Searching Documents Based on Relevance and Type

Jun Xu[1], Yunbo Cao[1], Hang Li[1], Nick Craswell[2], and Yalou Huang[3]

[1] Microsoft Research Asia, No. 49 Zhichun Road, Beijing, China
[2] Microsoft Research Cambridge, UK
[3] Nankai University, No. 94 Weijin Road, Tianjin, China
{junxu, Yunbo.Cao, hangli, nickcr}@microsoft.com,
yellow@nankai.edu.cn

Abstract. This paper extends previous work on document retrieval and document type classification, addressing the problem of 'typed search'. Specifically, given a query and a designated document type, the search system retrieves and ranks documents not only based on the relevance to the query, but also based on the likelihood of being the designated document type. The paper formalizes the problem in a general framework consisting of 'relevance model' and 'type model'. The relevance model indicates whether or not a document is relevant to a query. The type model indicates whether or not a document belongs to the designated document type. We consider three methods for combing the models: linear combination of scores, thresholding on the type score, and a hybrid of the previous two methods. We take course page search and instruction document search as examples and have conducted a series of experiments. Experimental results show our proposed approaches can significantly outperform the baseline methods.

1 Introduction

Traditionally, the document search problem can be described as follows. The user submits a query to the search system and the search system attempts to return documents that the user will find relevant. In many cases, the user not only has an idea of what 'document content' they are looking for, but also what 'type of document'. For example, sometimes users know that they want to search for information from technical papers, homepages, or instruction documents.

In this paper we consider a setting for search, which we refer to as 'typed search'. In typed search, we ask the user to enter a query as usual and at the same time allow them to designate the document type which they want. Then the system returns documents that are not only relevant to the query, but also likely to be of the designated type. Assuming the user indeed will be more satisfied by documents of the requested type, a typed search system is potentially more effective than a system where the user can not specify a document type.

Web search engines like Google, MSN Search, and Yahoo! already provide search by type features, but usually in cases where a perfect distinction can be made between types. For example, by selecting scopes such as Image search or News search the user can specify the type of result that they require. Similarly there may be operators such as 'filetype:pdf'. Not all document types are easily specified, and in some cases it

G. Amati, C. Carpineto, and G. Romano (Eds.): ECIR 2007, LNCS 4425, pp. 629 – 636, 2007.

may even be difficult for humans. In this paper we consider useful document types where it is not easy to make a perfect type classifier.

A number of papers considered the need for document genre classification and the development of type classifiers[14],[15]. This paper considers the problem of typed search, assuming that a type has already been identified that is helpful to end-users and a classifier can be developed. For this specific search problem past research was limited to search on special types of documents such as homepage [1],[2],[7],[13].

This paper aims at being a thorough investigation of this search problem. We choose two document types as examples in our experiments, namely course page search (search for course web pages of colleges) and instruction document search (search for instruction documents). We try to answer the following three questions which we think are crucial for constructing typed search systems. (1) Is it possible to develop a general framework for typed search? (2) What is the best strategy for combining the relevance information and type information? (3) Is it possible to construct typed search systems that are easily to be extended to different types and are easily adapted to different domains?

For our general framework, we propose the use of two probabilistic models for typed search: relevance model and type model. The former represents the relevance of documents to queries, and the latter represents the likelihood of documents of being the designated type. In our experiments we use BM25 as relevance model and Logistic Regression as type model.

We propose three methods for combing the relevance and type models: linear combination, thresholding, and a hybrid method using both thresholding and linear combination. We found that linear combination and thresholding can work well with the default parameter settings. Hybrid can take advantage of the other two strategies and works best, but it needs parameter tuning.

Our methods outperform the baselines on both instruction document search and course page search. It is also possible to conduct domain adaptation, applying a type model trained on one corpus to a separate corpus. Therefore, it is feasible to create generic typed search systems.

2 Related Work

Our work on document types is related to work on identifying and classifying documents by genre, such as [6],[10][14],[15]. We use the term ‘type’ to indicate maximum generality. Our framework can be applied for any given type of document, even if not everyone would agree that it constitutes a distinct ‘genre’. Some authors use ‘genre’ and ‘type’ interchangeably, defining genre as: a document type based on similarity of form and purpose [4].

