Personalized Access to Contextual Information by using an Assistant for Query Reformulation

Ounas ASFARI
ERIC Laboratory
University of Lyon 2
Bron, France
Ounas.Asfari@univ-lyon2.fr

Bich-Liên Doan, Yolaine Bourda
SUPELEC/Department of Computer Science
Gif-Sur-Yvette, France
Bich-lien.Doan@supelec.fr,
Yolaine.Bourda@supelec.fr

Jean-Paul Sansonnet
LIMSI-CNRS
University of Paris 11
Orsay, France
Jps@limsi.fr

***Abstract*— Access to relevant information adapted to the needs and the context of the user is a real challenge in Web Search, owing to the increases of heterogeneous resources and the varied data on the web. There are always certain needs behind the user query, these queries are often ambiguous and shortened, and thus we need to handle these queries intelligently to satisfy the user's needs. For improving user query processing, we present a context-based hybrid method for query expansion that automatically generates new reformulated queries in order to guide the information retrieval system to provide context-based personalized results depending on the user profile and his/her context. Here, we consider the user context as the actual state of the task that the user is undertaking when the information retrieval process takes place. Thus State Reformulated Queries (SRQ) are generated according to the task states and the user profile which is constructed by considering related concepts from existing concepts in domain ontology. Using a task model, we will show that it is possible to determine the user's current task automatically. We present an experimental study in order to quantify the improvement provided by our system compared to the direct querying of a search engine without reformulation, or compared to the personalized reformulation based on a user profile only. The preliminary results have proved the relevance of our approach in certain contexts.**

Keywords-Information Retrieval; Query Reformulation; Context; Task modeling; Personalization; user profile.

I. INTRODUCTION

The Internet offers almost unlimited access to all kinds of information (text, audiovisual, etc.), there is a vast, growing expanse of data to search, heterogeneous data, and an expanding base of users with many diverse information needs; thus, the Information Retrieval (IR) field has been more critical than ever. Information Retrieval Systems (IRS) aims to retrieve relevant documents in response to a user need, which is usually expressed as a query. The retrieved documents are returned to the user in decreasing order of relevance, which is typically determined by weighting models. As the volume of the heterogeneous resources on the web increases and the data becomes more varied, massive response results are issued to user queries. Thus, large amounts of information are returned in which it is often difficult to distinguish relevant information from secondary information or even noise; this is due to information retrieval systems IRS that generally handle user queries without considering the contexts in which users submit these queries [1]. Therefore it is difficult to obtain desired results from the returned results by IRS. In recent research, IR researchers have begun to expand their efforts to satisfy the information needs that users express in their queries by considering the personalized information retrieval area and by using the context notion in information retrieval.

Recent studies, like [2], have tried to enhance a user query with user's preferences, by creating a dynamic user profile, in order to provide personalized results. However, a user profile may not be sufficient for a variety of user queries. Take as an example a user who enters the query "*Java*" into a personalized Web search engine. Let us now suppose that the user has an interest for computer programming. With this information at hand, it should be possible for a personalized search engine to disambiguate the original query "*Java*". The user should receive results about Java programming language in the top results. But in particular situations, the supposed user may need information about the Java Island, to prepare a trip for example, or information about the Java Coffee that is not specified in his profile. Thus the user will hardly find these results subjectively interesting in a particular situation. One disadvantage of automatic personalization techniques is that they are generally applied out of context. Thus, not all of the user interests are relevant all of the time, usually only a subset is active for a given situation, and the rest cannot be considered as relevant preferences.

To overcome the previous problem and to address some of the limitations of classic personalization systems, studies taking into account the user context are currently undertaken [3]. The user context can be assimilated to all factors that can describe his intentions and perceptions of his surroundings [3]. These factors may cover various aspects: environment (light, services, people, etc.), spatial-temporal (location, time, direction, etc.), personal (physiological, mental, professional, etc.), social (friends, colleagues, etc.), task (goals, information task), technical, etc. Fig. 1 shows these factors and examples for each one [4].

