Using Query Contexts in Information Retrieval

Jing Bai [1], Jian-Yun Nie [1], Hugues Bouchard [2], Guihong Cao [1]

[1] Department IRO, University of Montreal
CP. 6128, succursale Centre-ville, Montreal, Quebec, H3C 3J7, Canada
{baijing, nie, caogui}@iro.umontreal.ca

[2] Yahoo! Inc.
Montreal, Quebec, Canada
bouchard@yahoo-inc.com

ABSTRACT

User query is an element that specifies an information need, but it is not the only one. Studies in literature have found many contextual factors that strongly influence the interpretation of a query. Recent studies have tried to consider the user's interests by creating a user profile. However, a single profile for a user may not be sufficient for a variety of queries of the user. In this study, we propose to use *query-specific contexts* instead of user-centric ones, including *context around query* and *context within query*. The former specifies the environment of a query such as the domain of interest, while the latter refers to context words within the query, which is particularly useful for the selection of relevant term relations. In this paper, both types of context are integrated in an IR model based on language modeling. Our experiments on several TREC collections show that each of the context factors brings significant improvements in retrieval effectiveness.

Categories and Subject Descriptors

H.3.3 [**Information storage and retrieval**]: Information Search and Retrieval – Retrieval Models

General Terms

Algorithms, Performance, Experimentation, Theory.

Keywords

Query contexts, Domain model, Term relation, Language model.

1. INTRODUCTION

Queries, especially short queries, do not provide a complete specification of the information need. Many relevant terms can be absent from queries and terms included may be ambiguous. These issues have been addressed in a large number of previous studies. Typical solutions include expanding either document or query representation [19][35] by exploiting different resources [24][31], using word sense disambiguation [25], etc. In these studies, however, it has been generally assumed that query is the only element available about the user's information need. In reality, query is always formulated in a search context. As it has been found in many previous studies [2][14][20][21][26], contextual factors have a strong influence on relevance judgments. These factors include, among many others, the user's domain of interest, knowledge, preferences, etc. All these elements specify the contexts around the query. So we call them *context around query* in this paper. It has been demonstrated that user's query should be placed in its context for a correct interpretation.

Recent studies have investigated the integration of some contexts around the query [9][30][23]. Typically, a user profile is constructed to reflect the user's domains of interest and background. A user profile is used to favor the documents that are more closely related to the profile. However, a single profile for a user can group a variety of different domains, which are not always relevant to a particular query. For example, if a user working in computer science issues a query "*Java hotel*", the documents on "*Java language*" will be incorrectly favored. A possible solution to this problem is to use query-related profiles or models instead of user-centric ones. In this paper, we propose to model topic domains, among which the related one(s) will be selected for a given query. This method allows us to select more appropriate query-specific context around the query.

Another strong contextual factor identified in literature is domain knowledge, or domain-specific term relations, such as "*program→computer*" in computer science. Using this relation, one would be able to expand the query "*program*" with the term "*computer*". However, domain knowledge is available only for a few domains (e.g. "*Medicine*"). The shortage of domain knowledge has led to the utilization of general knowledge for query expansion [31], which is more available from resources such as thesauri, or it can be automatically extracted from documents [24][27]. However, the use of general knowledge gives rise to an enormous problem of knowledge ambiguity [31]: we are often unable to determine if a relation applies to a query. For example, usually little information is available to determine whether "*program→computer*" is applicable to queries "*Java program*" and "*TV program*". Therefore, the relation has been applied to all queries containing "*program*" in previous studies, leading to a wrong expansion for "*TV program*".

Looking at the two query examples, however, people can easily determine whether the relation is applicable, by considering the context words "*Java*" and "*TV*". So the important question is how we can serve these context words in queries to select the appropriate relations to apply. These context words form a *context within query*. In some previous studies [24][31], context words in a query have been used to select expansion terms suggested by term relations, which are, however, context-independent (such as "*program→computer*"). Although improvements are observed in some cases, they are limited. We argue that the problem stems from the lack of necessary context information in relations themselves, and a more radical solution lies in the addition of contexts in relations. The method we propose is to add context words into the condition of a relation, such as "{*Java, program*} *→computer*", to limit its applicability to the appropriate context.

SIGIR'07, July 23–27, 2007, Ámsterdam, The Netherlands.

This paper aims to make contributions on the following aspects:

- Query-specific domain model: We construct more specific domain models instead of a single user model grouping all the domains. The domain related to a specific query is selected (either manually or automatically) for each query.
- Context within query: We integrate context words in term relations so that only appropriate relations can be applied to the query.
- Multiple contextual factors: Finally, we propose a framework based on language modeling approach to integrate multiple contextual factors.

Our approach has been tested on several TREC collections. The experiments clearly show that both types of context can result in significant improvements in retrieval effectiveness, and their effects are complementary. We will also show that it is possible to determine the query domain automatically, and this results in comparable effectiveness to a manual specification of domain.

This paper is organized as follows. In section 2, we review some related work and introduce the principle of our approach. Section 3 presents our general model. Then sections 4 and 5 describe respectively the domain model and the knowledge model. Section 6 explains the method for parameter training. Experiments are presented in section 7 and conclusions in section 8.

2. CONTEXTS AND UTILIZATION IN IR

There are many contextual factors in IR: the user's domain of interest, knowledge about the subject, preference, document recency, and so on [2][14]. Among them, the user's domain of interest and knowledge are considered to be among the most important ones [20][21]. In this section, we review some of the studies in IR concerning these aspects.