Homepage search can be regarded as a specific typed search. Much research work was conducted on that issue. For instance, TREC had a task called ‘home/named page finding’[2]. Many systems were developed for the task [1],[7],[13]. In homepage search, both relevance information and type information are needed in web pages ranking. For example, information about the URL can be used to indicate type [13], because homepages tend to have shallow URLs ending in ‘/’. In [16], users have the option of specifying some concepts (e.g., a catalog, a call for paper, etc.) of interest when submitting a query.

Table 1. Two views of documents

	Relevant	Irrelevant
Designated type	**A**	B
Not designated type	C	D

Our experiments apply BM25[11] and Logistic Regression[5]. BM25 was introduced as part of Okapi, a system for document retrieval with a probabilistic approach. Specifically, BM25 attempts to rank documents in order of decreasing probability of relevance. Logistic Regression (LR) is one model for classification. LR outputs probability values rather than confidence scores in classification.

3 Problem Description

The search system has a mechanism allowing users to designate the types of documents which users can search. The type of a document (or web page) represents the genre or the functional category of the document. Users can use a menu or a special search operator to designate document types.

If the user knows what type of document they wish to find, they can select that type. They then type a search query as usual. The search system receives the query and the document type. It automatically retrieves and ranks documents on the basis of not only relevance but also likelihood of being the specified type.

Typed search is useful for helping users to find information. Traditional information retrieval conducts search on the basis of relevance of documents to the query [8],[9][12]. Similarly, typed search needs to assure that the retrieved documents are relevant to the query. However, typed search also needs to assure that the retrieved documents belong to the designated document type. Table 1 shows two views of documents. From Table 1, we see that **A** is the set of documents that we want to collect in typed search. By introducing types into search, one can drastically reduce the numbers of documents returned to users.

Various document types can be considered such as resume, blog, homepage, email, etc. For a recent example of a detailed study of document types, see [4]. Here, we assume that the search provider (e.g., librarian, search engine designer) can identify typed searches that are valuable to users, and apply our typed search approach.

4 Our Approach

4.1 General Framework

We propose a general framework for ranking in typed search. Given a query q and a document d, we rank the documents with the conditional probability $\Pr(r,t \mid q,d)$, where r and t take 1 or 0 as values and they denote 'relevant or not' and 'in the same type or not'. In instruction document search, for example, $t=1$ means that a document is an instruction document. In typed search, we rank documents using the probability scores of documents.

Here, we assume that r and t are conditionally independent given q and d. We further assume that t is only dependent on d, not on q. Hence we have,

$$\Pr(r,t \mid q,d) \approx \Pr(r \mid q,d) \cdot \Pr(t \mid q,d) \approx \Pr(r \mid q,d) \cdot \Pr(t \mid d) \quad (1)$$

We take Equation (1) as a general model for typed search. We call the two sub-models $\Pr(r \mid q,d)$ and $\Pr(t \mid d)$ 'relevance model' and 'type model', respectively. The relevance model judges whether or not a document is relevant to the query. The type model judges whether or not a document is in the designated document type.

4.2 Relevance Model and Type Model

Given a query and a document, the relevance model outputs a relevance score. In typed search, for a given query, we create a list of <*document, relevance_score*> pairs using the relevance model.

In this paper, we employ BM25[11] as the relevance model. In practice, for indexing, we index the title and the body of a document separately, calculate a BM25 score for each, then linearly combine the scores. We view this combination of scores for the title and the body as the *relevance_score*.

Given a document, the type model outputs a type score. In typed search, we create a list of <*document, type_score*> pairs using the type model.

We take a statistical machine learning approach to constructing a type model. More specifically, given a training data set $D = \{x_i, y_i\}_1^n$, we construct a model $\Pr(y \mid x)$ that can minimize the number of errors when predicting y given x (generalization error). Here $x_i \in X$ and $y_i \in \{1,-1\}$. x represents a document and y represents whether or not a document is a document in the designated type. When applied to a new document x, the model predicts the corresponding y and outputs the score of the prediction. In this paper, we adopt Logistic Regression[5] as our type model. Logistic Regression calculates the 'type probability' of $\Pr(y=1 \mid x)$ a document.

