Proximity-based Rocchio's Model for Pseudo Relevance Feedback

Sai Venkata Maneesh Tipirineni
UMass Amherst
stipirineni@cs.umass.edu

Bhuvana Sai Surapaneni
UMass Amherst
bsurapaneni@cs.umass.edu

ABSTRACT

Rocchio's model for pseudo relevance feedback is a classic query expansion method. In traditional Rocchio's model, user's search query is revised to include potentially related terms from relevant feedback documents, to improve the search engine's precision and recall. But the term associations between the candidate expansion terms and query terms are ignored. Incorporating term proximity information into the traditional Rocchio model has been proved to give promising results. This paper verified the proximity based Rocchio's model, called PRoc, proposed by Miao et al. [1]. Experiment results on TREC, Robust04, WT10G collections show that these PROC models are superior to other relevance feedback models like RM3, DMM etc.

Keywords

Pseudo relevance feedback, Query Expansion, Rocchio's Model, Term Proximity, Proximity Term Frequency

1. INTRODUCTION

Pseudo relevance feedback automates the manual part of relevance feedback, so that the user gets improved retrieval performance without extended interaction. Relevance information is utilized by using the contents of the relevant documents to either adjust the weights of terms in the original query, or by using those contents to add words to the query. Relevance feedback is often implemented using the Rocchio model. It assumes that the top ranked documents in first pass are relevant documents and use them as the feedback documents to refine the query, by adding potentially relevant terms. The representation of the query is refined using the initial query and feedback documents as follows:

$$Q_1 = \alpha \cdot Q_0 + \beta \cdot \sum_{r \in R} \frac{r}{|R|} \quad (1)$$

where Q_0 and Q_1 represent the original and first iteration query vectors, r is the expansion term weight vector and α,$\beta$are tuning constants controlling how much we rely on the original query and the feedback information. In this method, the candidate terms are ranked by their term frequencies. This does not capture the relevance of the candidate term to the query topic. The terms closer to the query terms are more likely to be relevant. A term, even with highest term frequency, which occurs far away from query terms is very likely to be irrelevant. This may introduce irrelevant terms into the search query, misguiding the search process. Thus, instead of term frequency, we need to use proximity term frequency (ptf). Proximity term frequency models the frequency of the term as well as the semantics to the query in terms of proximity. Miao et al. [1] proposed three methods for computing ptf. In addition to the proximity information, importance of the query terms is taken into account by integrating inverse document frequency (idf) with ptf.

2. PROXIMITY-BASED ROCCHIO'S MODEL

In this section, we present the three proposed methods for adopting proximity information in to Rocchio's model.

2.1 Window-based Method (Proc1)

In window-based n-gram counting method, the document is segmented into a list of sliding windows, with each window having a fixed window size wSize. Each window contains wSize consecutive tokens. The n-gram frequency is defined as the number of windows in which all n-gram terms co-occur. ptf is computed as follows:

$$ptf(t) = \sum_{i=1}^{|Q|} c(t, q_i) IDF(q_i) \quad (2)$$

Where q_i is a query term, C(t, q_i) is the number of windows in which the candidate term and the query terms co-occur, |Q| is the number of query terms and IDF(q_i) equals to log(N $- N_t + 0.5)/(N_t + 0.5)$. N is the number of documents in the collection, and N_t is the number of documents that contain q_i. PRoc1 does not take actual distance between candidate term and expansion term into account.

2.2 Kernel-based method (Proc2)

In this method, gaussian kernel is used for measuring term proximity between a candidate expansion term t and a query term q.

$$K(t, q) = exp[\frac{-(p_t - p_q)^2}{2\sigma^2}] \quad (3)$$

where p_t and p_q are respectively the positions of candidate term t and query term q in a document, σ is a tuning parameter which controls the scale of Gaussian distribution. It is similar to wSize. ptf is calculated in kernel based method as follows:

$$ptf(t) = \sum_{i=1}^{|Q|} K(t, q_i) IDF(q_i) \quad (4)$$

where q_i is a query term, |Q| is the number of query terms, and IDF(q_i) is the same as in PRoc1. N is the number of documents in the collection, and N_t is the number of documents that contain q_i. Proc2 is a soft proximity measure in the sense that it boosts the weight of a candidate term even if it is not close enough to a query term.

