Relevance Propagation for Topic Distillation UIUC TREC-2003 Web Track Experiments

Azadeh Shakery, ChengXiang Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Abstract

In this paper, we report our experiments on the Web Track TREC-2003. We submitted five runs for the topic distillation task. Our goal was to evaluate the standard language modeling algorithms for topic distillation, as well as to explore the impact of combining link and content information.

We proposed a new general relevance propagation model for combining link and content information, and explored a number of specific methods derived from the model. The experiment results show that combining link and content information generally performs better than using only content information, though the amount of improvement is sensitive to the document collection and tuning of parameters.

1 Introduction

We participated in the topic distillation task of TREC-2003 Web Track with two goals:

1. Evaluating our basic language modeling algorithms for topic distillation.

2. Exploring how to effectively combine the link information with the content information.

The reports about this task from TREC-2002 seem to indicate that a standard content-based retrieval algorithm, such as Okapi, performs very well for topic distillation. So we decided to test how a basic language modeling approach would perform. Specifically, we used the standard query likelihood method with Dirichlet prior smoothing [8] as well as the two-stage language modeling algorithm, which is supposed to tune the parameters automatically according to the query [9].

Intuitively, the link information may provide some clues as to whether a page is a key resource or not. It is thus interesting to explore how we may combine the link information with the content information to improve the accuracy in finding key resources. We propose a new general relevance propagation model for combining link information with the content information. The relevance propagation model naturally captures the intuition that a web page's value depends on the page's content value (self relevance) as well as the values of all the pages that are linked to this page (through either inlinks or outlinks). It allows every page to propagate its self relevance value to the neighboring pages through links to generate a "hyper-relevance" value for each page.

We consider several interesting special cases of this general relevance propagation model, and derive several specific methods for combining link information with content information. We use the query likelihood method with Dirichlet prior smoothing as well as the two-stage smoothing for computing the self relevance value of a page (solely based on the content). We then apply these different propagation methods to propagate every page's self relevance value along links to obtain a hyper-relevance value for each page. The hyper-relevance values are used to produce the final ranking for selecting key resources.

The experiment results show that combining link and content information generally performs better than using only content information, though the amount of improvement is sensitive to the document collection and tuning of parameters.

The rest of this paper is organized as follows. In Section 2, we briefly describe the language modeling algorithms that we experimented with. In Section 3, we present the general Relevance Propagation Model and three specific propagation methods. In Section 4, we briefly discuss how to implement the general relevance propagation model. In Section 5, we analyze the results of our experiments on TREC-2002 and TREC-2003 data. Finally, in Section 6, we give the conclusions and future research directions.

2 Language Modeling Algorithms

Our basic content-based retrieval algorithm is the Kullback-Leibler (KL) divergence between the query language model and the document language model [1, 7]. This method is a generalization of the query likelihood retrieval method proposed in [5] and can support feedback more naturally than the query likelihood method. In this retrieval method, in order to compute a score for a document w.r.t. a query, we first compute a query language model and a document language model, and then compute

the KL-divergence of these two models. The main issue in computing the document language model is smoothing, and we explore two smoothing methods – Dirichlet prior and two-stage smoothing. We also explore two different ways of computing a query language model, one using the query text alone and one using both the query text and some pseudo feedback documents. The details of these methods can be found in [10].

From these different combinations, we decide to use the following approaches as our baseline "content only" runs:

1. **Dirichlet Prior Smoothing, no feedback**: This is the simplest language modeling approach.

2. **Two-stage smoothing, mixture model feedback**: This is a relatively sophisticated language modeling approach.

3 Relevance Propagation Model

The results of TREC-2002 Web Track did not seem to favor approaches combining link and content information. The best official results were from Tsinghua university [11]; they explored the use of out-degree, as well as anchor text to find key resources. What they found was that anchor-text was useful, but out-degree was not. The second and third best results on TREC-2002 Web Track were from City University [2] and Chinese-Academy [6] respectively. None of these groups used the link information in finding the key resources.

