevant documents containing all query terms, instead of just part of them. Therefore, RankedBooleand model with "AND" operator played a very important role at here to use its "AND" operator and check if the document had all query terms.

On the other hand, the top 3 less useful features are Indri with "inlink", TF-IDF divides document length, and PageRank for a document. From the experiment, would find that not only Indri with "inlink" attribute, but also BM25(f14) and term overlap(f16) with "inlink" attribute are also not pretty good. One of the possible reasons might be the importance of "inlink" could decrease because attribute with "title","url",and "body" might all have high correlation with "inlink". To discuss this with more details, I re-run the experiment by only use Indri, BM25, and term overlap (f14,f,15,f16) attributes with "inlink". In fact, the result is pretty good (MAP=0.2464). Therefore, we could states that "inlink" might highly correlated with other attributes, and the important (weight) of the "inlink" attribute decreases.

For the custom feature TF-IDF divides document length, the experiment result also shows this feature is less useful. This result proved that BM25 is better than just only using TF-IDF since

they had high correlation, and it also demonstrated that the default parameters in BM25 are well tuned.

For PageRank, it also didn't have many effects for the models. This result is out of my expectation. In general, we might think that PageRank is useful for discovering relevant documents. Although this assumption might be corrected, it seems that in this model we also use some other features (like spam scores) highly correlated with PageRank feature. Therefore, the important of PageRank might decreases in this model.

In summary, we could discover that LETOR with SVM^{rank} is very useful in improving the performance. The key idea is that it provided a nice machine learning model to ensemble different features to learn a better model. For some features, which could get decent performance before, they now might have lower weights in LETOR model since they might be high correlation with the other features, which had better performance than those decent performance features.