2.3 HAL Method (Proc3)

HAL method gives term weights proportional to proximity and only consider terms which co-occur in a certain window of size wSize as relevant. A HAL space is built by moving sliding windows of length wSize across the corpus. Words within the window are considered as co-occurring, with strengths inversely proportional to the distance between them. A term is represented as a semantic vector as follows:

$$HAL(t'||t) = \sum_{k=1}^{wSize} w(k)n(t,k,t') \quad (5)$$

where k is the distance from term t to t, n(t, k, t') is the co-occurrence frequency within the sliding windows when the distance equals k, and w(k) = wSize − k + 1. This vector denotes a proximity relationship of each candidate term with the entire query. The HAL based ptf is computed as follows:

$$ptf(t) = vec(t) \cdot vec(Q) = \sum_{i=1}^{|Q|} HAL(t||q_i)IDF(q_i) \quad (6)$$

The weighted HAL method includes the information of term distances and co-occurrence frequencies completely.

3. EXPERIMENTAL SETTING

3.1 Test Collections

We chose to test our models and baselines on three different test collections of different sizes. The TREC1-3 collection which has an uncompressed size of 2.8GB and 150 queries. The Robust04 collection which has an uncompressed size of 2.36GB and 249 queries. And the massive WT10G collection which is a medium size crawl of Web documents and has an uncompressed size of around 10GB of data and 100 queries. All words are stemmed using Krovetz English Stemmer and stopwords from Inquery english stopword list containing 418 stopwords are removed.

3.2 Baseline Models

We chose to compare our Proc models with four different state-of-the-art feedback models in order to evaluate our performance rigorously. For all the feedback models, we used Language Modelling (LM) retrieval model smoothed with Dirichlet prior smoothing (with hyperparameter μ) as shown in equation below.

$$p(w|d) = \frac{c(w_d) + \mu p(w|C)}{|d| + \mu} \quad (7)$$

where c(w_d) is the frequency of query term w in document d, p(w|C) is the probability of term w in collection C and |d| is the length of document d.

3.2.1 RM3

RM3 attempts to generate a generalized notion of relevance R basically inferring that R generates both the query Q and the set of relevant documents θ_D. RM3 is an interpolated version of RM1 model. The formula for RM1 is:

$$p(w|R) \propto \sum_{\theta_D} p(w|\theta_D)p(\theta_D)P(Q|\theta_D) \quad (8)$$

where the relevance model p(w|R) is often used to estimate the feedback language model θ_F , and then interpolated with the original query model θ_Q . This is because sometimes the original query terms might not have the highest weights in the expanded feedback model which might affect the performance in a negative way. The following is the formula for RM3 and RM4(mentioned below):

$$\theta_{Q'} = (1-\alpha) * \theta_Q + \alpha * \theta_F \quad (9)$$

where α is the parameter that controls the weight of the feedback words. The Dirichlet smoothing method(shown above) is used to smooth the language model of each pseudo-relevant document θ_D in both RM3 and RM4 (mentioned below).

3.2.2 RM4

RM4 is a slightly involved version of RM3 and is an interpolated version of RM2. The formula for RM2 is shown below:

$$P(t|q,R) \propto \left(\sum_{D_j} P(t|D_j)\right) \cdot \prod_{q_i \in q} \sum_{D \in \{D_R\}} P(q_i|D) \frac{P(t|D)}{\sum_{D_j} P(t|D_j)} \quad (10)$$

where t is the candidate expansion term and the relevance model p(t|q,R) is often used to estimate the feedback language model θ_F , and then interpolated with the original query model θ_D as is the case with RM3. The same interpolation formula mentioned above is used to compute both RM3 and RM4.

3.2.3 Divergence Minimization Model (DMM)

The DMM model attempts to cast the estimation of the feedback model as an optimization problem by assuming that the feedback model θ_F should be very close to the language model of every pseudo-relevant document but far away from the collection language model. The following is formula for DMM:

$$p(w|\theta_F) \propto exp\left[\frac{1}{1-\lambda}\left(\frac{1}{|F|}\sum_{i=1}^{|F|} logp(w|\theta_i) - \lambda logp(w|C)\right)\right] \quad (11)$$

where p(w|θ_i) is smoothed using Dirichlet prior smoothing in the same way as the document language model in the retrieval step. Finally, the query language model is updated by interpolating θ_F with the original query model θ_Q in the same way as RM3 and RM4 above. Here, hyperparameter λ is called weighting parameter and it controls the whole collection i.e., corpus.

3.2.4 Simple Mixture Model (SMM)

In the SMM, the words in feedback document set are assume drawn from the background model and topic model i.e., the feedback documents. Thus the log- likelihood for the entire set of feedback documents is given by the formula:

$$logp(F|\theta_F) = \sum_{w \in V} c(w,F) log\big((1-\lambda)p(w|\theta_F) + \lambda p(w|C)\big) \quad (12)$$

where c(w,F) is the count of word w in the set of feedback documents F, and λis the probability of choosing the word from the whole corpus. θ_F models the feedback document model.Finally, the query language model is updated by interpolating θ_F with the original query model θ_Q in the same way as the baselines above.