Domain of interest and *context around query*

A domain of interest specifies a particular background for the interpretation of a query. It can be used in different ways. Most often, a user profile is created to encompass all the domains of interest of a user [23]. In [5], a user profile contains a set of topic categories of ODP (Open Directory Project, http://dmoz.org) identified by the user. The documents (Web pages) classified in these categories are used to create a term vector, which represents the whole domains of interest of the user. On the other hand, [9][15][26][30], as well as Google Personalized Search [12] use the documents read by the user, stored on user's computer or extracted from user's search history. In all these studies, we observe that a single user profile (usually a statistical model or vector) is created for a user without distinguishing the different topic domains. The systematic application of the user profile can incorrectly bias the results for queries unrelated to the profile. This situation can often occur in practice as a user can search for a variety of topics outside the domains that he has previously searched in or identified.

A possible solution to this problem is the creation of multiple profiles, one for a separate domain of interest. The domains related to a query are then identified according to the query. This will enable us to use a more appropriate query-specific profile, instead of a user-centric one. This approach is used in [18] in which ODP directories are used. However, only a small scale experiment has been carried out. A similar approach is used in [8], where domain models are created using ODP categories and user queries are manually mapped to them. However, the experiments showed variable results. It remains unclear whether domain models can be effectively used in IR.

In this study, we also model topic domains. We will carry out experiments on both automatic and manual identification of query domains. Domain models will also be integrated with other factors. In the following discussion, we will call the topic domain of a query a *context around query* to contrast with another *context within query* that we will introduce.

Knowledge and *context within query*

Due to the unavailability of domain-specific knowledge, general knowledge resources such as Wordnet and term relations extracted automatically have been used for query expansion [27][31]. In both cases, the relations are defined between two single terms such as "$t_1 \rightarrow t_2$". If a query contains term t_1, then t_2 is always considered as a candidate for expansion. As we mentioned earlier, we are faced with the problem of relation ambiguity: some relations apply to a query and some others should not. For example, "*program*$\rightarrow$*computer*" should not be applied to "*TV program*" even if the latter contains "*program*". However, little information is available in the relation to help us determine if an application context is appropriate.

To remedy this problem, approaches have been proposed to make a selection of expansion terms after the application of relations [24][31]. Typically, one defines some sort of global relation between the expansion term and the whole query, which is usually a sum of its relations to every query word. Although some inappropriate expansion terms can be removed because they are only weakly connected to some query terms, many others remain. For example, if the relation "*program*$\rightarrow$*computer*" is strong enough, "*computer*" will have a strong global relation to the whole query "*TV program*" and it still remains as an expansion term.

It is possible to integrate stronger control on the utilization of knowledge. For example, [17] defined strong logical relations to encode knowledge of different domains. If the application of a relation leads to a conflict with the query (or with other pieces of evidence), then it is not applied. However, this approach requires encoding all the logical consequences including contradictions in knowledge, which is difficult to implement in practice.

In our earlier study [1], a simpler and more general approach is proposed to solve the problem at its source, i.e. the lack of context information in term relations: by introducing stricter conditions in a relation, for example "{*Java, program*}$\rightarrow$*computer*" and "{*algorithm, program*}$\rightarrow$*computer*", the applicability of the relations will be naturally restricted to correct contexts. As a result, "*computer*" will be used to expand queries "*Java program*" or "*program algorithm*", but not "*TV program*". This principle is similar to that of [33] for word sense disambiguation. However, we do not explicitly assign a meaning to a word; rather we try to make differences between word usages in different contexts. From this point of view, our approach is more similar to word sense discrimination [27].

In this paper, we use the same approach and we will integrate it into a more global model with other context factors. As the context words added into relations allow us to exploit the word context within the query, we call such factors *context within query.* Within query context exists in many queries. In fact, users

often do not use a single ambiguous word such as "*Java*" as query (if they are aware of its ambiguity). Some context words are often used together with it. In these cases, contexts within query are created and can be exploited.

Query profile and other factors

Many attempts have been made in IR to create query-specific profiles. We can consider implicit feedback or blind feedback [7][16][29][32][35] in this family. A short-term feedback model is created for the given query from feedback documents, which has been proven to be effective to capture some aspects of the user's intent behind the query. In order to create a good query model, such a query-specific feedback model should be integrated.

There are many other contextual factors ([26]) that we do not deal with in this paper. However, it seems clear that many factors are complementary. As found in [32], a feedback model creates a local context related to the query, while the general knowledge or the whole corpus defines a global context. Both types of contexts have been proven useful [32]. Domain model specifies yet another type of useful information: it reflects a set of specific background terms for a domain, for example "*pollution*", "*rain*", "*greenhouse*", etc. for the domain of "*Environment*". These terms are often presumed when a user issues a query such as "*waste cleanup*" in the domain. It is useful to add them into the query. We see a clear complementarity among these factors. It is then useful to combine them together in a single IR model.

In this study, we will integrate all the above factors within a unified framework based on language modeling. Each component contextual factor will determines a different ranking score, and the final document ranking combines all of them. This is described in the following section.

3. GENERAL IR MODEL

In the language modeling framework, a typical score function is defined in KL-divergence as follows:

$$Score(Q,D) = \sum_{t \in V} P(t \mid \theta_Q) \log P(t \mid \theta_D) \propto -KL(\theta_Q \parallel \theta_D) \quad (1)$$

where θ_D is a (unigram) language model created for a document D, θ_Q a language model for the query Q, and V the vocabulary.

Smoothing on document model is recognized to be crucial [35], and one of common smoothing methods is the Jelinek-Mercer interpolation smoothing:

$$P(t \mid \theta'_D) = (1-\lambda) P(t \mid \theta_D) + \lambda P(t \mid \theta_C) \quad (2)$$

where λ is an interpolation parameter and θ_C the collection model.

