Journal of Computational Information Systems 11: 24 (2015) 7795–7803
Available at http://www.Jofcis.com

Early Termination Prediction in Search Engine [⋆]

Yilin WANG [1], Gang WANG [1], Xiaoguang LIU [2,*]

[1] *Nankai-Baidu Joint Lab, College of Computer and Control Engineering, Nankai University, Tianjin 300071, China*

[2] *Nankai-Baidu Joint Lab, College of Software, Nankai University, Tianjin 300071, China*

Abstract

Early termination is an important pruning technique for optimizing the performance of query processing in search engines. It returns the result without scanning the entire posting lists of the query terms and therefore it can reduce the time of query processing. Early termination depends on the score of processed documents. So it is not conducive to parallelize it for a single query. If the position of early termination is predicted, it can help parallelism and scheduling. In this paper, we propose mathematical models to predict the size of the intersection set of inverted lists. On the basis of that, we analyze different features and propose a machine learning method to predict early termination. The experimental results show that our prediction is reasonably accurate.

Keywords: Search Engine; Early Termination; Prediction; Correlation

1 Introduction

The search engine system is to collect information from the Internet, establish the index, and return the relevant information for queries. With the development of Internet, information increases faster and faster, and users more and more depend on search engines. Search engines must provide high throughput and fast response time to satisfy user requirements. It is significant to optimize query performance.

Recently, many optimization methods take different strategies based on prediction. Starting to process a query, search engines predict the performance of query processing firstly, then schedule the query based on the predicted results to balance and optimize both response time and throughput. Macdonald et al. [7] proposed a query scheduling algorithm based on the response time prediction. The method proposed in [5] predicts the query execution time and parallelizes

[⋆]This work was partially supported by NSF of China (Grant numbers: 61373018, 11301288), Program for New Century Excellent Talents in University (Grant number: NCET130301), and the Fundamental Research Funds for the Central Universities (Grant number: 65141021).

[*]Corresponding author.

Email addresses: `wangyilin@nbjl.nankai.edu.cn` (Yilin WANG), `wgzwp@nbjl.nankai.edu.cn` (Gang WANG), `liuxg@nbjl.nankai.edu.cn` (Xiaoguang LIU).

1553–9105 / Copyright © 2015 Binary Information Press
DOI: 10.12733/jcis15C0260
December 15, 2015

long queries. Most of the contemporary researches studied on the prediction method of query processing time, but payed less attention to other features.

The position of early termination determines whether a document will be processed. The result of prediction can be used to estimate the computational quantity and schedule queries, which contributes to the load balancing of a distributed search engine. When parallelizing for single query, if it is predicted that the position of early termination is before d_{max}, it is reasonable not to parallel process the documents after d_{max}, which can reduce the load while parallelizing. In this paper, we study on the method of early termination prediction. The contributions of our work are as follows:

- **Inverted lists intersection size prediction** Given a group of terms, we propose mathematical models to predict the number of documents in the intersection set of their inverted lists. The experimental results show the high linear correlation between the predicted and real values.
- **Early termination prediction** Given a query and a boundary parameter, we develop a classifier to predict whether the position of early termination before the boundary. The experimental results show the high prediction accuracy of our method.

2 Background and Related Work

2.1 Query processing

A document is a sequence of terms. A search engine indexes many documents and assign a unique number $docId_d$ to each document. The common index structure in search engines is inverted index. An inverted index stores the information about a document set D and a term set T and consists of inverted lists. For each term $t \in T$, the inverted list $l(t)$ consists of postings. For each document $d \in D$ that contains term t, $posting_{t,d}$ stores $docId_d$ and term frequency $tf_{t,d}$. In this paper, a document d is considered as a set of terms: $d = \{t_1, t_2, t_3 \dots t_{|d|}\}$, and an inverted list $l(t)$ is considered as a set of documents: $l(t) = \{d | t \in d, d \in D\}$.

A query may contain several terms and the user often want to get the documents that contains all terms. This paper focuses on AND queries which require the result documents contains all query terms. A query q is considered as a set of terms: $q = \{t_1, t_2, t_3 \dots t_{|q|}\}$. All documents that match query q form a set $R(q)$ which is equal to the intersection of the inverted lists of all query terms: $R(q) = \{d | t \in d, t \in q, d \in D\} = \bigcap_{t \in q} l(t)$.

There are two strategies of query processing [12]:

- **Term-at-a-time (TAAT)** Process the inverted lists of query terms one after another. Start to process next query term only after having completed the processing of current term.
- **Document-at-a-time (DAAT)** Process documents one by one. Start to process next document only after having evaluated current document for all query terms.