Table 1: PRoc compares with baselines on Trec123

# of feedback terms	Metric	PRoc1	PRoc2	PRoc3	SMM	DMM	RM3	RM4
10	AP	0.271	**0.278**	0.277	0.263	0.247	0.264	0.260
	nDCG	0.527	**0.53**	0.529	0.529	0.512	0.520	0.516
	ERR	0.282	0.28	0.281	**0.284**	0.267	0.283	0.256
20	AP	0.274	0.281	**0.284**	0.270	0.256	0.273	0.264
	nDCG	0.536	0.56	**0.542**	0.533	0.526	0.539	0.514
	ERR	0.286	**0.291**	0.283	0.278	0.276	0.284	0.274
30	AP	0.280	0.285	**0.286**	0.272	0.261	0.276	0.269
	nDCG	0.537	0.565	**0.573**	0.536	0.522	0.561	0.556
	ERR	**0.284**	0.29	0.284	0.281	0.277	0.284	0.279

Table 2: PRoc compares with baselines on Robust04

# of feedback terms	Metric	PRoc1	PRoc2	PRoc3	SMM	DMM	RM3	RM4
10	AP	0.259	0.260	**0.266**	0.252	0.249	0.264	0.254
	nDCG	0.435	0.432	0.434	0.443	0.443	**0.450**	0.434
	ERR	0.322	0.323	**0.328**	0.327	**0.328**	0.320	0.324
20	AP	0.262	0.261	**0.267**	0.255	0.253	0.266	0.251
	nDCG	0.438	0.445	**0.447**	0.438	0.446	0.451	0.430
	ERR	0.328	0.325	0.333	0.320	**0.329**	0.322	0.317
30	AP	0.263	0.267	**0.271**	0.257	0.256	0.269	0.248
	nDCG	0.447	0.455	**0.458**	0.441	0.447	0.455	0.426
	ERR	0.329	0.329	**0.335**	0.322	0.328	0.320	0.324

Table 3: PRoc compares with baselines on Wt10g

# of feedback terms	Metric	PRoc1	PRoc2	PRoc3	SMM	DMM	RM3	RM4
10	AP	0.189	0.189	**0.192**	0.186	0.191	0.191	0.168
	nDCG	0.291	0.298	0.297	0.285	**0.301**	0.298	0.263
	ERR	0.301	0.302	0.306	0.303	0.305	**0.317**	0.295
20	AP	0.192	**0.198**	0.197	0.187	0.190	0.192	0.159
	nDCG	0.298	0.3	0.304	0.279	**0.305**	0.291	0.246
	ERR	0.306	0.304	**0.309**	0.295	0.308	0.308	0.263
30	AP	0.193	**0.205**	0.201	0.188	0.192	0.191	0.153
	nDCG	0.303	0.305	**0.308**	0.271	0.307	0.288	0.236
	ERR	0.304	**0.308**	**0.308**	0.292	0.310	0.306	0.250

3.3 Parameter Settings

Most of the pseudo relevance feedback retrieval models have several controlling parameters. In order to find the optimal parameter setting, we used the training method for both the Proc and Baseline methods. We performed 2-fold cross-validation. For every data set, we partitioned the queries into two sets, training and testing. The parameters learned from the training set are used on testing set for evaluation. We set the number of feedback documents as 20 in all our experiments with Proc and baseline methods. We experimented with different number of feedback terms namely, 10, 20 and 30. For the Dirichlet smoothing, we explored values of μ from 500 to 2000 with a step of 500. The value of RM3 and RM4 was empirically set to 0 in accordance to Lv and Zhai's work[2] and our previous experience (in course assignments). The α value in RM3, RM4, DMM and SMM which controls the weight of the feedback was set to the best value obtained from the set of numbers between 0 and 1 with step 0.1. The weighting parameter λ in DMM is set to the best value between 0 and 1 with a step of 0.1. We swept over values from 10 to 100 with a step of 10. Finally, the α value in the traditional Rocchio model is set to 1 and β value which controls the contribution of the feedback to the new query formulation was swept for in the range of 0.1 to 1 with step 0.1.

4. COMPARISON AND RESULTS

The results of the baseline methods are obtained after 2-fold cross-validation. We used three different measures of effectiveness to evaluate our models and baselines: MAP taking the top 1000 documents into account, nDCG and ERR taking top 10 documents into account. We feel that by evaluating the models against these three different measures would give a detailed perspective of how the models fare in different conditions.