In the basic language modeling approaches, the query model is estimated by Maximum Likelihood Estimation (MLE) without any smoothing. In such a setting, the basic retrieval operation is still limited to keyword matching, according to a few words in the query. To improve retrieval effectiveness, it is important to create a more complete query model that represents better the information need. In particular, all the related and presumed words should be included in the query model. A more complete query model by several methods have been proposed using feedback documents [16][35] or using term relations [1][10][34]. In these cases, we construct two models for the query: the initial query model containing only the original terms, and a new model containing the added terms. They are then combined through interpolation.

In this paper, we generalize this approach and integrate more models for the query. Let us use θ_Q^0 to denote the original query model, θ_Q^F for the feedback model created from feedback documents, θ_Q^{Dom} for a domain model and θ_Q^K for a knowledge model created by applying term relations. θ_Q^0 can be created by MLE. θ_Q^F has been used in several previous studies [16][35]. In this paper, θ_Q^F is extracted using the 20 blind feedback documents. We will describe the details to construct θ_Q^{Dom} and θ_Q^K in Section 4 and 5.

Given these models, we create the following final query model by interpolation:

$$P(t \mid \theta_Q) = \sum_{i \in X} \alpha_i P(t \mid \theta_Q^i) \quad (3)$$

where $X=\{0, Dom, K, F\}$ is the set of all component models and α_i (with $\sum_{i \in X} \alpha_i = 1$) are their mixture weights.

Then the document score in Equation (1) is extended as follows:

$$Score(Q,D) = \sum_{t \in V} \sum_{i \in X} \alpha_i P(t \mid \theta_Q^i) \log P(t \mid \theta_D) = \sum_{i \in X} \alpha_i Score_i(Q,D) \quad (4)$$

where $Score_i(Q,D) = \sum_{t \in V} P(t \mid \theta_Q^i) \log P(t \mid \theta_D)$ is the score according to each component model. Here we can see that our strategy of enhancing the query model by contextual factors is equivalent to document re-ranking, which is used in [5][15][30].

The remaining problem is to construct domain models and knowledge model and to combine all the models (parameter setting). We describe this in the following sections.

4. CONSTRUCTING AND USING DOMAIN MODELS

As in previous studies, we exploit a set of documents already classified in each domain. These documents can be identified in two different ways: 1) One can take advantages of an existing domain hierarchy and the documents manually classified in them, such as ODP. In that case, a new query should be classified into the same domains either manually or automatically. 2) A user can define his own domains. By assigning a domain to his queries, the system can gather a set of answers to the queries automatically, which are then considered to be in-domain documents. The answers could be those that the user have read, browsed through, or judged relevant to an in-domain query, or they can be simply the top-ranked retrieval results.

An earlier study [4] has compared the above two strategies using TREC queries 51-150, for which a domain has been manually assigned. These domains have been mapped to ODP categories. It is found that both approaches mentioned above are equally effective and result in comparable performance. Therefore, in this study, we only use the second approach. This choice is also motivated by the possibility to compare between manual and automatic assignment of domain to a new query. This will be explained in detail in our experiments.

Whatever the strategy, we will obtain a set of documents for each domain, from which a language model can be extracted. If maximum likelihood estimation is used directly on these documents, the resulting domain model will contain both domain-

specific terms and general terms, and the former do not emerge. Therefore, we employ an EM process to extract the specific part of the domain as follows: we assume that the documents in a domain are generated by a domain-specific model (to be extracted) and general language model (collection model). Then the likelihood of a document in the domain can be formulated as follows:

$$P(D \mid \theta'_{Dom}) = \prod_{t \in D} [(1-\eta)P(t \mid \theta_{Dom}) + \eta P(t \mid \theta_C)]^{c(t;D)} \quad (5)$$

where $c(t; D)$ is the count of t in document D and η is a smoothing parameter (which will be fixed at 0.5 as in [35]). The EM algorithm is used to extract the domain model θ_{Dom} that maximizes $P(Dom \mid \theta'_{Dom})$ (where Dom is the set of documents in the domain), that is:

$$\theta_{Dom} = \arg\max_{\theta_{Dom}} P(Dom \mid \theta'_{Dom}) = \arg\max_{\theta_{Dom}} \prod_{D \in Dom} \prod_{t \in D} [(1-\eta)P(t \mid \theta_{Dom}) + \eta P(t \mid \theta_C)]^{c(t;D)} \quad (6)$$

This is the same process as the one used to extract feedback model in [35]. It is able to extract the most specific words of the domain from the documents while filtering out the common words of the language. This can be observed in the following table, which shows some words in the domain model of "*Environment*" before and after EM iterations (50 iterations).

Table 1. Term probabilities before/after EM

Term	Initial	Final	change	Term	Initial	Final	change
air	0.00358	0.00558	+ 56%	year	0.00357	0.00052	- 86%
environment	0.00213	0.00340	+ 60%	system	0.00212	$7.13*e^{-6}$	- 99%
rain	0.00197	0.00336	+ 71%	program	0.00189	0.00040	- 79%
pollution	0.00177	0.00301	+ 70%	million	0.00131	$5.80*e^{-6}$	- 99%
storm	0.00176	0.00302	+ 72%	make	0.00108	$5.79*e^{-5}$	- 95%
flood	0.00164	0.00281	+ 71%	company	0.00099	$8.52*e^{-8}$	- 99%
tornado	0.00072	0.00125	+ 74%	president	0.00077	$2.71*e^{-6}$	- 99%
greenhouse	0.00034	0.00058	+ 72%	month	0.00073	$3.88*e^{-5}$	- 95%

Given a set of domain models, the related ones have to be assigned to a new query. This can be done manually by the user or automatically by the system using query classification. We will compare both approaches.