However, intuitively, the content similarity of a page to a query, on its own, may not be sufficient for selecting a key resource, and the link information can be useful in finding key resources. A good resource is a page whose content is related to the query topic, and which has links to and/or from other related resources. So two factors are important in selecting good resources: the content of the page and the relevance of the pages which have links to/from the page.

Motivated by these observations, we propose a new general relevance propagation model for combining link and content information. Specifically, We define the "hyper-relevance" score of each page as a function of three variables, the content similarity of the page to the query ("self relevance"), a weighted sum of the hyper-relevance scores of the pages that point to the page, and a weighted sum of the hyper-relevance scores of the pages that are pointed to by the page. Formally, the relevance propagation model can be written as :

$$h(p) = f(S(p), \sum_{p_i \to p} h(p_i) w_I(p_i, p), \sum_{p \to p_j} h(p_j) w_O(p, p_j))$$

where h(p) is the hyper-relevance score of page p, S(p) is the content similarity between the page p and the query (i.e., "self relevance"), and w_I and w_O are weighting functions for in-link and out-link pages, respectively.

In principle, the choice of function f could be arbitrary. An interesting choice is a linear combination of the variables shown below:

$$\begin{aligned} h(p) &= \alpha S(p) + \beta \sum_{p_i \to p} h(p_i) w_I(p_i, p) \\ &\quad + \gamma \sum_{p \to p_j} h(p_j) w_O(p, p_j)) \\ \alpha + \beta + \gamma &= 1 \end{aligned}$$

The hyper-relevance scores can be computed iteratively until they converge to a limit, which is the final score of the page for ranking; more detail is presented in Section 4.

We now consider three special cases of this general relevance propagation model. Each special case corresponds to some reasonable user behavior.

3.1 Weighted In-Link

This model of user behavior is quite similar to the model of PageRank [4], except that it is not query-independent. The random surfer is given a start page at random. He keeps traversing links until he gets bored and starts from another random page. The probability that the random surfer visits a page is its hyper-relevance score.

This model has some characteristics which distinguish it from PageRank. First, it works on a subset of pages which are in the working set, rather than working on the whole set of data. (The details of constructing the working set is given in section 4.2) One of the properties of pages in the working set is that their content similarity to the query is above a threshold, so they can not be completely unrelated to the query. Second, the probability that the random surfer traverses an edge is proportional to the similarity of the target page and the query. That is, it is more probable that the user traverses and edge which leads to a more similar page, than jumping to a less similar page. Besides, when the random surfer gets bored, it jumps to new pages based on their similarity to the query.

This behavior can be formally modeled as follows. In each iteration, the new score of each page is computed as:

$$h(p) = \alpha S(p) + (1 - \alpha) \sum_{p_i \to p} h(p_i) w(p_i \to p)$$

$$w(p_i \to p) \propto S(p)$$

3.2 Weighted Out-Link

In this model, we assume that given a page to a user, he reads the content of the page with probability α and he traverses the outgoing edges with probability $(1 - \alpha)$. We also assume that it is more likely that the user traverses an edge that leads to a page whose content-similarity is higher.

Formally, we compute the hyper-relevance of each page iteratively using:

$$h(p) = \alpha S(p) + (1 - \alpha) \sum_{p \to p_j} h(p_j) w(p \to p_j)$$

$$w(p \rightarrow p_j) \propto S(p_j)$$

In each iteration, the hyper-relevance of each page is computed as a combination of its self-relevance and the hyper-relevance of the pages that it points to. The pages that are linked from a page do not have the same impact on its weight. Pages whose contents are more similar to the query are assumed to have more impact on the score of the page than those which are less similar. This effect is indicated in $w(p \rightarrow p_j)$. The hyper-relevance scores of pages are computed iteratively until they converge to a value, which are used to rank the pages.

3.3 Uniform Out-Link

In this special case, we assume that at each page, the user reads the content of the page, and with probability $(1-\alpha)$ he reads all the pages that are linked from the page. So the score of each page will be equal to a combination of its self relevance and the scores of all its linked pages.

