

Notes on Selected Topics of CS410

ChengXiang Zhai

March 13, 2008

1 Normalized Discounted Cumulative Gain (NDCG)

MAP is a commonly used single number measure to evaluate a ranked list of search results when a document is judged as either relevant or non-relevant (i.e., binary relevance judgment). However, since relevance is a matter of degree, we would often like to judge the relevance of a document in multiple levels. For example, we may distinguish 4 levels: (1) level 1: non-relevant; (2) level 2: borderline; (3) level 3: relevant; (4) level 4: highly relevant. In such a case, how do we evaluate search results? This is what NDCG is designed for.

NDCG stands for Normalized Discounted Cumulative Gain. This measure is defined at a given rank (e.g., rank n), and the basic idea is to compute the “gain” of a user seeing the top n documents. Intuitively, this “gain” is related to the sum of the gain from each of the n documents. We define the gain of each document as its relevance level. For example, a highly relevant document may have a gain of 3, and a non-relevant document may have a gain of 0. In general, suppose we have a ranked list of results d_1, \dots, d_n in the order of ranking (i.e., d_1 is the top-ranked document). We can denote the gain of d_i by r_i (rating of document d_i).

The cumulative gain of the result is defined as

$$CG(d_1, \dots, d_n) = \sum_{i=1}^n r_i$$

In CG , each of the n documents is treated equally. For example, a highly relevant document would always contribute the same value (e.g., 3) no matter where it is ranked. Intuitively, a top-ranked highly relevant document is more valuable than a lowly ranked one. Thus we introduce a discounting coefficient to discount the gain of lowly ranked documents. Specifically, the Discounted Cumulative Gain is defined as:

$$DCG(d_1, \dots, d_n) = r_1 + \sum_{i=2}^n \frac{r_i}{\log_2 i}$$

When we evaluate a retrieval function, we generally need to take the average of DCGs over a set of topics/queries. Unfortunately, DCGs for different topics generally have different ranges of values. Specifically, a topic with fewer relevant documents (e.g., just two relevant documents) will have a smaller range of values of DCG than a topic with many relevant documents (e.g., 100). For example, if a topic has just two relevant documents and each relevant document has a rating of “2”, then the maximum DCG at rank 10 one can possibly achieve would be $2 + 2/\log_2 2 = 4$. This is clearly much smaller than what one can achieve when the topic has more than 10 relevant documents. This causes a problem in averaging over these DCG values because a topic with more relevant documents would dominate the average value.

To solve this problem, NDCG introduces a normalization step. The idea is to compute the maximum DCG at rank n one can possibly achieve. This value can be obtained by assuming the ideal ranking of results for the topic, i.e., putting the documents with the highest ratings on the top. Let us assume that the ratings of the ideal ranking are R_1, \dots, R_n . Then the maximum DCG would be:

$$MaxDCG = R_1 + \sum_{i=2}^n \frac{R_i}{\log_2 i}$$

And the NDCG value of our original results d_1, \dots, d_n would be

$$NDCG(d_1, \dots, d_n) = DCG(d_1, \dots, d_n) / MaxDCG$$

We do such normalization for every topic and then take the average value over all the topics as the final NDCG value for all the results. This value can then be used to compare two different ranking methods.

2 Beta-Gamma Thresholding

Beta-Gamma Thresholding (BGT) is a method for setting the threshold in adaptive information filtering (i.e., content-based filtering). The method is a special case of the general “empirical utility optimization” method for threshold setting.

In adaptive filtering, at any time, we may assume that all the past documents the system has seen are available for us as training data. We also assume that the user would give us feedback on the delivered documents. Thus we will have a small number of judged documents. The general idea of empirical utility optimization is to simply rank all these training documents with the current profile (query) vector and then try out each possible cutoff (i.e., first try rank 1, then try rank 2, etc.) as a potential threshold point. For each possible cutoff, we could obtain a score threshold as well as a utility value computed based on the assumption that the documents above the threshold will be delivered to the user and all unjudged documents are non-relevant. If we try all the possible cutoffs, we can find a threshold that gives the best utility, which we can take as our optimal threshold (θ_{opt}).

Unfortunately, this basic empirical utility optimization method has some problems. First, it does not accommodate exploration. If for some reason, we accidentally deliver a few non-relevant documents to a user at the beginning, the system would learn to set a very high threshold. As a result, the system may essentially “shut off” the delivery of any future documents even the user may be interested in some lowly ranked documents. Thus it would be desirable to lower the threshold a bit to hopefully obtain more feedback from a user about some “near-miss” documents. We call this “exploration” (as opposed to “exploitation”). Second, since the unjudged documents are all assumed to be non-relevant, the computed utility value is an underestimate of the real utility value at that cutoff. Besides, the vector is typically updated by adding terms from the judged relevant documents. As a result, the relevant documents in the training data would have a much higher score than an unseen average relevant document because the training documents have directly contributed terms to the query/profile vector. This is another reason why we may need to lower the computed optimal threshold a bit. We call this “bias”.

The idea of BGT is to address these two needs for lowering the threshold. Specifically, it would allow the system to lower the threshold until the utility becomes negative. That is, it goes down the ranked list to find another threshold point (θ_{zero}) where the utility value is zero and interpolates the two threshold values. That is,

$$\theta = (1 - \alpha)\theta_{opt} + \alpha\theta_{zero}$$

where α controls the amount of deviation from the optimal threshold.

To address the two different needs for lowering the threshold mentioned above, it further decomposes α into two terms:

$$\alpha = \beta + (1 - \beta)e^{-N\gamma}$$

where N is the number of judged documents the system has seen so far and β and γ are two parameters.

β is introduced to address the bias and its value will have to be empirically set. γ is to control the amount of exploration. The idea of the second term in the equation above is that as we see more and more judged documents, we would stay closer and closer to θ_{opt} because more judged documents indicate that it's more likely that we've already had sufficient exploration. Eventually, as N gets very big, the second term essentially disappears. γ controls the amount of exploration before the second term "disappears". The larger γ is, the more quickly, the second term would disappear, thus less exploration.