Query classification has been investigated in several studies [18][28]. In this study, we use a simple classification method: the selected domain is the one with which the query's KL-divergence score is the lowest, i.e.:

$$\theta_Q^{Dom} = \arg\min_{\theta_{Dom}} \sum_{t \in Q} P(t \mid \theta_Q^0) \log P(t \mid \theta_{Dom}) \quad (7)$$

This classification method is an extension to Naïve Bayes as shown in [22]. The score depending on the domain model is then as follows:

$$Score_{Dom}(Q, D) = \sum_{t \in V} P(t \mid \theta_Q^{Dom}) \log P(t \mid \theta_D) \quad (8)$$

Although the above equation requires using all the terms in the vocabulary, in practice, only the strongest terms in the domain model are useful and the terms with low probabilities are often noise. Therefore, we only retain the top 100 strongest terms. The same strategy is used for Knowledge model.

Although domain models are more refined than a single user profile, the topics in a single domain can still be very different, making the domain model too large. This is particularly true for large domains such as "Science and technology" defined in TREC queries. Using such a large domain model as the background can introduce much noise terms. Therefore, we further construct a sub-domain model more related to the given query, by using a subset of in-domain documents that are related to the query. These documents are the top-ranked documents retrieved with the original query within the domain. This approach is indeed a combination of domain and feedback models. In our experiments, we will see that this further specification of sub-domain is necessary in some cases, but not in all, especially when Feedback model is also used.

5. EXTRACTING CONTEXT-DEPENDENT TERM RELATIONS FROM DOCUMENTS

In this paper, we extract term relations from the document collection automatically.

In general, a term relation can be represented as $A \rightarrow B$. Both A and B have been restricted to single terms in previous studies. A single term in A means that the relation is applicable to all the queries containing that term. As we explained earlier, this is the source of many wrong applications. The solution we propose is to add more context terms into A, so that it is applicable only when all the terms in A appear in a query. For example, instead of creating a context-independent relation "*Java*→*program*", we will create "{*Java*, *computer*}→*program*", which means that "*program*" is selected when both "*Java*" and "*computer*" appear in a query. The term added in the condition specifies a stricter context to apply the relation. We call this type of relation *context-dependent relation.*

In principle, the addition is not restricted to one term. However, we will make this restriction due to the following reasons:

- User queries are usually very short. Adding more terms into the condition will create many rarely applicable relations;
- In most cases, an ambiguous word such as "*Java*" can be effectively disambiguated by one useful context word such as "*computer*" or "*hotel*";
- The addition of more terms will also lead to a higher space and time complexity for extracting and storing term relations.

The extraction of relations of type "$\{t_j, t_k\} \rightarrow t_i$" can be performed using mining algorithms for association rules [13]. Here, we use a simple co-occurrence analysis. Windows of fixed size (10 words in our case) are used to obtain co-occurrence counts of three terms, and the probability $P(t_i \mid t_j t_k)$ is determined as follows:

$$P(t_i \mid t_j t_k) = c(t_i, t_j, t_k) \Big/ \sum_{t_l} c(t_l, t_j, t_k) \quad (9)$$

where $c(t_i, t_j, t_k)$ is the count of co-occurrences.

In order to reduce space requirement, we further apply the following filtering criteria:

- The two terms in the condition should appear at least certain time together in the collection (10 in our case) and they should be related. We use the following pointwise mutual information as a measure of relatedness ($MI > 0$) [6]:

$$MI(t_j, t_k) = \log \frac{P(t_j, t_k)}{P(t_j)P(t_k)}$$

- The probability of a relation should be higher than a threshold (0.0001 in our case);

Having a set of relations, the corresponding Knowledge model is defined as follows:

$$P(t \mid \theta_Q^K) = \sum_{(t_j t_k) \in Q} P(t_i \mid t_j t_k) P(t_j t_k \mid \theta_Q^0) = \sum_{(t_j t_k) \in Q} P(t_i \mid t_j t_k) P(t_j \mid \theta_Q^0) P(t_k \mid \theta_Q^0) \quad (10)$$

where $(t_j t_k) \in Q$ means any combination of two terms in the query. This is a direct extension of the translation model proposed in [3] to our context-dependent relations. The score according to the Knowledge model is then defined as follows:

$$Score_K(Q, D) = \sum_{t_i \in V} \sum_{(t_j t_k) \in Q} P(t_i \mid t_j t_k) P(t_j \mid \theta_Q^0) P(t_k \mid \theta_Q^0) \log P(t_i \mid \theta_D) \quad (11)$$

Again, only the top 100 expansion terms are used.

6. MODEL PARAMETERS

There are several parameters in our model: λ in Equation (2) and α_i ($i \in \{0, Dom, K, F\}$) in Equation (3). As the parameter λ only affects document model, we will set it to the same value in all our experiments. The value λ=0.5 is determined to maximize the effectiveness of the baseline models (see Section 7.2) on the training data: TREC queries 1-50 and documents on Disk 2.

The mixture weights α_i of component models are trained on the same training data using the following method of line search [11] to maximize the Mean Average Precision (MAP): each parameter is considered as a search direction. We start by searching in one direction – testing all the values in that direction, while keeping the values in other directions unchanged. Each direction is searched in turn, until no improvement in MAP is observed.

In order to avoid being trapped at a local maximum, we started from 10 random points and the best setting is selected.