Formally the hyper-relevance of each page is computed iteratively. In each iteration, the hyper-relevance of each page is:

$$h(p) = S(p) + (1-\alpha) \sum_{p \rightarrow p_j} h(p_j)$$

4 Implementation Issues

4.1 Preprocessing

We indexed all the web pages before dealing with queries. We used the Lemur toolkit for document indexing [3]. For document tokenization, we used the Fox stopword list and Porter stemmer.

4.2 Constructing the Working Set

We do not run our experiments on the whole set of data. Instead, for each query, we first construct a working set of pages and then find the top ranked pages among the pages of this subset.

To construct the working set, we first find the top 100000 pages which have the highest content similarity to the query from the whole collection of pages. We assume that the pages that are not among these pages can not be key resources for the given query. From these 100000 pages, a small number (about 200) of the most similar pages are selected to be the core set of pages. We then expand the core set to the working set by adding the pages that are among the 100000 pages and which point to the pages in the core set or are pointed to by the pages in the core set. We then run our experiments on these working sets. Note that each working set is specific to a query.

4.3 Computing the Hyper-relevance Values

In order to be able to compute the scores efficiently, we should come up with a way to find the hyper-relevance scores easily and in a feasible amount of time. In our implementation, we use matrix multiplication iteratively to compute the scores. We can use existing matrix multiplication methods to speed up the computation process.

Suppose that query Q and parameters are given. Our goal is to compute the limit hyper-relevance scores. To this end, we construct a square matrix $U_{n \times n}$ where n is the number of pages in the working set.

To construct the matrix, we first set all the entries to *zero* ($u_{ij} = 0, 1 \leq i, j \leq n$). We then add the influence of out-links: for each $1 \leq i, j \leq n$, we add $\beta \times w(p_i \rightarrow p_j)$ to the entry u_{ij}. Then we add the influence of in-links by adding $\gamma \times w(p_i \rightarrow p_j)$ to the entries u_{ji}. The only thing that remains is to add the effect of self content similarity. To add this effect, we add $\alpha \times S(p_i)$ to all the entries $u_{ij}, 1 \leq j \leq n$.

We then construct a vector H_n of hyper-relevance scores. At the beginning, we set all the entries in H to be $\frac{1}{n}$. The vector H should maintain the property that the sum of all the entries equal to *one* in every step.

In each step, we multiply U by H and we normalize the H vector. Each entry h_i in H corresponds to the hyper-relevance value of page p_i. It is easy to show that after multiplication, the hyper-relevance values are updated according to the general relevance propagation model.

If we do the multiplication and normalization iteratively, the hyper-relevance scores will converge to a limit, which is the final score we use for sorting results.

5 Experiment Results

5.1 Preliminary Experiments

Run	P@10	Content	Links	
			In	Out
Dir. Base	0.255	√		
Dir + W. IN	0.267	√	√	
Dir + W. OUT	0.265	√		√
Dir + U. OUT	0.265	√		√

Table 1: Experiments on TREC-2002 Data

The results of our algorithms on TREC-2002 data were quite promising. We explored the three presented methods, as well as a couple of other ways of propagating the relevance scores, and all the methods outperformed the baseline content-only language modeling method.

Table 5.1 summarizes our experiments on TREC-2002 data. Figure 1 shows the results of our algorithms compared to the baseline method.

As can be seen from the chart, all the methods have improvements over baseline.

Uniform out-link always perform better than baseline, while Weighted out-link has improvements for $\alpha > 0.2$ and Weighted in-link has improvements for $\alpha > 0.7$.