For a query, a lot of documents match it while a few is useful, therefore search engines rank each document and return documents with high ranks. Because users usually only need the documents most meet their requirements and search engines only need get the results displaying in current page, it only needs to return top k documents. That is called top-k query processing.

2.2 Ranking function

A ranking function computes a score for each document. It is usually a combination of different kinds of scores. The most common scores to rank documents are static rank score (SR score), information retrieval score (IR score) and term proximity score (TP score) [10]. SR score describes the importance of a document, it is independent of a specific query. IR score is based on term frequency. TP score considers the positions of query terms.

In this paper, we use the ranking function [15] that combines SR score and IR score. SR score can be computed by PageRank algorithm, which is based on the hyper link relationships of web pages. It is independent of a query and processed off-line. IR score can be calculated using the BM25 formula [9] which considers both term frequency and document length. It is the linear combination of IR score of query terms and the formula is as follows:

$$IR(d,q) = \sum_{t\in q} \omega_t \frac{(k_1+1)tf_{d,t}}{tf_{d,t} + k_1(1-b+b\dfrac{dl_d}{avdl})} \tag{1}$$

where $tf_{d,t}$ is the frequency of term t appearing in document d, dl_d is the length of document d, $avdl$ is the average length of all the documents, $k_1 \in [0, +\infty)$ and $b \in [0, 1]$ are two parameters. ω_t is inverse document frequency (IDF), which is calculated by the following formula:

$$\omega_t = \log \frac{N - n_t + 0.5}{n_t + 0.5} \tag{2}$$

where N is the total number of documents and n_t is the number of documents containing term t.

Given a document d and a query q, the ranking function using in this paper is the weighted average of SR and IR score:

$$S(d,q) = \alpha \cdot SR(d) + \beta \cdot IR(d,q) \tag{3}$$

where parameters fulfill $\alpha \in [0, 1]$, $\beta \in [0, 1]$, $\alpha + \beta = 1$. SR and IR score should be normalized to range $[0, 1]$ and the normalized IR score $IR(d, q)$ is calculated as follows:

$$IR(d,q) = \frac{\sum\limits_{t\in q} \omega_t IR(d,t)}{\sum\limits_{t\in q} \omega_t} \tag{4}$$

where $IR(d, t)$ is the normalized IR score, which is calculated as follows:

$$IR(d,t) = \frac{tf_{d,t}}{tf_{d,t} + k_1(1-b+b\dfrac{dl_d}{avdl})} \tag{5}$$

2.3 Early termination

Once a search engine finds the current result set for a query is good enough, it stops searching and returns the current result set. That is called early termination [1]. Most of the early termination algorithms rebuild the index based on scores. The method in [8] sorts the index by within-document frequency related to IR score. Anh et al. [2] proposed an algorithm that partitions a posting list into segments. In each segment, the documents have same impact value relevant to IR score. The strategy proposed in [6] partitions a posting list into two segments based on the IR score of documents and sorts each segment by SR score. In addition, Yan et al. [14] studied on the early termination algorithm with TP score.

The early termination method used in this paper is a rank-safe [11] algorithm proposed by Zhang et al. [15]. It computes a global score $GS(d)$ for each document d and resorts documents descending by the global score, where $GS(d)$ represents the upper bound of document score $S(d)$. By Eq. (4), for any query q, $IR(d,q) \leq \max_{t\in q} IR(d,t) \leq \max_{t\in d} IR(d,t)$, and the upper bound of IR score (UBIR score) can be calculated using the following formula:

$$UBIR(d) = \max_{t\in d} IR(d,t) \tag{6}$$

Noting that both $SR(d)$ and $UBIR(d)$ are term-independent, $GS(d)$ is calculated as follows:

$$GS(d) = \alpha \cdot SR(d) + \beta \cdot UBIR(d) \tag{7}$$

DAAT strategy can be combined with early termination effectively. A rank-safe early termination algorithm evaluates documents one by one and maintains the current top-k results of processed documents. The process stops and returns the current top-k results once the following terminating condition is satisfied:

$$S_k \geq S_t \tag{8}$$

where S_k is the score of the current k-th document, and S_t is the score upper bound of unprocessed documents. Rebuilding the index as above, while processing the current document d, S_t can be calculated by $S_t = GS(d)$. The first document that satisfies the terminating condition is the position of early termination.