7. EXPERIMENTS

7.1 Setting

The main test data are those from TREC 1-3 ad hoc and filtering tracks, including queries 1-150, and documents on Disks 1-3. The choice of this test collection is due to the availability of manually specified domain for each query. This allows us to compare with an approach using automatic domain identification. Below is an example of topic:

<num> Number: 103
<dom> Domain: Law and Government
<title> Topic: Welfare Reform

We only use topic titles in all our tests. Queries 1-50 are used for training and 51-150 for testing. 13 domains are defined in these queries and their distributions among the two sets of queries are shown in Fig. 1. We can see that the distribution varies strongly between domains and between the two query sets.

We have also tested on TREC 7 and 8 data. For this series of tests, each collection is used in turn as training data while the other is used for testing. Some statistics of the data are described in Tab. 2.

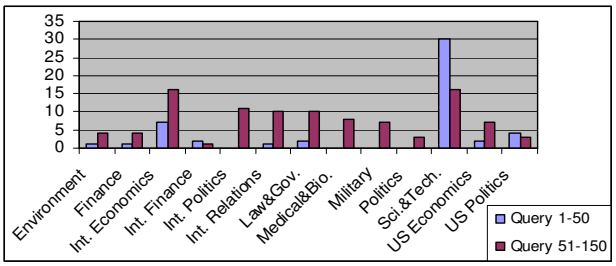

Figure 1. Distribution of domains

Table 2. TREC collection statistics

Collection	Document	Size (GB)	Voc.	# of Doc.	Query
Training	Disk 2	0.86	350,085	231,219	1-50
Disks 1-3	Disks 1-3	3.10	785,932	1,078,166	51-150
TREC7	Disks 4-5	1.85	630,383	528,155	351-400
TREC8	Disks 4-5	1.85	630,383	528,155	401-450

All the documents are preprocessed using Porter stemmer in Lemur and the standard stoplist is used. Some queries (4, 5 and 3 in the three query sets) only contain one word. For these queries, knowledge model is not applicable.

On domain models, we examine several questions:

- When query domain is specified manually, is it useful to incorporate the domain model?
- If the query domain is not specified, can it be determined automatically? How effective is this method?
- We described two ways to gather documents for a domain: either using documents judged relevant to queries in the domain or using documents retrieved for these queries. How do they compare?

On Knowledge model, in addition to testing its effectiveness, we also want to compare the context-dependent relations with context-independent ones.

Finally, we will see the impact of each component model when all the factors are combined.

7.2 Baseline Methods

Two baseline models are used: the classical unigram model without any expansion, and the model with Feedback. In all the experiments, document models are created using Jelinek-Mercer smoothing. This choice is made according to the observation in [36] that the method performs very well for long queries. In our case, as queries are expanded, they perform similarly to long queries. In our preliminary tests, we also found this method performed better than the other methods (e.g. Dirichlet), especially for the main baseline method with Feedback model. Table 3 shows the retrieval effectiveness on all the collections.

7.3 Knowledge Models

This model is combined with both baseline models (with or without feedback). We also compare the context-dependent knowledge model with the traditional context-independent term relations (defined between two single terms), which are used to expand queries. This latter selects expansion terms with strongest global relation to the query. This relation is measured by the sum of relations to each of the query terms. This method is equivalent to [24]. It is also similar to the translation model [3]. We call it

Table 3. Baseline models

Coll.	Measure	Unigram Model	
		Without FB	With FB
Disks 1-3	AvgP	0.1570	0.2344 (+49.30%)
	Recall /48 355	15 711	19 513
	P@10	0.4050	0.5010
TREC7	AvgP	0.1656	0.2176 (+31.40%)
	Recall /4 674	2 237	2 777
	P@10	0.3420	0.3860
TREC8	AvgP	0.2387	0.2909 (+21.87%)
	Recall /4 728	2 764	3 237
	P@10	0.4340	0.4860

Table 4. Knowledge models

Coll.	Measure	Co-occurrence		Knowledge model	
		Without FB	With FB	Without FB	With FB
Disks1-3	AvgP	0.1884 (+20.00%)++	0.2432 (+3.75%)**	0.2164 (+37.83%)++	0.2463 (+5.08%)**
	Recall /48 355	17 430	20 020	18 944	20 260
	P@10	0.4640	0.5160	0.5050	0.5120
TREC7	AvgP	0.1823 (+10.08%)++	0.2350 (+8.00%)*	0.2157 (+30.25%)++	0.2401 (+10.34%)**
	Recall /4 674	2 329	2 933	2 709	2 985
	P@10	0.3780	0.3760	0.3900	0.3900
TREC8	AvgP	0.2519 (+5.53%)	0.2926 (+0.58%)	0.2724 (+14.12%)++	0.3007 (+3.37%)
	Recall /4 728	2 829	3 279	3 090	3 338
	P@10	0.4360	0.4940	0.4720	0.5000

(The column **WithoutFB** is compared to the baseline model without feedback, while **WithFB** is compared to the baseline with feedback. ++ and + mean significant changes in t-test with respect to the baseline without feedback, at the level of $p<0.01$ and $p<0.05$, respectively. ** and * are similar but compared to the baseline model with feedback.)

Co-occurrence model in Table 4. T-test is also performed for statistical significance.