In our approach, we actually use the *type_score* of a document:

$$type\_score = \log \frac{\Pr(y=1 \mid x)}{1-\Pr(y=1 \mid x)} \quad (2)$$

4.3 Combining Strategy

We propose three strategies for combing the scores calculated by the relevance and type models. They are linear combination, thresholding, and hybrid respectively. We rank documents using the combined scores in typed search.

In linear combination, we calculate *ranking_score* by linearly interpolating *relevance_score* and *type_score*.

$$ranking\_score = \lambda \cdot type\_score + (1-\lambda) \cdot relevance\_score, \quad (3)$$

where $\lambda \in [0,1]$ is a parameter. In thresholding, we calculate *ranking_score* by descretizing *type_score* to 1 or 0 based on a predetermined threshold.

$$ranking\_score = \begin{cases} relevance\_score & \text{if } Pr(y=1|x) > \theta \\ 0 & \text{otherwise} \end{cases}. \quad (4)$$

Here $\theta \in [0,1]$ is threshold. In cases where we are certain of our *type_score* we can apply a strict threshold, as is the case for News search on the Web (all pages that are

not news are strictly filtered out). If we are less confident about our *type_score* we can apply a lower threshold or try the linear combination strategy.

In hybrid method, we calculate *ranking_score* using both linearly combination and thresholding. Here $\lambda \in [0,1]$ and $\theta \in [0,1]$ are parameters.

$$ranking\_score = \begin{cases} \lambda \cdot type\_score + (1-\lambda) \cdot relevance\_score & \text{if } Pr(y=1 \mid x) > \theta \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

How to determine the parameter values (λ and θ) is an issue we need to consider. In linear combination, by default can be set as 0.5, since Equation (3) is equivalent to Equation (1), when λ=0.5. In thresholding, θ can also be set as 0.5 by default, since the document is likely to belong to the type when $Pr(y=1|x)>0.5$. In our experiments, we found that nearly best search performances can be achieved when λ and θ are 0.5. In the hybrid method, however, λ and θ have to be tuned empirically.

5 Experiments

In our first experiment, the document type is 'course page', a page describing a course, as would be available on a university website. In the second experiment, the document type is 'instruction document', for example an online manual.

As the first baseline method, we solely use the scores of BM25 to conduct ranking. This can also be seen as an extreme case of our proposed approach in which the parameter λ in Equation (3) equals 0.

As the second baseline method, we solely use the scores of Logistic Regression in ranking, as explained in Section 4.2. This baseline is the other extreme case of our proposed approach, when the parameter λ in Equation (3) equals 1.

As the third baseline method, we add keywords to queries and employ BM25. For course search, we combine the original query words with the keyword 'course' to generate a new query. (We tried to use other keywords, but they did not work well). For instruction document search, we combine the query with 'howto' and 'how to'.

As the fourth baseline method, we use BM25 to conduct ranking, then employ rules to filter the results. As rules, we implement the major features in the type model.

The third and fourth baselines are the simplest ways of combining type information and relevance information in typed search ranking. Hereafter, we denote the four baseline methods as 'BM25', 'Logistic Regression', 'BM25 + Keyword', and 'BM25 + Heuristics'. (We denote our methods as 'Combined (linear)', 'Combined (thresholding)', and 'Combined (hybrid)'.)

We make use of MAP (Mean Average Precision) and *MRR* (Mean Reciprocal Rank) for evaluation of typed search.

5.1 Course Page Search

Data Set

We used 'Four Universities Data Set'[1] in the experiment. The 8,282 WWW-pages in the data set were assigned to six category labels (Course, Student, Staff, Department, Project, and Other). In our case, course pages are positive examples.

[1] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/

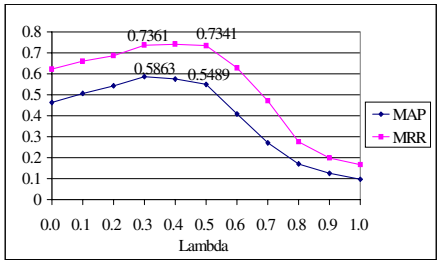

Fig. 1. Performance linear combination w.r.t. λ for course page search.

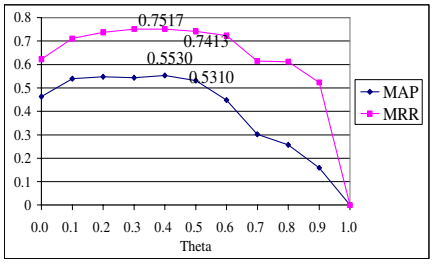

Fig. 2. Performance of thresholding w.r.t. θ for course page search.