3 Inverted Lists Intersection Set Size Prediction

3.1 Intersection set size calculation

Given a set of terms $q = \{t_1, t_2, t_3 \ldots t_{|q|}\}$, we can use the probability method to estimate the size of intersection set $|R(q)|$. If we can calculate the probability $P(q)$ that all terms of q appear in a random document, then the expectation of intersection set size $|R(q)|$ is:

$$|R(q)| = |D| \cdot P(q) \tag{9}$$

where $|D|$ is the number of documents in the entire index. Defining $P(t)$ as the probability that term t appears in a random document, it is obvious that $P(q)$ has much to do with $P(t)$ for $t \in q$ and we can calculate $P(t)$ as follows:

$$P(t) = \frac{|l(t)|}{|D|} \tag{10}$$

where $|l(t)|$ is the size of inverted list $l(t)$. We set different assumptions and then try to calculate $P(q)$ using $P(t)$.

- **Independent model**

  Assuming that terms are independent of each other, $P(q)$ can be calculated as follows:

$$P(q) = \prod_{t \in q} P(t) \tag{11}$$

- **Correlation model**

  In fact, terms are correlated with each other. It is difficult to think of the relation of all query terms directly. So we combine some of them into a group, then consider the correlation between a group g and a term t and calculate as follows:

$$P(g \cup \{t\}) = P(g)P(t) + \rho(g,t)\sqrt{P(g)(1-P(g))P(t)(1-P(t))} \tag{12}$$

  where $\rho(g,t)$ is the correlation coefficient of g and t. For each off-line query q, we use DAAT strategy to calculate the intersection set. In each step, we can get $R(g)$ and $R(g \cup \{t\})$, then $\rho(g,t)$ can be computed, where $g \subset q$, $t \in q$. For the term groups that have same size, we compile statistics and calculate the average $\rho(g,t)$ of them. Because the off-line query set is limited, we use a function $\rho'(|g \cup \{t\}|)$ to fit the statistical results, which is used to compute parameter $\rho(g,t)$.

- **Exponential model**

  If query length is above two, the correlation model is asymmetric for the query terms. Different order for computation will get different results, but in fact the correlation is independent of the order. Therefore, we try to propose an exponential model to calculate $P(q)$ directly, which is on the basis of the independent model and add an exponential parameter $k(q)$ to describe the correlation:

$$P(q) = (\prod_{t \in q} P(t))^{k(q)} \tag{13}$$

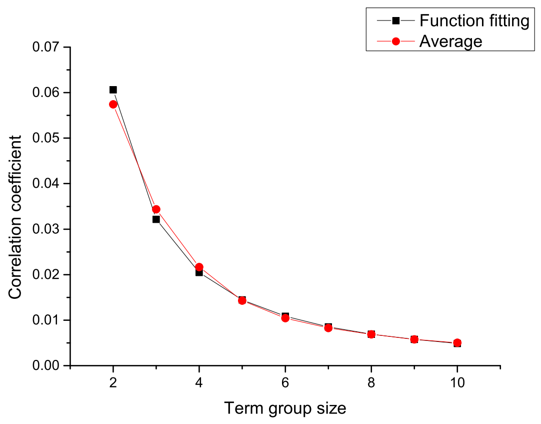
  For each off-line query q, we can get the size of intersection set $|R(q)|$, then $k(q)$ can be computed. We compute the average $k(q)$ of the queries whose size are same, and use a function $k'(|q|)$ fitting the average values to compute parameter $k(q)$.

3.2 Experimental results

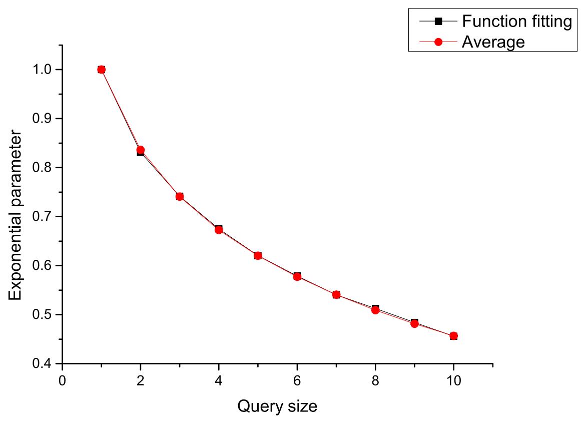
We use the TREC GOV [13] data set which contains about 1.25 million documents and Terabyte 2006 [3] query set which contains about 100 thousands queries. We use different elementary functions to fit the statistical average. The best fitting result is shown in Fig. 1 and the function is as follows:

$$\rho(g,t) = \rho'(|g \cup \{t\}|) = 0.1792 \cdot |g \cup \{t\}|^{-1.564} \tag{14}$$

$$k(q) = k'(|q|) = 1 - 0236 \cdot \ln |q| \tag{15}$$



(a) Correlation model



(b) Exponential model

Fig. 1: Average and function fitting values of parameters in correlation and exponential model

Table 1: Linear regression analysis of result set size and predicted value

Model	Average	Linear correlation
ORACLE	3011	1.000
Independent	1081	0.881
Correlation	1551	0.904
Exponential	3236	0.943

The fitting and average values are close for correlation model and almost equal for exponential model.