As we can see, simple co-occurrence relations can produce relatively strong improvements; but context-dependent relations can produce much stronger improvements in all cases, especially when feedback is not used. All the improvements over co-occurrence model are statistically significant (this is not shown in the table). The large differences between the two types of relation clearly show that context-dependent relations are more appropriate for query expansion. This confirms the hypothesis we made, that by incorporating context information into relations, we can better determine the appropriate relations to apply and thus avoid introducing inappropriate expansion terms. The following example can further confirm this observation, where we show the strongest expansion terms suggested by both types of relation for the query #384 "*space station moon*":

Co-occurrence Relations: year 0.016552 power 0.013226 time 0.010925 1 0.009422 develop 0.008932 offic 0.008485 oper 0.008408 2 0.007875 earth 0.007843 work 0.007801 radio 0.007701 system 0.007627 build 0.007451 000 0.007403 includ 0.007377 state 0.007076 program 0.007062 nation 0.006937 open 0.006889 servic 0.006809 air 0.006734 space 0.006685 nuclear 0.006521 full 0.006425 make 0.006410 compani 0.006262 peopl 0.006244 project 0.006147 unit 0.006114 gener 0.006036 dai 0.006029

Context-Dependent Relations: space 0.053913 mar 0.046589 earth 0.041786 man 0.037770 program 0.033077 project 0.026901 base 0.025213 orbit 0.025190 build 0.025042 mission 0.023974 call 0.022573 explor 0.021601 launch 0.019574 develop 0.019153 shuttl 0.016966 plan 0.016641 flight 0.016169 station 0.016045 intern 0.016002 energi 0.015556 oper 0.014536 power 0.014224 transport 0.012944 construct 0.012160 nasa 0.011985 nation 0.011855 perman 0.011521 japan 0.011433 apollo 0.010997 lunar 0.010898

In comparison with the baseline model with feedback (Tab. 3), we see that the improvements made by Knowledge model alone are slightly lower. However, when both models are combined, there are additional improvements over the Feedback model, and these improvements are statistically significant in 2 cases out of 3. This demonstrates that the impacts produced by feedback and term relations are different and complementary.

7.4 Domain Models

In this section, we test several strategies to create and use domain models, by exploiting the domain information of the query set in various ways.

Strategies for creating domain models:

C1 - With the **relevant documents** for the in-domain queries: this strategy simulates the case where we have an existing directory in which documents relevant to the domain are included.

C2 - With the **top-100 documents** retrieved with the in-domain queries: this strategy simulates the case where the user specifies a domain for his queries without judging document relevance, and the system gathers related documents from his search history.

Strategies for using domain models:

U1 - The domain model is determined by the **user manually**.

U2 - The domain model is determined by the **system.**

7.4.1 Creating Domain models

We test strategies C1 and C2. In this series of tests, each of the queries 51-150 is used in turn as the test query while the other queries and their relevant documents (C1) or top-ranked retrieved documents (C2) are used to create domain models. The same method is used on queries 1-50 to tune the parameters.

Table 5. Domain models with relevant documents (C1)

Coll.	Measure	Domain		Sub-Domain	
		Without FB	With FB	Without FB	With FB
Disks1-3 (U1)	AvgP	0.1700 (+8.28%)++	0.2454 (+4.69%)**	0.1918 (+22.17%)++	0.2461 (+4.99%)**
	Recall /48 355	16 517	20 141	17 872	20 212
	P@10	0.4370	0.5130	0.4490	0.5150
TREC7 (U2)	AvgP	0.1715 (+3.56%)++	0.2389 (+9.79%)*	0.1842 (+11.23%)++	0.2408 (+10.66%)**
	Recall /4 674	2 270	2 965	2 428	2 987
	P@10	0.3720	0.3740	0.3880	0.3760
TREC8 (U2)	AvgP	0.2442 (+2. 30%)	0.2957 (+1.65%)	0.2563 (+7.37%)	0.2967 (+1.99%)
	Recall /4 728	2 796	3 308	2 873	3 302
	P@10	0.4420	0.5000	0.4280	0.5020

Table 6. Domain models with top-100 documents (C2)

Coll.	Measure	Domain		Sub-Domain	
		Without FB	With FB	Without FB	With FB
Disks1-3 (U1)	AvgP	0.1718 (+9.43%)++	0.2456 (+4.78%)**	0.1799 (+14.59%)++	0.2452 (+4.61%)**
	Recall /48 355	16 558	20 131	17 341	20 155
	P@10	0.4300	0.5140	0.4220	0.5110
TREC7 (U2)	AvgP	0.1765 (+6.58%)++	0.2395 (+10.06%)**	0.1785 (+7.79%)++	0.2393 (+9.97%)**
	Recall /4 674	2 319	2 969	2 254	2 968
	P@10	0.3780	0.3820	0.3820	0.3820
TREC8 (U2)	AvgP	0.2434 (+1.97%)	0.2949 (+1.38%)	0.2441 (+2.26%)	0.2961 (+1.79%)
	Recall /4 728	2 772	3 318	2 734	3 311
	P@10	0.4380	0.4960	0.4280	0.5020

We also compare the domain models created with all the in-domain documents (**Domain**) and with only the top-10 retrieved documents in the domain with the query (**Sub-Domain**). In these tests, we use manual identification of query domain for Disks 1-3 (U1), but automatic identification for TREC7 and 8 (U2).

First, it is interesting to notice that the incorporation of domain models can generally improve retrieval effectiveness in all the cases. The improvements on Disks 1-3 and TREC7 are statistically significant. However, the improvement scales are smaller than using Feedback and Relation models. Looking at the distribution of the domains (Fig. 1), this observation is not surprising: for many domains, we only have few training queries, thus few in-domain documents to create domain models. In addition, topics in the same domain can vary greatly, in particular in large domains such as "science and technology", "international politics", etc.

Second, we observe that the two methods to create domain models perform equally well (Tab. 6 vs. Tab. 5). In other words, providing relevance judgments for queries does not add much advantage for the purpose of creating domain models. This may seem surprising. An analysis immediately shows the reason: a domain model (in the way we created) only captures term distribution in the domain. Relevant documents for all in-domain queries vary greatly. Therefore, in some large domains, characteristic terms have variable effects on queries. On the other hand, as we only use term distribution, even if the top documents retrieved for the in-domain queries are irrelevant, they can still contain domain characteristic terms similarly to relevant documents. Thus both strategies produce very similar effects. This result opens the door for a simpler method that does not require relevance judgments, for example using search history.

