Selective Term Proximity Scoring Via BP-ANN

Ju Yang, Jiancong Tong, Rebecca J. Stones, Zhaohua Zhang, Benjun Ye, Gang Wang, Xiaoguang Liu

College of Computer and Control Engineering, Nankai University, Tianjin, China

Email: {yangju,jctong, rebecca.stones82, zhangzhaohua, yebj, wgzwp, liuxg}@nbjl.nankai.edu.cn

ABSTRACT

When two terms occur together in a document, the probability of a close relationship between them and the document itself is greater if they are in nearby positions. However, ranking functions including term proximity (TP) require larger indexes than traditional document-level indexing, which slows down query processing. Previous studies also show that this technique is not effective for all types of queries. Here we propose a document ranking model which decides for which queries it would be beneficial to use a proximity-based ranking, based on a collection of features of the query. We use a machine learning approach in determining whether utilizing TP will be beneficial. Experiments show that the proposed model returns improved rankings while also reducing the overhead incurred as a result of using TP statistics.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval-Search process

Keywords

Information retrieval, Term proximity, Query Effectiveness Prediction

1. INTRODUCTION

Search engine users want relevant documents returned quickly when searching. Searching is often performed using an inverted index [21], which stores a list of occurring terms, and for each term, documents in which that term occurs are recorded, along with term frequency within each document and all the corresponding position information.

The search process for conjunctive queries goes through two main phases: list intersection and ranking [6]. We first find the documents that contain all the query terms, then rank them according to their relevance to the query; ideally, the most relevant documents are returned. Traditional

methods for assessing if a document is relevant to a query use two kinds of features: (a) term-independent features, e.g., PageRank; and (b) term-dependent features. Term-dependent features focus on term frequency and inverted document frequency. BM25 [14] is one of the most widely used; given query q, the document d is assigned the score:

$$S_{\text{BM25}}(q, d) = \sum_{\text{term } t \in q} w_t \frac{f_{d,t}(k_1 + 1)}{f_{d,t} + k_1(1 - b + \frac{b|d|}{\text{avg}_d})}, \quad (1)$$

where k_1 and b are predefined constants, $|d|$ is the length of document d, avg_d is the average document length in the collection, $f_{d,t}$ is the frequency of t in document d, and w_t is the inverse document frequency (idf) of term t.

The order in which terms appear in the document and the distance between their locations are both important ranking criteria. Consider the two-term query "search engine" for ranking the following toy example documents:

d_1: ... *word* **search engine** *word word word word* **search engine** *word word word word word word* ...

d_2: ... **search** *word word* **search** *word word* **engine search** *word* **engine search** *word* **engine** *word* ...

In this example, document d_1 can be regarded as more relevant, despite $f_{d_2,t} > f_{d_1,t}$ for both terms $t \in \{\text{search}, \text{engine}\}$. Consequently, there is active research on methods of integrating term proximity (TP) into the usual "bag of words" ranking [22].

TP score has been demonstrated to have an overall positive effect on search quality [18]. However, there are two caveats: (a) some queries return inferior rankings when utilizing TP score, and (b) incorporating TP score increases computational time. This motivates us to propose a model which selects which queries would likely benefit from incorporating TP score into their ranking.

The remainder of the paper is organized as follows: Section 2 summarizes related work on proximity ranking models. Section 3 introduces our proposed method. Section 4 details the experiment setup and presents the results. In Section 5, we present our conclusions and future research directions are suggested.

2. RELATED WORK

There are two types of models using proximity in ranking: (a) complex ranking functions that combine hundreds of features (TP being one of them) using sophisticated machine learning techniques [9], and (b) variations of the classic ranking models. While the former achieves more effective

results than the latter, it is sometimes too computationally expensive to use. Recent work has explored approaches to achieve a better balance between retrieval effectiveness and efficiency [20]. However, in this paper, we aim to treat each query flexibly, so we focus on the latter.