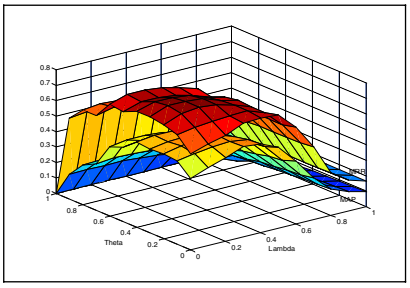

Fig. 3. Performance of hybrid w.r.t. λ and θ for course page search

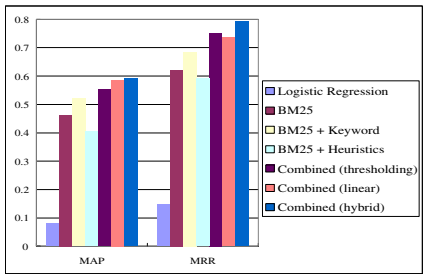

Fig. 4. Performance of course page search

As search queries, we collected the course names from the web sites of the Computer Science department of CMU[2] and MIT Open Courseware, Electrical Engineering and Computer Science[3].For each query, we retrieved at most top 100 course web pages. The retrieved web pages were judged manually by human annotators whether they are really relevant to the query one by one. In this way, all the retrieved documents for each query got the labels **A**, **B**, or **C-D** as in Section 3. Our final query set contains 52 queries. On average, a query has 7.8 relevant web documents.

Experiment on Typed Search

We compared the performances of the typed search ranking methods. We randomly divided queries into four even subsets and carried out 4-fold cross-validation. The result reported below are thus those averaged over the four trials.

We tried various values for the parameter λ and θ in our methods of linear combination, thresholding and hybrid. Fig. 1, Fig. 2, and Fig. 3 show the performance curves when the parameters changes. The best result is accomplished by hybrid, if we happen to know the parameter values.

Fig. 4 shows the results of our proposed methods (linear combination, thresholding, and hybrid method) together with baseline methods. Our methods perform much

[2] https://acis.as.cmu.edu/gale2/open/Schedule/SOCServlet?Formname=ByDept
[3] http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/index.htm

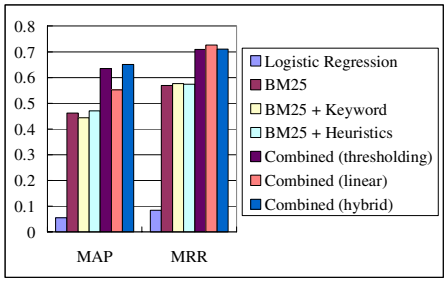

Fig. 5. Performance of instruction search

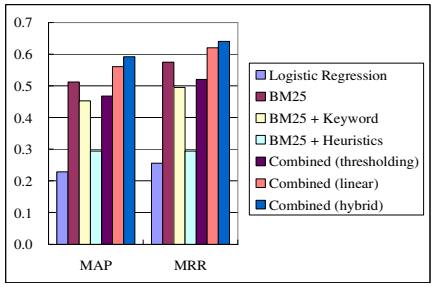

Fig. 6. Performance of domain adaptation.

better than baselines. The results indicate that the traditional information retrieval approach cannot solve the problem of 'typed search' well. Our approach of combing type information and relevance information is effective.

It is not surprising to see that the baseline 'BM25' cannot work well for the task, because it is designed for search of relevant documents. As we have discussed in Section 3, typed search needs consider both relevance and document type. For a similar reason, it is also not surprising to observe that the baseline 'Logistic Regression' cannot achieve good result.

For 'BM25+Keyword', it is hard to construct a query that can filter out documents that not belong to the desired type. For 'BM25+Heuristics', it is hard to make a combination between BM25 and rules. Thus, employing probabilistic models for both relevance and type ranking, as in our approach, appears to be a reasonable choice.

5.2 Instruction Document Search

In the experiment, we investigated typed search applied to instruction document. We created a document set by crawling from the intranet of an international company. We also collected all the real queries about instruction document search from the query log of a search engine on the intranet.