Third, without Feedback model, the sub-domain models constructed with relevant documents perform much better than the whole domain models (Tab. 5). However, once Feedback model is used, the advantage disappears. On one hand, this confirms our earlier hypothesis that a domain may be too large to be able to suggest relevant terms for new queries in the domain. It indirectly validates our first hypothesis that a single user model or profile may be too large, so smaller domain models are preferred. On the other hand, sub-domain models capture similar characteristics to Feedback model. So when the latter is used, sub-domain models become superfluous. However, if domain models are constructed with top-ranked documents (Tab. 6), sub-domain models make much less differences. This can be explained by the fact that the domains constructed with top-ranked documents tend to be more uniform than relevant documents with respect to term distribution, as the top retrieved documents usually have stronger statistical correspondence with the queries than the relevant documents.

7.4.2 Determining Query Domain Automatically

It is not realistic to always ask users to specify a domain for their queries. Here, we examine the possibility to automatically identify query domains. Table 7 shows the results with this strategy using both strategies for domain model construction. We can observe that the effectiveness is only slightly lower than those produced with manual identification of query domain (Tab. 5 & 6, Domain models). This shows that automatic domain identification is a way to select domain model as effective as manual identification. This also demonstrates the feasibility to use domain models for queries when no domain information is provided.

Table 7. Automatic query domain identification (U2)

Coll.	Measure	Dom. with rel. doc. (C1)		Dom. with top-100 doc. (C2)	
		Without FB	With FB	Without FB	With FB
Disks 1-3 (U2)	AvgP	0.1650 (+5.10%)++	0.2444 (+4.27%)**	0.1670 (+6.37%)++	0.2449 (+4.48%)**
	Recall	16 343	20 061	16 414	20 090
	P@10	0.4270	0.5100	0.4090	0.5140

Table 8. Complete models (C1)

Coll.	Measure	All Doc. Domain	
		Man. dom. id. (U1)	Auto. dom. id. (U2)
Disks 1-3	AvgP	0.2501 (+6.70%) **	0.2489 (+6.19%) **
	Recall /48 355	20 514	20 367
	P@10	0.5200	0.5230
TREC7	AvgP	N/A	0.2462 (+13.14%) **
	Recall /4 674		3 014
	P@10		0.3960
TREC8	AvgP	N/A	0.3029 (+4.13%) **
	Recall /4 728		3 321
	P@10		0.5020

Looking at the accuracy of the automatic domain identification, however, it is surprisingly low: for queries 51-150, only 38% of the determined domains correspond to the manual identifications. This is much lower than the above 80% rates reported in [18]. A detailed analysis reveals that the main reason is the closeness of several domains in TREC queries (e.g. "International relations", "International politics", "Politics"). However, in this situation, wrong domains assigned to queries are not always irrelevant and useless. For example, even when a query in "International relations" is classified in "International politics", the latter domain can still suggest useful terms to the query. Therefore, the relatively low classification accuracy does not mean low usefulness of the domain models.

7.5 Complete Models

The results with the complete model are shown in Table 8. This model integrates all the components described in this paper: Original query model, Feedback model, Domain model and Knowledge model. We have tested both strategies to create domain models, but the differences between them are very small. So we only report the results with the relevant documents.

Our first observation is that the complete models produce the best results. All the improvements over the baseline model (with feedback) are statistically significant. This result confirms that the integration of contextual factors is effective. Compared to the other results, we see consistent, although small in some cases, improvements over all the partial models.

Looking at the mixture weights, which may reflect the importance of each model, we observed that the best settings in all the collections vary in the following ranges: $0.1 \leq \alpha_0 \leq 0.2$, $0.1 \leq \alpha_{Dom} \leq 0.2$, $0.1 \leq \alpha_K \leq 0.2$ and $0.5 \leq \alpha_F \leq 0.6$. We see that the most important factor is Feedback model. This is also the single factor which produced the highest improvements over the original query model. This observation seems to indicate that this model has the highest capability to capture the information need behind the query. However, even with lower weights, the other models do have strong impacts on the final effectiveness. This demonstrates the benefit of integrating more contextual factors in IR.

8. CONCLUSIONS

Traditional IR approaches usually consider the query as the only element available for the user information need. Many previous studies have investigated the integration of some contextual factors in IR models, typically by incorporating a user profile. In this paper, we argue that a single user profile (or model) can contain a too large variety of different topics so that new queries can be incorrectly biased. Similarly to some previous studies, we propose to model topic domains instead of the user.

Previous investigations on context focused on factors around the query. We showed in this paper that factors within the query are also important – they help select the appropriate term relations to apply in query expansion.

We have integrated the above contextual factors, together with feedback model, in a single language model. Our experimental results strongly confirm the benefit of using contexts in IR. This work also shows that the language modeling framework is appropriate for integrating many contextual factors.

This work can be further improved on several aspects, including other methods to extract term relations, to integrate more context words in conditions and to identify query domains. It would also be interesting to test the method on Web search using user search history. We will investigate these problems in our future research.

9. REFERENCES

[1] Bai, J., Nie, J.Y., Cao, G., Context-dependent term relations for information retrieval, *EMNLP'06*, pp. 551-559, 2006.

[2] Belkin, N.J., Interaction with texts: Information retrieval as information seeking behavior, *Information Retrieval'93: Von der modellierung zu anwendung*, pp. 55-66, Konstanz: Krause & Womser-Hacker, 1993.