Table. 1 shows the results of our predictions. Because the main target of this paper is to predict early termination, we compare the linear correlations instead of errors. ORACLE is the statistical results of real values. We can find that the computed value of independent model is obviously lower, which prove the correlation between different terms. The lower average value of correlation model is because of the mathematical properties of Eq. (12). Overall, the predicted value saves most messages of real value.

4 Early Termination Prediction

4.1 Features selection

Given a top-k query q and a boundary document d_{max}, from the input, we should extract the features of document d_{max} and query $|q|$. The score of document d_{max} is a part of terminating condition. The length of query q is an important query property and term IDFs appear in the ranking function. Next, thinking of query processing, the condition that the position of early termination before d_{max} satisfies existing as least k documents $d_1, d_2, d_3, \ldots, d_k$ fulfill $S(d_i, q) \geq GS(d_{max})$, where $i = 1, 2, 3, \ldots, k$. For a random document d, if the probability $S(d, q) >$

$GS(d_{max})$ is p, then the probability of early termination P_{ET} is as follows:

$$P_{ET} = \sum_{i=k}^{|R(q)|} \binom{|R(q)|}{i} p^i (1-p)^{|R(q)|-i} \tag{16}$$

where we can get the conclusion that P_{ET} is highly related to $|R(q)|$ and p, where $|R(q)|$ is the size of the intersection set and p is related to the score distribution of the documents in the intersection set.

The features are divided into four groups and listed in Table. 2, where SR_q, IR_q, S_q, GS_q are random variables representing the scores of a random document in the intersection set of query q. It is impossible to calculate score distribution features for a query exactly without processing the query, but we can compute and save the features of each term off-line, then based on some assumptions, estimate using the values of term. Given any two terms t_1 and t_2, SR_{t_1} and SR_{t_2}, GS_{t_1} and GS_{t_2} are perfectly correlate for same documents, where random variables SR_t, IR_t and GS_t representing the scores of a random document in the inverted list of term t. In addition, IR_{t_1} and IR_{t_2} are assumed perfectly independent of each other. And given a query q, SR_q and IR_q are assumed perfectly independent of each other. When extracting the features of score distribution, we suppose the random variables satisfy the following formulas:

$$SR_q = \frac{1}{|q|}\sum_{t\in q} SR_t,\ IR_q = \frac{\sum_{t\in q}\omega_t \cdot IR_t}{\sum_{t\in q}\omega_t},\ S_q = \alpha \cdot SR_q + \beta \cdot IR_q,\ GS_q = \frac{1}{|q|}\sum_{t\in q} GS_t.$$

Table 2: Feature groups

Feature group	Feature
Boundary features	$docId_{d_{max}}$, $SR(d_{max})$, $GS(d_{max})$
Query features	$\|q\|$, $\max_{t\in q}\omega_t$, $\max_{t\in q}\omega_t$, $\text{average}_{t\in q}\,\omega_t$ and $\text{variance}_{t\in q}\,\omega_t$
Intersection set size features	The predicted value of all three models
Score distribution features	Expectation and variance of SR_q, IR_q, S_q, GS_q

4.2 Experimental results

The documents data set is TREC GOV. We run top-3 query processing, selected 16000 queries from Terabyte 2006. The queries satisfy that half of them can lead to early termination and half not. We used the bagging method and REP classification tree provided by WEKA package [4], half of queries for training and the other half for testing. If true class is early termination, the result may be the following four cases:

- **True positive (TP)** Predicted value is true and real value is true.
- **True negative (TN)** Predicted value is false and real value is false.
- **False positive (FP)** Predicted value is true but real value is false.

- **False negative (FN)** Predicted value is false but real value is true.