Rasolofo and Savoy [13] proposed a TP-based ranking scheme BM25TP, a modified version of BM25 (1), which incorporates term proximity (a similar scheme was presented by Büttcher *et al.* [4]). In BM25TP, the rank of document d is given by:

$$S_{\text{BM25TP}}^{\text{RS}}(q, d) = S_{\text{BM25}}(q, d) + S_{\text{TP}}^{\text{ACC}}(q, d) \qquad (2)$$

$$S_{\text{TP}}^{\text{ACC}}(q, d) = \sum_{t \in Q} \min\{1, w_t\} \frac{\text{acc}_d(t)(k_1 + 1)}{\text{acc}_d(t) + k_1\left(1 - b + \frac{b|d|}{\text{avg}_d}\right)}, \qquad (3)$$

where $\text{acc}_d(t)$ denotes the proximity accumulator for term t:

$$\text{acc}_d(t) = \sum_{s \neq t} w_t \, \text{tpi}_d(t, s)$$

and

$$\text{tpi}_d(t, s) = \sum_{\substack{\text{occurrence } o(t) \text{ of } t \\ \text{in document } d}} \text{dist}(o(t), s)^{-2}$$

for each given term pair (t, s), where $t \neq s$, and $\text{dist}(o(t), s)$ is the number of terms between the position of $o(t)$ and the position of the preceding occurrence of the term s.

An assortment of other methods of utilizing TP in ranking have been studied. Akritidis *et al.* [1] not only takes into account term proximity, but also the order of terms in the query. Zhu *et al.* [24] put forward some new ideas based on web page structure and set estimation rules for early termination to speed up top-k computation. Svore *et al.* [17] and Song *et al.* [16] utilized TP in the form of "spans" to improve the accuracy of ranking functions. Tao and Zhai [18] introduced five measures and combined them with an existing retrieval model with two newly designed heuristic constraints; their experiments showed significant performance improvement on the KL-divergence language model and the BM25 model. Lv and Zhai [11] presented four proximity-based density functions to estimate different positional language models (PLMs), namely the Gaussian, triangular, cosine, and circle. Metzler *et al.* [3, 12] developed a general Markov random field (MRF) retrieval model that captures various kinds of term dependencies: full independence, sequential dependence, and full dependence. Cummins and O'Riordan [5] outlined an extensive list of possible term proximity measures, and incorporated them to the original framework by machine learning methods.

Different queries benefit from different proximity features and methods [5, 10, 17]. Moreover, a too-complicated ranking formula may be a burden to use, both for the operator and by requiring too much overhead. This motivates us to propose a method where TP statistics are utilized only when they are most useful.

3. SELECTIVE TP MODEL

3.1 Features considered

Table 1 lists the features considered in this work. These features can be roughly divided into two categories: (a) query dependent, and (b) term dependent. Term dependent

Table 1: Summary of query features used

query features
number of relevant documents
term frequency features
mean; min; max; sum idf
sum of squared idfs
sum of squares of ascendant idfs
sum of squares of descendant idfs
term position features
mean; min; max; sum pos
square statistics

features are divided into two subcategories: frequency-based and position-based.

The query dependent features we include is the number of documents related to the query. The inverted document frequency (idf) indicates the overall importance of a term, and we utilize its statistics: mean, min, max, and sum; the sum of squared idfs, and the sum of squared differences between ascendant or descendant idf values between consecutive terms in a query[1]. The position-based features include the average position of a term in a document, averaged over all documents, which we call the general position (abbreviated pos). Most pos statistics used are analogous to the idf statistics. These features vary in their ability to distinguish queries from each other; we specify the features actually used in Section 4.

3.2 Term proximity score

Three of the most popular TP ranking functions, which we test our selective model based on, are introduced here. Two are BM25TP, given by (2), and MRF by Metzler *et al.* [12]. MRF is defined by the following ranking function (full details are omitted for space reasons):

$$S_{\text{MRF}}(q, d) := S_{\text{TF}}(q, d) + \sum_{c \in O} \lambda_O f_O(c) + \sum_{c \in O \cup U} \lambda_U f_U(c); \qquad (4)$$

$$S_{\text{TF}}(q, d) := \sum_{c \in T} \lambda_T f_T(c). \qquad (5)$$

where T is the set of 2-cliques involving a query term and a document, O is the set of cliques containing the document node and two or more query terms that appear contiguously within the query, and U is the set for query terms appearing non-contiguously within the query. In this paper, we use an extension model proposed by [3].