[3] Berger, A., Lafferty, J., Information retrieval as statistical translation, *SIGIR'99*, pp. 222-229, 1999.

[4] Bouchard, H., Nie, J.Y., Modèles de langue appliqués à la recherche d'information contextuelle, *Conf. en Recherche d'Information et Applications (CORIA)*, Lyon, 2006.

[5] Chirita, P.A., Paiu, R., Nejdl, W., Kohlschütter, C., Using ODP metadata to personalize search, *SIGIR*, pp. 178-185, 2005.

[6] Church, K. W., Hanks, P., Word association norms, mutual information, and lexicography. *ACL*, pp. 22-29, 1989.

[7] Croft, W. B., Cronen-Townsend, S., Lavrenko, V., Relevance feedback and personalization: A language modeling perspective, In: *The DELOS-NSF Workshop on Personalization and Recommender Systems Digital Libraries*, pp. 49-54, 2006.

[8] Croft, W. B., Wei, X., Context-based topic models for query modification, CIIR Technical Report, University of Massachusetts, 2005.

[9] Dumais, S., Cutrell, E., Cadiz, J., Jancke, G., Sarin, R., Robbins, D. C., Stuff I've seen: a system for personal information retrieval and re-use, *SIGIR'03*, pp. 72-79, 2003.

[10] Fang, H., Zhai, C., Semantic term matching in axiomatic approaches to information retrieval, *SIGIR'06,* pp.115-122, 2006.

[11] Gao, J., Qi, H., Xia, X., Nie, J.-Y., Linear discriminative model for information retrieval. *SIGIR'05*, pp. 290-297, 2005.

[12] Goole Personalized Search, http://www.google.com/psearch.

[13] Hipp, J., Guntzer, U., Nakhaeizadeh, G., Algorithms for association rule mining - a general survey and comparison. *SIGKDD explorations*, 2 (1), pp. 58-64, 2000.

[14] Ingwersen, P., Jäverlin, K., Information retrieval in context: IRiX, *SIGIR Forum*, 39: pp. 31-39, 2004.

[15] Kim, H.-R., Chan, P.K., Personalized ranking of search results with learned user interest hierarchies from bookmarks, *WEBKDD'05 Workshop* at *ACM-KDD*, pp. 32-43, 2005.

[16] Lavrenko, V., Croft, W. B., Relevance-based language models, *SIGIR'01*, pp. 120-127, 2001.

[17] Lau, R., Bruza, P., Song, D., Belief revision for adaptive information retrieval, *SIGIR'04*, pp. 130-137, 2004.

[18] Liu, F., Yu,C., Meng, W., Personalized web search by mapping user queries to categories, *CIKM'02,* pp. 558-565.

[19] Liu, X., Croft, W. B., Cluster-based retrieval using language models, *SIGIR '04*, pp. 186-193, 2004.

[20] Morris, R.C., Toward a user-centered information service, *JASIS*, 45: pp. 20-30, 1994.

[21] Park, T.K., Toward a theory of user-based relevance: A call for a new paradigm of inquiry, *JASIS*, 45: pp. 135-141, 1994.

[22] Peng, F., Schuurmans, D., Wang, S. Augmenting Naive Bayes Classifiers with Statistical Language Models. *Inf. Retr.* 7(3-4): pp. 317-345, 2004.

[23] Pitkow, J., Schütze, H., Cass, T., Cooley, R., Turnbull, D., Edmonds, A., Adar, E., Breuel, T., Personalized Search, *Communications of ACM*, 45: pp. 50-55, 2002.

[24] Qiu, Y., Frei, H.P. Concept based query expansion. *SIGIR'93,* pp.160-169, 1993.

[25] Sanderson, M., Retrieving with good sense, *Inf. Ret.,* 2(1): pp. 49-69, 2000.

[26] Schamber, L., Eisenberg, M.B., Nilan, M.S., A re-examination of relevance: Towards a dynamic, situational definition, *Information Processing and Management*, 26(6): pp. 755-774, 1990.

[27] Schütze, H., Pedersen J.O., A cooccurrence-based thesaurus and two applications to information retrieval, *Information Processing and Management*, 33(3): pp. 307-318, 1997.

[28] Shen, D., Pan, R., Sun, J-T., Pan, J.J., Wu, K., Yin, J., Yang, Q. Query enrichment for web-query classification. *ACM-TOIS*, 24(3): pp. 320-352, 2006.

[29] Shen, X., Tan, B., Zhai, C., Context-sensitive information retrieval using implicit feedback, *SIGIR'05,* pp. 43-50, 2005.

[30] Teevan, J., Dumais, S.T., Horvitz, E., Personalizing search via automated analysis of interests and activities, *SIGIR'05,* pp. 449-456, 2005.

[31] Voorhees, E., Query expansion using lexical-semantic relations. *SIGIR'94*, pp. 61-69, 1994.

[32] Xu, J., Croft, W.B., Query expansion using local and global document analysis, *SIGIR'96,* pp. 4-11, 1996.

[33] Yarowsky, D. Unsupervised word sense disambiguation rivaling supervised methods. *ACL*, pp. 189-196. 1995.

[34] Zhou X., Hu X., Zhang X., Lin X., Song I-Y., Context-sensitive semantic smoothing for the language modeling approach to genomic IR, *SIGIR'06*, pp. 170-177, 2006.

[35] Zhai, C., Lafferty, J., Model-based feedback in the language modeling approach to information retrieval, *CIKM'01,* pp. 403-410, 2001.

[36] Zhai, C., Lafferty, J., A study of smoothing methods for language models applied to ad hoc information retrieval. *SIGIR*, pp.334-342, 2001.