5. RESULTS AND DISCUSSIONS

As we can see from the results above, RM3 performs better than all other baselines models. The other models are not far off but RM3 leads the pack across almost all scenarios. In this section, we compare the results of our proc models to the baselines across different evaluation measures. The values presented above are the average values of the measures over all the β values and wSize values discussed in the above section. All the Proc models tend to perform better than the RM4,SMM and DMM methods in most settings and comparable to RM3. Proc2 and Proc3 show better performance than RM3 while Proc1 falls short in most cases. As we can see from the above results, on the Trec123 show a marked performance improvement over the baselines. They show a performance improvement of around 3%, 5% and 5% respectively in AP. Similar improvements are shown in nDCG and ERR as well. One thing to note is that for the Trec123 dataset, the performance of all the models including baselines increases with the number of feedback terms. Overall the models show an improvement of (3.3% - 5%) considering only the best AP values across all Proc models. They show an improvement of (0.33% - 2.15%) improvement in nDCG and (0% - 5.3%) improvement in ERR considering the best values over all Proc Models. The performance improvement off all models is marginal over the baseline RM3 method. However, Proc3 performs better than all the baseline methods in all measures. But, the degree of improvement is not substantial in most cases. Just like with Trec123, even with Robust04, the values generally tend to increase with increasing number of feedback terms. The models do not perform up to expectations of the wt10g dataset. There is also a general decrease in performance over all models. Also, the effect of increasing number of feedback words is not as pronounced in this dataset as it was in the others. This might be because the dataset is massive and a small increase in the number of feedback words does not affect the performance much. Proc1 suffers very badly falling below the baseline in most cases. Proc3 manages to perform better than most baselines and is much more stable than both the other methods. In general too, even though the results of all models are close to each other, Proc3 seems to top the lot. From these results, we could say that the Proc models were able to successfully model the proximity information to an extent and are match and fare better than RM3 which is one of the robust feedback models out there. Note that most pairs have been tested to be statistically significant at $p<0.05$ on the paired t-test with some exceptions.

6. ANALYSIS AND DISCUSSION

6.1 Effect of β on Proc Models

Parameter β in the traditional Rocchio equation controls how much we rely on the feedback information. If $\beta = 0$, we do not take any feedback information. In this section, we analyze the effect on the performance of our Proc models. Figure 2 and 4 (shown above) show the performance of our Proc models with varying β on Trec123 and Robust04 datasets on three different measures. The values shown in the graphs are the average values for the measures over window sizes of 10 to 100 with a step size of 10. Also, we set the number of feedback documents empirically to be 10. We analyzed β values from 0.1 to 0.9 with a step size of 0.2. It is safe to say that there is definite increase in performance with increasing β from 0.1 to 0.9 which is in line with the work in [3] where the authors studied the effect of β on their vari-

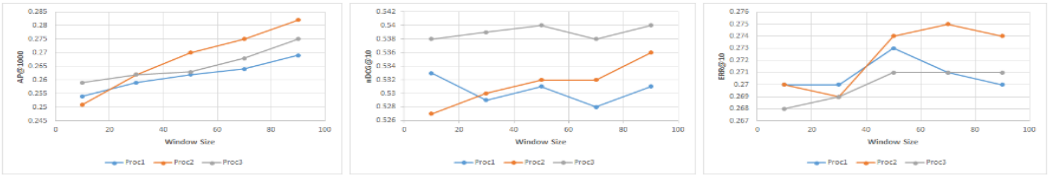

Figure 1. AP, nDCG and ERR values for Proc1(blue), Proc2(orange), Proc3(grey) over Trec1-3 with varying wsize

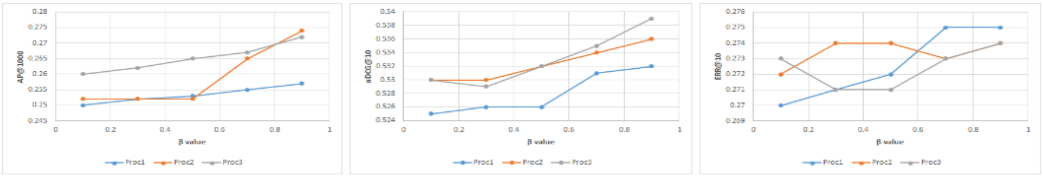

Figure 2. AP, nDCG and ERR values for Proc1(blue), Proc2(orange), Proc3(grey) over Trec1-3 with varying β