The third one we use is from Tao and Zhai [18] who instead calculate a term-proximity-based rank by:

$$S_{\text{EXP}}^{\text{TZ}}(q, d) := S_{\text{TP}}^{\text{TZ}}(q, d) + S_{\text{BM25}}(q, d); \qquad (6)$$

$$S_{\text{TP}}^{\text{TZ}}(q, d) := \log\left(\alpha + \exp(-\text{min\_dist}(q, d))\right)$$

where $\text{min\_dist}(q, d)$ is the minimum distance between any occurrence of any two query terms in document d, and α is a parameter. Tao and Zhai state that (6) provides stable performance when α is set to 0.3, which we use for our experiments.

---

[1]Ascendant (resp. descendant) idfs refer to consecutive term pairs whose first idf value is less (resp. greater) than the second.

Some studies have identified a decaying function between two words to calculate the strength of their association [7, 19, 23].

3.3 Our approach

As our ranking functions, we use (4) and a generalized combination of (2) and (6):

$$S_{EXP}(q,d) := \epsilon S_{TP}^{TZ}(q,d) + (1-\epsilon) S_{BM25}(q,d);$$

$$S_{BM25TP}(q,d) := \beta S_{TP}^{ACC}(q,d) + (1-\beta) S_{BM25}(q,d). \quad (7)$$

Parameters ϵ and β are used to adjust the weighting of BM25 and the TP score. We test $\epsilon, \beta \in \{0.1, 0.2, \ldots, 0.9\}$ for the two query sets in our experiments, and choose the parameters leading to the best mean average precision (MAP).

The MAP of each query is used to evaluate the performance of the two ranking models. If a query gets better results using (7) than (1) or (4) than (5), its features will be labeled as 1, otherwise 0. These results are used to train a (supervised) classifier, determining whether or not using TP score is likely to benefit the document rankings for arbitrary queries.

We use a Back Propagation Artificial Neural Network (BP-ANN) to build our selective TP model, because of its powerful learning ability and rapid forecasting speed. BP-ANN [15] uses a back-propagation algorithm to modify the internal network weights during the training process. In our experiment, we establish a one-node (denoting the query type) output layer BP-ANN, which contains one hidden layer and whose input nodes are query features.

4. EXPERIMENTS

All experiments are performed on the GOV2 data set using Porter stemming. We use the query set MQ2007 and MQ2008 for evaluation. The BM25 scores used in this paper are extracted from LETOR4.0[2].

Figure 1 shows the MAP values of the rankings for queries, as the proportion of queries using TP (EXP-score) varies. The queries are sorted by how beneficial it would be to use TP scores in their ranking, with the most benefited coming first. Figure 1 shows that using TP scores does not always improve retrieval quality (assigning more than around 40% has no benefit). We also test methods assigning a label 0 or 1 to a query randomly, which again shows that naively increasing of proportion of queries utilizing TP will not necessarily result in a performance improvement.

Only relevant features are necessary in the BP-ANN model construction, so we remove unnecessary features. To determine feature importance, we combine statistical methods (ranksum, z-score, and χ-squared), a searching algorithm (decision tree), and a feature weight algorithm (relief).

We find that max idf, sum idf, and sum of squared idfs have relatively more importance among the term frequency features, and max pos, min pos, sum pos, and mean pos have relatively more importance among the term position features. As such, we primarily use max pos, min pos, sum pos, and mean pos for EXP; sum idf, max idf, and min pos for MRF; and min idf, sum of squared idfs, sum of squared differences between descendant idfs, and sum pos for BM25TP.
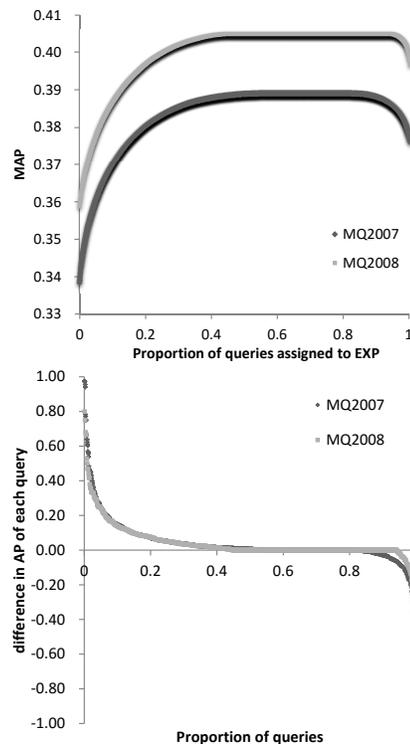
Figure 1: MAP on the two query sets as the proportion of queries using TP score varies; Sorted difference in average precision of each query