The boundary features must be chosen as input parameter. Using different combinations of other three groups, the results are shown in Table 3, where “Q” means query features. “N” means intersection set size features, “S” means score features, “+” means using the real intersection set size instead of the estimated values. The average accuracy rate of “QNS” is about 0.014 more than “NS” and “QS”, which is an obvious gap when the accuracy rate is above 0.8. It shows both “N” and “S” are important features to predict early termination. The results of “QNS” and “NS”, “QN” and “N” are similar, which shows that “Q” and “N” are overlapped each other. That is because query length and term IDFs are related to intersection set size. Besides, feature group “Q” also contains a little information of scores. But “QNS” is obviously better than “QS”, which shows that “N” is more essential and important than “Q”. If we only use one of “N”, “Q” and “S”, we can find that the results of “N” and “Q” is better than “S”. A possible reason is that the intersection set sizes of different queries vary widely but the score distributions are similar. Using all features, TP rate reaches to 0.800 and TN rate reaches to 0.849. Comparing with only using query features, the average error rate declines 8.3% relatively. But it is obvious that the results of “QNS+” is much better than “QNS”. That means the accuracy rate of early termination prediction may be improved greatly if we can estimate the intersection set size more accurate.

Table 3: Classification results using different features

	TP rate	TN rate	FP rate	FN rate
ORACLE	1.000	1.000	0.000	0.000
QNS+	0.870	0.867	0.133	0.130
QNS	0.800	0.849	0.151	0.200
NS	0.797	0.848	0.152	0.203
QS	0.805	0.816	0.184	0.195
QN	0.778	0.842	0.158	0.222
N	0.785	0.834	0.166	0.215
Q	0.776	0.841	0.159	0.224
S	0.810	0.762	0.238	0.190

5 Conclusions

In this paper, we proposed mathematical models based on different hypotheses to predict the intersection set size. On the basis of that, we analyze the terminating condition to get the important factors affecting early termination, extract relevant features and propose a machine learning method to predict early termination. The experimental results show that our prediction method reach a high level of accuracy. Comparing the results using different features, an obvious conclusion is that, the estimation of intersection set size play an important role in early termination prediction. In the future work, we will study on the optimization strategies based on early termination prediction. In addition, we can study on the prediction method for other early termination algorithms.

References

[1] V. N. Anh, O. de Kretser, and A. Moffat, Vector-space ranking with effective early termination, in it Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, 2001, pp. 35-42.

[2] V. N. Anh and A. Moffat, Pruned query evaluation using pre-computed impacts, in Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, 2006, pp. 372-379.

[3] S. Bttcher, C. L. Clarke, and I. Soboroff, The TREC 2006 Terabyte Track, in TREC, 2006, p. 39.

[4] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, The WEKA data mining software: an update, ACM SIGKDD explorations newsletter, Vol. 11, pp. 10-18, 2009.

[5] M. Jeon, Y. He, S. Elnikety, A. L. Cox, and S. Rixner, Adaptive parallelism for web search, in Proceedings of the 8th ACM European Conference on Computer Systems, 2013, pp. 155-168.

[6] X. Long and T. Suel, Optimized query execution in large search engines with global page ordering, in Proceedings of the 29th international conference on Very large data bases-Volume 29, 2003, pp. 129-140.

[7] C. Macdonald, N. Tonellotto, and I. Ounis, Learning to predict response times for online query scheduling, in Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval, 2012, pp. 621-630.

[8] M. Persin, J. Zobel, and R. Sacks-Davis, Filtered document retrieval with frequency-sorted indexes, JASIS, Vol. 47, pp. 749-764, 1996.

[9] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford, Okapi at TREC-3, NIST Special Publication, pp. 109-109, 1995.

[10] R. Schenkel, A. Broschart, S. Hwang, M. Theobald, and G. Weikum, Efficient text proximity search, in String Processing and Information Retrieval, 2007, pp. 287-299.

[11] T. Strohman and W. B. Croft, Efficient document retrieval in main memory, in Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, 2007, pp. 175-182.

[12] H. Turtle and J. Flood, Query evaluation: strategies and optimizations, Information Processing & Management, Vol. 31, pp. 831-850, 1995.

[13] E. M. Voorhees, Overview of TREC 2003, in TREC, 2003, pp. 1-13.

[14] H. Yan, S. Shi, F. Zhang, T. Suel, and J. R. Wen, Efficient term proximity search with term-pair indexes, in Proceedings of the 19th ACM international conference on Information and knowledge management, 2010, pp. 1229-1238.

[15] F. Zhang, S. Shi, H. Yan, and J. R. Wen, Revisiting globally sorted indexes for efficient document retrieval, in Proceedings of the third ACM international conference on Web search and data mining, 2010, pp. 371-380.