The user context has been applied in many fields, and of course in information retrieval area. Context in IR has been subject to a wide scope of interpretation and application [5]. The problem to be addressed here includes how to represent the context, how to determine it at runtime, and how to use it to influence the activation of user preferences. It is very difficult to take into consideration all the contextual factors in one information retrieval system, so the researchers often define the context as certain factors, such as desktop information [6], physical user location [7], recently visited Web pages [8], session interaction data [9], etc.

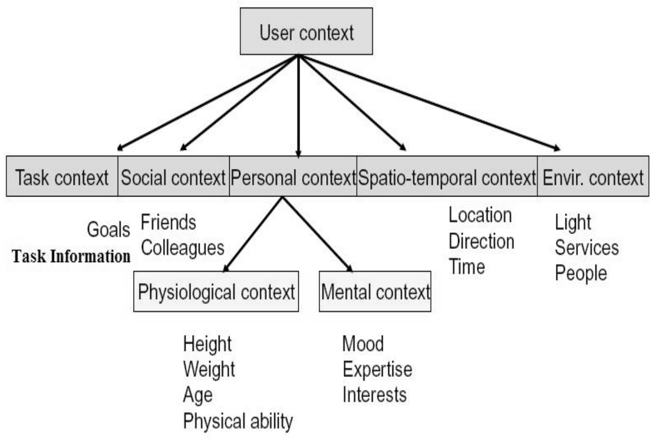


Figure 1. A context model.

In this paper, our definition of the context is that the context describes the user’s current task, its changes over time and its states, i.e., we take into consideration the task which the user is undertaking when the information retrieval process occurs. Consequently, in this paper, when we talk about the context, we talk about the user’s current task and its states over times.

In the present, it has become common to seek daily information on the Web, including such tasks as using information retrieval system for shopping, travel booking, academic research, and so on. Thus, it is important to attempt to determine not only what the user is looking for, but also the task that he is trying to accomplish. Indeed understanding the user task is critical to improve the processing of user needs. On the other hand, the increase of mobile devices (such as PDA, cellular phone, laptop…) including diverse platforms, various work environments, have created new considerations and stakes to be satisfied. So, it is expected to use the mobile devices anywhere to seek information needed to perform the task at hand. This is the case of the mobile user. As we consider the user’s current task, thus we take into account the case of mobile user when he seeks information, needed to perform his current task, by using the mobile devices. Knowing that, the information needs of mobile users to perform tasks are related to contextual factors such as user interests, user current task, location, direction, etc. Here, the problem is that the classic information retrieval systems do not consider the case of mobile users and provide same results to them for different needs, contexts, intentions and personalities, so too many irrelevant results are provided, it is often difficult to distinguish context-relevant information from the irrelevant results.

User query is an element that specifies an information need, but the majorities of these queries are short (85% of users search with no more than 3 keywords [10]) and ambiguous, and often fail to represent the information need, especially the queries of the mobile user, which do not provide a complete specification of the information need. Many relevant terms can be absent from queries and terms included may be ambiguous, thus queries must be processed intelligently to address more of the user’s intended requirements. Typical solution includes expanding query representation that refers to methods of query reformulation, i.e., any kind of transformation applied to a query to facilitate a more effective retrieval. Thus in the query reformulation process the initial user query is reformulated by adding relevant terms. Many approaches use different techniques to select these relevant terms, the difference between them depend on the source of these terms, which may extract from results of previous research (relevance feedback) or from an external resource (semantic resource, user profile,…etc), or depend on the method which is used to select relevant terms to be added to the initial query.