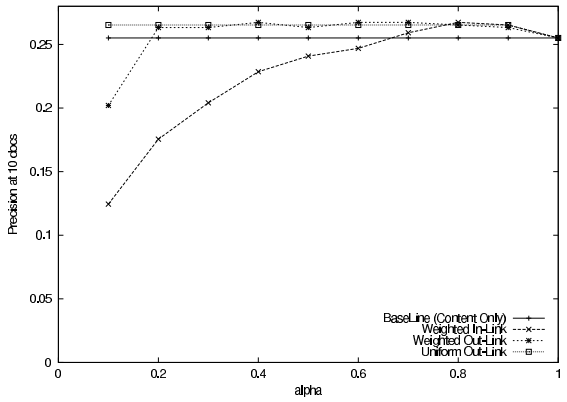

Figure 1: Experiments on TREC-2002 Data

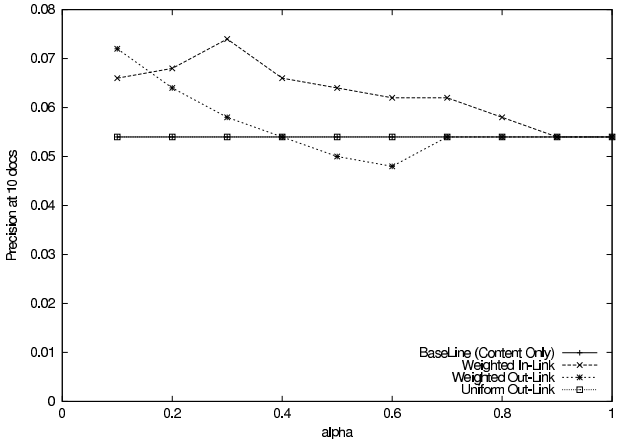

Figure 2: Experiments on TREC-2003 Data

5.2 Official TREC-2003 Results

Table 5.2 summarizes our experiments on TREC-2003 data. The second column in the table is precision at 10 for the submitted runs, while the third column shows precision at 10 when the parameters are tuned to get the optimal results.

Run	P@10		Cont.	Links	
	$\alpha = 0.8$	$\alpha = 0.1$		In	Out
Dir. Base	0.054*	0.054	$\surd$		
Dir+W. In	0.058*	0.066	$\surd$	$\surd$	
Dir+W. Out	0.054*	0.072	$\surd$		$\surd$
Dir+U. Out	0.054*	0.054	$\surd$		$\surd$
2s Base	0.064*	0.064	$\surd$		
2s+W. In	0.066	0.078	$\surd$	$\surd$	
2s+W. Out	0.062	0.082	$\surd$		$\surd$
2s+U. Out	0.062	0.062	$\surd$		$\surd$

Table 2: Experiments on TREC-2003 Data
$*$: Submitted Runs

Unlike our results on TREC-2002 data, our experiments on TREC-2003 data were not that promising. Figure 2 shows the results of our algorithms on TREC-2003 data. We see that for a majority settings of the parameter, the weighted out-link approach actually improves the performance clearly, but the α value that we obtained by training on the 2002 data corresponds to a bad setting for TREC-2003. When we optimize the parameter α on the TREC 2003 test set, both the weighted in-link and weighted out-link methods outperform the baseline.

As can be seen from the chart, the baseline method did not give good results on this year's data and although the proposed algorithms improved the ranking a bit, the overall precision was not good.

Comparing our two baseline runs (i.e., "2s Base"and "Dir Base"), we see that the two-stage smoothing coupled with mixture model for feedback performs significantly better than the the Dirichlet prior smoothing, confirming the effectiveness of two-stage smoothing and dictionary based feedback.

5.3 Discussion

We tried to find the reason for our poor results for this year's task. One difference between TREC-2002 data and TREC-2003 data is the number of relevant documents for each query. The average number of relevant documents per query is 31.48 for TREC-2002, while it is only 10.32 for TREC-2003 data. To find out whether this is a reason for our poor performance. We do experiments on two subsets of queries: a selected subset of queries from TREC-2002 whose average number of relevant documents is smaller than the average over all the queries and a selected subset of queries from TREC-2003 whose average number of relevant documents is larger than the average over all queries.

Our subset of queries from TREC-2002 data contains 25 queries with 10.48 relevant documents on average. The subset of queries from TREC-2003 data contains 20 queries with 19 relevant documents on average.

We tried our algorithms on these two subsets of queries. Figure 3 and Figure 4 show the results for the TREC-2002 subset and TREC-2003 subset, respectively.

As can be seen from Figure 3, the performance is not as good as the performance we obtained using the whole set of queries, but is not as poor as the results we obtained for TREC-2003 either.

On the other hand, Figure 4 shows that the performance is better than the performance we obtained using the whole set of queries, but is not as good as the TREC-2002 results.