As other researchers have observed [5, 8, 10, 17], we also find that query length is an important factor in distinguishing whether or not using TP score will be beneficial, so we train independent models for different query length. Because of their effective performance in a more systematic study [2], we consider queries with 3 to 5 terms.

After each query is labeled, we train a neural network on 70% of all the data and test the effectiveness and efficiency of the model on the remaining data. In our BP networks, the input layer has 3 to 5 features and the output layer has 1 node. The maximum number of iterations is set to 1000 and the learning rate of the network is 0.01. We choose the sigmoid function

$$f(x) = \frac{1}{1+e^{-x}}$$

as the activation function and test the performance of different hidden layer nodes. All the networks aim to correctly predict queries labeled 1 as much as possible, motivated by Figure 2 which indicates that mispredicting queries labeled 1 is consistently worse than other mispredictions. The bias on mispredictions is used as a reference for process parameter adjustment in the network. The number of hidden layer nodes used and its momentum coefficient α are listed in Table 2, along with the precision and recall values on the training and test data.

We compare three TP-based rankings for each TP score: _tpAll (where TP is always used for ranking), _tpS which calculates TP score depending on BP-ANN predictions (the proposed ranking method), and _oracle, a theoretically perfect situation where we know *a priori* whether or not a query
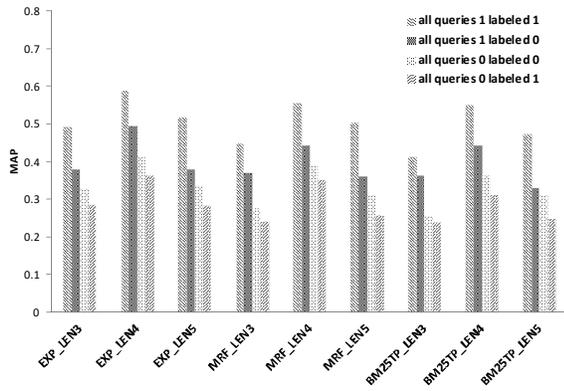
Figure 2: MAP degradation caused by wrong judgement for different query sets

would benefit from TP score. For comparison, we also include a non-TP-based ranking `_tpNo` given by (1) and (5).

We use three methods for measuring the quality of the rankings: MAP, precision for top-k results, and Mean Normalized Discounted Cumulative Gain (Mean NDCG), given in Table 3. We also list the number of queries benefiting from calculation of TP score and the throughput.

Table 3 shows that the TP rankings (`_tpAll` and `_tpS`) consistently exhibit significantly better rankings than without TP (`_tpNo`). We also see that the selective model (`_tpS`) returns slightly better rankings than `_tpAll` while having better throughput. In terms of MAP, we see that MRF is consistently superior to the other ranking formulas. However, `_tpS` used with EXP is the nearest to the corresponding `_oracle` (the best possible MAP). Further, we can also find that `_tpS` shows a better performance (vs. `_tpAll`) in terms of $k = 1$ precision, which is a critical measure for exact queries, such as queries restricted to web sites.

5. CONCLUDING REMARKS

Recent studies have achieved promising retrieval performance by taking term proximity into consideration in relevance scoring. In this work, we propose a modified TP score ranking scheme which predicts which queries will benefit from using TP score in their rankings. In this way, we can: (a) achieve a better ranking from utilizing TP scores, and (b) achieve rankings with slightly better quality than the rankings given when always incorporating the TP score, but with better throughput. In essence, we utilize TP score only when it's helpful.

Our work could be extended in several directions, e.g.: (a) The use of more features, particularly those that capture a notion of term proximity, could be explored. (b) We could use a more complicated weighting of the queries' benefit from using TP score (here we use a simple 1 vs. 0 weighting). This would enable us to use a linear regression model, which may achieve more effective results. (c) Since different features benefit different types of queries, we could train a collection of models, individually designed for a single query type.