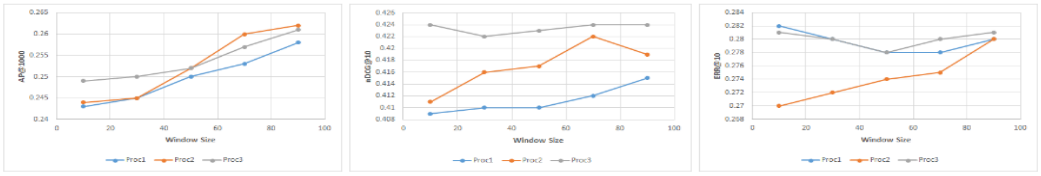

Figure 3. AP, nDCG and ERR values for Proc1(blue), Proc2(orange), Proc3(grey) over Robust04 with varying wsize

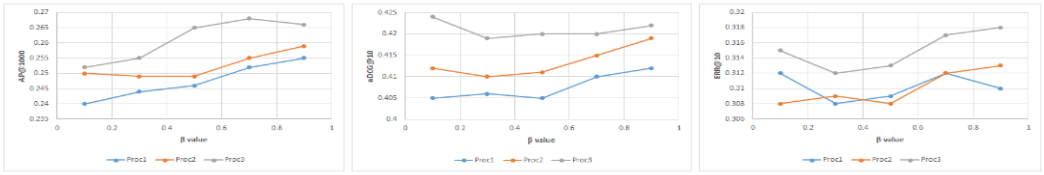

Figure 4. AP, nDCG and ERR values for Proc1(blue), Proc2(orange), Proc3(grey) over Robust04 with varying β

ation of Rocchio model and the traditional Rocchio model. As we can see from above, all Proc models show increase in effectiveness as β increases with some exceptions. One recurring pattern we observed was that there was decrease in performance for a brief period followed by an increase in performance later on. This might be hapeening because the weight of the feedback terms might not be significant to affect the query in a meaningful way. This phenomenon is not observed in AP values but only nDCG and ERR. The ranking might be changing due to the noise introduced by the terms. The AP performance does not decrease because of this noise as it is more recall oriented than the other two measures and small changes in the order of ranking might not affect it as much as the other two. We also found out in our experiments that the performance drops down drastically (by about 50% for β =5) when the value of β is increased to higher values and keeps decreasing from 1. So, setting β value close to 1 might give optimal results.

6.2 Effect of Window Size on Proc Models

wSize parameter determines the maximum distance at which two words are considered to be related and hence is crucial to the performance of our Proc models. Figure 1 and 3 (shown above) show the performance of our Proc models with varying window sizes on Trec123 and Robust04 datasets on three different measures. The values shown in the graphs are the average values for the measures over β from 0.1 to 0.9 with a step size of 0.2. As above, we empirically set the number of feedback documents to be 10. And we analyzed the window size from 10 to 90 with a step size of 20. For the sake of analysis, we assumed wsize in Proc2 to be equal to hyperparameter σ. In general, from our experiments we observed that the graphs with feedback terms 20 and 30 also behaved similarly as the ones above. As we can see from above, the values generally increase with increasing window size. This shouldn't come as a surprise because increasing the window size increases the probability of finding relevant information. But we should be careful with increasing wSize to bigger values as this introduces noise and brings us back to the initial problem of the feedback information not being related enough to the query. This problem did not occur with Trec123 and Robust04 datasets but the optimal value for wSize in the wt10g dataset was 10. We infer that there might be some kind of relationship between window size and size of corpus. Moreover, Proc1 and Proc2 models are relatively affected more by variation in window size compared to Proc3. Proc3 tends to be stable w.r.t wSize because it's weight function tends to normalize the weights by considering wSize (in its formula). The dip in nDCG and ERR values at the end of the graphs can be attributed to the same phenomenon we discussed in the above section where noise starts to creep in and rankings are disturbed. This is not evidently visible in AP at first but nDCG and ERR values tend to be more sensitive to it.

7. REFERENCES

[1] Jun Miao, Jimmy Huang, Zheng Ye. *Proximity-based rocchio's model for pseudo relevance.* [*In CIKM '09: pages 1895-1898, New York, NY, USA, 2009. ACM*].

[2] Yuanhua Lv and ChengXiang Zhai. *A comparative study of methods for estimating query language models with pseudo feedback.* [*In SIGIR '12: Pages 535-544, Portland, Oregon, USA, 2012. ACM*].

[3] Zheng Ye, Ben He, Xiangji Huang, and Hongfei Lin. *Revisiting rocchio's relevance feedback algorithm for probabilistic models.* [*pages 151-161. AIRS, 2010*].