We created a dataset which contains 50 queries, similar to Section 5.1. 61 documents are labeled with **A**, 352 with **B,** and the others with **C-D**. We conducted experiment with 5-fold cross-validation. The results are reported in Fig. 5. Our methods outperform baseline methods for instruction document search.

5.3 Domain Adaptation for Instruction Document Search

In the experiment, we tested whether a generic model can be constructed for typed search. Specifically, we investigated whether a type model trained on intranet can still work well on TREC W3C corpus [3].

We created a data set which contains 50 queries, similar to Section 5.1. Among the documents, 74 are labeled with **A**, 361 with B and others with **C-D**. Fig. 6 shows the results. The experimental results show that our method achieves good result on the TREC W3C corpus, although the type model is trained on a different domain.

6 Conclusions

In the paper, we have studied 'typed search' – search of documents based not only on relevance, but also document type. Our typed search framework combines two models: a relevance model and type model. We employed BM25 and Logistic Regression as the relevance model and the type model, respectively. Three methods are proposed for combining the models to obtain a final ranking score. Using course page search and instruction document search as examples, we have conducted experiments with real-world data. Experimental results indicate that our proposed approaches are effective for typed search and perform significantly better than the baselines. Experimental results also indicate that our proposed approach can perform consistently well across different domains.

References

1. Craswell, N., Hawking, D., and Robertson S.E.: Effective site finding using link anchor information. In: Proc. of the 24th SIGIR conference, New Orleans, USA. (2001), 250-257
2. Craswell, N. and Hawking, D.: Overview of the TREC-2004 Web Track. In: NIST Special Publication: 500-261, the Thirteen Text REtrieval Conference. (2004)
3. Craswell, N., Vries, A., and Soboroff, I.: Overview of the TREC-2005 Enterprise Track. The Fourteenth Text Retrieval Conference. (2005)
4. Freund, L., Toms, E.G., and Clarke, C.L.: Modeling task-genre relationships for IR in the workplace. In: Proc. of the 28th ACM SIGIR Conference, Salvador, Brazil. (2005)
5. Hastie, T., Tibshirani, R., and Friedman, J.: The Elements of Statistical Learning. Springer, New York (2001)
6. Kessler, B., Nunberg G., and Schutze, H.: Automatic Detection of Text Genre. In: Proc. of the 35th Association for Computational Linguistics, Madrid, Spain. (1997), 32–38
7. Kraajj, W., Westerveld, T., and Hiemstra, D.: The Importance of Prior Probabilities for Entry Page Search. In: Proc. of the 25th ACM SIGIR conference. (2002)
8. Mizzaro, S: Relevance: The whole history. Journal of the American Society for Information Science 48(9): (1997), 810-832
9. Mizzaro, S: How many relevancies in information retrieval? Interacting With Computers, 10(3):305-322, Vol. 10. (1998) 321-351.
10. Rauber, A. and Müller-Kögler, A.: Integrating automatic genre analysis into digital libraries. In: Proc. of the 1st ACM/IEEE Joint Conf. on Digital Libraries. Virginia, USA. (2001), 1-10.
11. Robertson, S.E., Walker, S., Beaulieu, M.M., Gatford, M., and Payne, A.: Okapi at TREC-4. In: Proc. of the 4th Text REtrieval Conference. (1996), 73-96
12. Salton, G., McGill, M.: Introduction to Modern Information Retrieval. McGraw-Hill (1983).
13. Song, R., Wen, J.R., Shi, S., Xin, G., Liu, T.Y., Qin, T., Zheng, X., Zhang, J., Xue, G., and Ma, W.Y.: Microsoft Research Asia at Web Track and Terabyte Track of TREC 2004. In: NIST Special Publication: 500-261, the 13th Text REtrieval Conference. (2004)
14. Voorhees, E.: Overview of the TREC 2003 Question Answering Track. In: Proc. of the 12th Annual Text Retrieval Conference. (2003).
15. Zaragoza, H., Craswell, N., Taylor, M., Saria, S., and Robertson, S.: Microsoft Cambridge at TREC 13: Web and Hard Tracks. In the 13th Text REtrieval Conference. (2004)
16. Matsuda, K. and Fukushima, T.: Task Oriented World Wide Web Retrieval by Document Type Classification. In: Proc. of the 8th CIKM. Kansas, USA. (1999)