The research, presented in this paper, combines the advantages of the two areas context and personalization in order to provide context-based personalized results as appropriate answer to the user query submitted in a particular context. In fact, the user query that is submitted to a typical Web search engine, or information retrieval system, is not sufficient to retrieve the desired results, thus an aid to the user to formulate his/her query before submitting it to the information retrieval system will be effective, especially in the case of the mobile user because his/her query is often short and related to a task at hand. In this study we do not consider the information retrieval models that mainly focus on the match between the resource (indexed files) and the user query to provide the relevant results, and do not attempt to understand the user query, but the main idea of this study is to propose an intelligent assistant that can generate new reformulated query before submitting it to the information retrieval system in order to personalize and contextualize the access to information. Thus we tries to improve the user query processing based on the user profile (personalization area) and the user context (context area). We will present an algorithm to generate context-related personalized queries from the initial user query. Thus, this paper presents a hybrid method to reformulate user queries depending on his/her profile, which contains the user’s interests and preferences, together with the user‘s context, which is considered as the actual state of his/her current task. The generated query is denoted: State Reformulated Query SRQ. We will prove that these SRQ queries will guide the IRS to provide context-based personalized results which are more relevant than those provided by using the initial user query and those provided by using the user query with simple personalization, depending only on the user profile, in the same context.

We propose that the user queries, which are submitted during the performance of one task at hand, are related to this task, indeed that are part of it. A task is a work package that may include one or more activities, in other words the activities are required to achieve the task. Thus the user task can be represented by using UML activity diagram in order to detect the transitions between the task states at time changes. The activities, in UML activity diagram, are states of doing something. For instance, if a user has to organize a workshop, there are many states for this task, such as the choice of the workshop topics and the choice of the program committee members, etc. Submitting two equivalent queries in tow different states, the relevant results at each task state will be different, so the proposed system has to provide the different relevant results at each state.

The rest of the paper is organized as follows: Section 2 shows the related work; Section 3 introduces models and algorithms to reformulate a user queries and it presents the architecture of our system; Section 4 shows the experimental study and the evaluation of our system; Finally, Section 5 gives the conclusion and future work to be done.

II. Related Work

Many studies have been employed to expand the user query in information retrieval area, as far as we know these studies do not depend on the user task, in this paper, we depend on a task model for expansion the user query, thus in section *A*, we describe related work where the query expansion had been investigated. In section *B*, we review studies where task model had been used.

A. Query Expansion

Query expansion is the process of augmenting the user's query with additional terms in order to improve results by including terms that would lead to retrieving more relevant documents. Many works have been done for providing personalized results by query reformulation. Approaches based on the user profile for query enrichment have been proposed, this process consists in integrating elements of the user profile into the user's query [11]. The limitation of these approaches is that they do not take into consideration the user context to activate elements from the user profile.

Studies on query reformulation by relevance feedback are proposed, the aim is to use the initial query in order to begin the search and then use information about whether or not the initial results are relevant to perform a new query [12]. Because relevance feedback requires the user to select which documents are relevant, it is quite common to use negative feedback. Furthermore the techniques of disambiguation aim to identify precisely the meaning referred by the terms of the query and focus on the documents containing the words quoted in the context defined by the corresponding meaning [13]. But this disambiguation may cause the query to move in a direction away from the user's intention and augment the query with terms related to the wrong interpretation.

Many approaches, like [14], try to reformulate the web queries based on a semantic knowledge by using ontology in order to extract the semantic domain of a word and add the related terms to the initial query, but sometimes these terms are related to the query only under a particular context. Others use sense information (WordNet) to expand the query [15].

In fact, most of the existing query expansion frameworks have an inherent problem of poor coherence between expansion terms and user's search goal. User's search goal, even for the same query, may be different at different states. This often leads to poor retrieval performance. In the logic cases, the user's current search is influenced by his/her current context and in many instances it is influenced by his/her recent searches. In this paper, we propose a hybrid query expansion method that automatically generates query expansion terms from the user profile and the user task. In our approach we exploit both a semantic knowledge (Ontology) and a linguistic knowledge (WordNet) to learn the user's task.