6. ACKNOWLEDGMENTS

7. REFERENCES

[1] L. Akritidis, D. Katsaros, and P. Bozanis. Improved retrieval effectiveness by efficient combination of term proximity and zone scoring: A simulation-based evaluation. *Simul. Model. Pract. Th.*, 22:74–91, 2012.

[2] L. Azzopardi. Query side evaluation: an empirical analysis of effectiveness and effort. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 556–563. ACM, 2009.

[3] M. Bendersky, D. Metzler, and W. B. Croft. Learning concept importance using a weighted dependence model. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 31–40. ACM, 2010.

[4] S. Büttcher, C. L. A. Clarke, and B. Lushman. Term proximity scoring for ad-hoc retrieval on very large text collections. In *Proc. SIGIR*, pages 621–622, 2006.

[5] R. Cummins and C. O'Riordan. Learning in a pairwise term-term proximity framework for information retrieval. In *Proc. SIGIR*, pages 251–258, 2009.

[6] C. Dimopoulos, S. Nepomnyachiy, and T. Suel. A candidate filtering mechanism for fast top-k query processing on modern CPUs. In *Proc. SIGIR*, pages 723–732, 2013.

[7] J. Gao, M. Zhou, J.-Y. Nie, H. He, and W. Chen. Resolving query translation ambiguity using a decaying co-occurrence model and syntactic dependence relations. In *Proc. SIGIR*, pages 183–190, 2002.

[8] S. Huston and W. B. Croft. A comparison of retrieval models using term dependencies. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 111–120. ACM, 2014.

[9] T.-Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *SIGIR*, pages 3–10, 2007.

[10] X. Lu, A. Moffat, and J. S. Culpepper. How effective are proximity scores in term dependency models? In *Proc. ADCS*, page 89, 2014.

[11] Y. Lv and C. Zhai. Positional language models for information retrieval.

[12] D. Metzler and W. B. Croft. A Markov random field model for term dependencies. In *Proc. SIGIR*, pages 472–479, 2005.

[13] Y. Rasolofo and J. Savoy. *Term proximity scoring for keyword-based retrieval systems*. Springer, 2003.

[14] S. E. Robertson and K. S. Jones. Relevance weighting of search terms. *J. Assoc. Inf. Sci. Technol.*, 27(3):129–146, 1976.

[15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5, 1988.

[16] R. Song, M. J. Taylor, J.-R. Wen, H.-W. Hon, and Y. Yu. Viewing term proximity from a different perspective. In *Advances in Information Retrieval*, pages 346–357. 2008.

Table 2: Parameters of BP-ANN and the precision and recall values on the training and test data.

	EXP			MRF			BM25TP		
Query len.	3	4	5	3	4	5	3	4	5
Prec. on train	59.29%	61.74%	61.15%	53.48%	60.26%	56.96%	54.02%	49.77%	42.18%
Recall on train	89.73%	99.39%	94.12%	91.25%	90.97%	91.84%	90.38%	87.60%	91.18%
Prec. on test	69.30%	72.53%	64.62%	53.98%	62.89%	55.88%	51.40%	51.06%	47.37%
Recall on test	98.80%	95.65%	95.45%	88.41%	92.42%	95.00%	82.09%	92.31%	90.00%
#hidden nodes	43	58	47	45	39	47	52	10	20
α	1	0.85	0.85	0.35	0.9	0.85	0.95	0.25	0.75

Table 3: Performance under the various ranking methods in terms of query length (Q len.), MAP, mean NDCG, precision, and throughput. We also include the number of queries which uses TP ranking (#TP-Q).