What we can conclude is that small number of relevant documents per query can be a factor in our poor performance in TREC-2003, but there should be other reasons as well, that we are trying to find out.

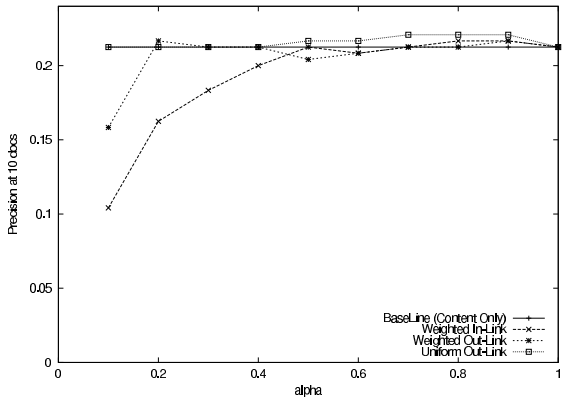

Figure 3: Experiments on the Subset of TREC-2002 Data

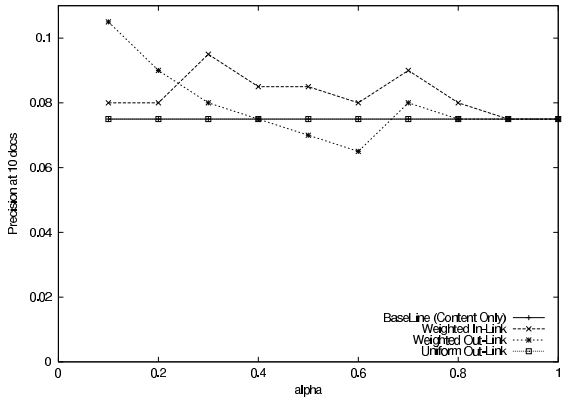

Figure 4: Experiments on the Subset of TREC-2003 Data

6 Conclusions and Future Directions

We explored two language modeling approaches for the topic distillation task: (1) basic query likelihood method with Dirichlet prior smoothing, and (2) two-stage smoothing with mixture model feedback. The results show that the the two-stage smoothing with feedback significantly outperforms the query likelihood method, confirming the effectiveness of two-stage smoothing [9] and mixture model feedback method [7]

We also proposed a new general relevance propagation model for combining link and content information, and explored a number of specific methods derived from the model. The experiment results show that combining link and content information generally performs better than using only content information, though the amount of improvement is sensitive to the document collection and tuning of parameters.

For the future work, we plan to do more experiments to find out what factors affect the performance of our algorithms. We will also explore how to tune the parameters for obtaining the optimal results.

References

[1] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of SIGIR'2001*, pages 111–119, Sept 2001.

[2] A. MacFarlane. Pliers at trec 2002. In *Proceedings of TREC 2002*, 2002.

[3] P. Ogilvie and J. Callan. Experiments using the lemur toolkit. In *Proceedings of the 2001 TREC conference*, 2002.

[4] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[5] J. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the ACM SIGIR*, pages 275–281, 1998.

[6] H. Xu, Z. Yang, B. Wang, B. Liu, J. Cheng, Y. Liu, Z. Yang, and X. Cheng. Trec 11 experiments at cas-ict: Filtering and web. In *Proceedings of TREC 2002*, 2002.

[7] C. Zhai and J. Lafferty. Model-based feedback in the KL-divergence retrieval model. In *Tenth International Conference on Information and Knowledge Management (CIKM 2001)*, pages 403–410, 2001.

[8] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of SIGIR'2001*, pages 334–342, Sept 2001.

[9] C. Zhai and J. Lafferty. Two-stage language models for information retrieval. In *Proceedings of SIGIR'2002*, pages 49–56, Aug 2002.

[10] C. Zhai, T. Tao, H. Fang, and Z. Shang. Improving the robustness of language models: Uiuc trec-2003 robust and genomics experiments. In *Notebook of TREC 2003*, 2003.

[11] M. Zhang, R. Song, C. Lin, S. Ma, Z. Jiang, Y. Jin, Y. Liu, and L. Zhao. Thu trec 2002: Web track experiments. In *Proceedings of TREC 2002*, 2002.