	Q len.	MAP	Mean NDCG	Prec. $k=1$	Prec. $k=3$	Prec. $k=10$	#TP-Q	Throughput (Q/s)
		0.4358	0.4497	0.3427	0.3683	0.3615	143	334.56
	3	0.4400	0.4582	0.3776	0.3893	0.3622	107	379.71
		0.4481	0.4733	0.3986	0.4033	0.3748	80	477.82
		0.3772	0.3967	0.3287	0.3287	0.3182	—	974.75
EXP_BM25_tpAll		0.3294	0.3524	0.2632	0.2544	0.2649	114	96.42
EXP_BM25_tpS	4	0.3340	0.3599	0.2807	0.2632	0.2667	91	100.06
EXP_oracle		0.3470	0.3763	0.2895	0.2749	0.2833	69	111.35
EXP_tpNo		0.3026	0.3264	0.2193	0.2368	0.2377	—	440.33
		0.3397	0.3667	0.2568	0.2523	0.2662	74	80.84
	5	0.3422	0.3687	0.2703	0.2568	0.2676	65	81.22
		0.3525	0.3782	0.2703	0.2703	0.2743	44	98.57
		0.3295	0.3482	0.2568	0.2613	0.2527	—	300.75
		0.4791	0.4883	0.4056	0.3916	0.3811	143	330.00
	3	0.4855	0.4924	0.4266	0.4103	0.3916	113	346.53
		0.5261	0.5362	0.4965	0.4592	0.4238	69	443.31
		0.4558	0.4480	0.3706	0.3473	0.3608	—	903.79
MRF_tpAll		0.4201	0.4365	0.4298	0.3333	0.3246	114	77.37
MRF_tpS	4	0.4238	0.4416	0.4123	0.3450	0.3289	97	77.92
MRF_oracle		0.4516	0.4869	0.4561	0.3919	0.3526	66	112.86
MRF_tpNo		0.3397	0.3514	0.1930	0.2427	0.2693	—	397.00
		0.3760	0.3953	0.2535	0.2864	0.2958	71	80.91
	5	0.3810	0.4003	0.2958	0.2911	0.2887	68	81.01
		0.4107	0.4395	0.3380	0.3521	0.3268	40	110.77
		0.3338	0.3502	0.2394	0.2488	0.2549	—	321.98
		0.4128	0.4302	0.3310	0.3615	0.3493	142	352.71
	3	0.4145	0.4354	0.3380	0.3662	0.3514	107	424.49
		0.4300	0.4575	0.4085	0.4061	0.3683	67	489.07
		0.3807	0.4008	0.3380	0.3404	0.3275	—	912.59
BM25TP_tpAll		0.3460	0.3542	0.2870	0.2580	0.2635	115	97.84
BM25TP_tpS	4	0.3506	0.3621	0.3130	0.2696	0.2730	94	150.02
BM25TP_oracle		0.3679	0.3931	0.3130	0.3014	0.2939	52	206.76
BM25TP_tpNo		0.3052	0.3273	0.2261	0.2435	0.2426	—	410.79
		0.3533	0.3580	0.2162	0.2703	0.2905	74	85.35
	5	0.3712	0.3832	0.2703	0.2973	0.2986	57	91.94
		0.3894	0.4036	0.3378	0.3288	0.3149	30	140.26
		0.3425	0.3587	0.2838	0.2928	0.2662	—	306.36

[17] K. M. Svore, P. H. Kanani, and N. Khan. How good is a span of terms: exploiting proximity to improve web retrieval. In *Proc. SIGIR*, pages 154–161, 2010.

[18] T. Tao and C. Zhai. An exploration of proximity measures in information retrieval. In *Proc. SIGIR*, pages 295–302, 2007.

[19] O. Vechtomova and Y. Wang. A study of the effect of term proximity on query expansion. *J. Inf. Sci.*, 32(4):324–333, 2006.

[20] L. Wang, J. Lin, and D. Metzler. A cascade ranking model for efficient ranked retrieval. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 105–114. ACM, 2011.

[21] H. Yan, S. Ding, and T. Suel. Inverted index compression and query processing with optimized document ordering. In *Proc. WWW*, pages 401–410, 2009.

[22] H. Yan, S. Shi, F. Zhang, T. Suel, and J.-R. Wen. Efficient term proximity search with term-pair indexes. In *Proc. CIKM*, pages 1229–1238, 2010.

[23] D. Yuret. Discovery of linguistic relations using lexical attraction. *arXiv preprint cmp-lg/9805009*, 1998.

[24] M. Zhu, S. Shi, M. Li, and J.-R. Wen. Effective top-k computation with term-proximity support. *Inf. Process. Manage.*, 45(4):401–412, 2009.