B. Task Model

One aspect of characterizing user's contexts is to consider the tasks which have led them to engage in information retrieval behavior. Users use documents to understand a task and solve a specific problem. Thus, when a user begins a task, he searches the information that will help solve the problem at hand. It must be distinguished between the task of information retrieval and the task that requires the information retrieval in one of its phases. In the second type, it is important to understand the task and its subtasks to detect the related context that will aid the task execution.

Various researchers have demonstrated that the desired search results differ according to types of tasks. According to [16], two types of tasks: Informational task which involves the intent to acquire some information assumed to be present on one or more web pages; transactional task which is based on the intent to perform some web-mediated activity. The approach [17] proves that the nature of the task has an impact on decisions of relevance and usefulness.

The task modeling consists of describing of an optimal procedure to achieve the goal, a sequence of actions or operations in a given environment. Watson's "Just-in-time" information retrieval system [18] monitors user's tasks, anticipates task-based information needs, and proactively provides users with task-relevant information. The effectiveness of such systems depends both on their capability to track user tasks and on their ability to retrieve information that satisfies task-based needs. Here, the user's tasks are monitored by capturing content from Internet Explorer and Microsoft Word applications.

In the approach [19], a language model of a user task is defined as a weighted mixture of task components: queries, result sets, click stream documents, and browsed documents. Approach [5] describes a study on the effect on retrieval performance of using additional information about the user and their search tasks when developing IRF (Implicit Relevance Feedback) algorithms.

In fact, while known to be useful in the development of interactive systems, task models are also known to be difficult to build and to maintain. This difficulty is due to the fact that in order to support a variety of task applications and analyses, task models should include representations of various levels of information, from the highest level user goals down to the lowest level events, and they should be represented in a single, coherent representation scheme.

III. MODELS AND ALGORITHMS

Our aim is to provide context-based personalized results in order to improve the precision of information retrieval systems by reformulating the initial user queries based on the user context and the user profile.

The identification and the description of the user working context when he/she initiates a search can be reduced to the identification of his/her current task and the identification of related terms from his/her profile. This relies on the observation of the on-going user's current task as a contextual factor (for example, user's task like; searching of a restaurant or a hotel, organize trip, etc.). Thus, we design an intelligent assistant to extract relevant terms to the current search session, but what do we mean in relevant terms? Terms are relevant if they are complete and specific:

- Complete: This means that the terms are related to a submitted query, user profile and user's task in the same time. (Query expansion).
- Specific: the terms do not contain stop words, duplicated terms and out of context terms. (Query refinement).

These terms are used to generate a new reformulated query which will submit to the information retrieval system to return context-based results. These terms are not obligatory to be related to the next session of the search at the same user's task.

Here, we will describe our approach which contains three models: Task model, user profile model and SRQ model, which is used to generate the State Reformulated Queries. The task model is responsible for defining the current working context by assigning one task to the initial query from the predefined tasks. The user profile model is responsible for exploiting user profile by using information contained in profile to adapt the retrieved results to this user. The SRQ model is responsible for collecting attributes from the current task, one attribute at least for each task state. The values of these attributes may be retrieved from the operational profile. Thus, to reformulate a user query we do a query expansion with the relevant terms and then we exclude the irrelevant terms (query refinement). The resulted query is denoted SRQ (State reformulated Query).

The several models will be described in the following sections.

A. General Language Model

Before describing the models, in this section, we will construct a new general language model for query expansion including the contextual factors and user profile in order to estimates the parameters in the model that is relevant to information retrieval systems. In the language modeling framework, a typical score function is defined in KL-divergence as follows [20]:

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

where: θ_D is a language model created for a document D. θ_q a language model for the query q, generally estimated by relative frequency of keywords in the query, and V the vocabulary. $P(t|\theta_D)$: The probability of term t in the document model. $P(t|\theta_q)$: The probability of term t in the query model.

$$P(q\mid D)=\prod P(\mathrm{t}\mid\theta_\mathrm{D})^{\mathrm{c}\,(t\,;q)} \qquad (2)$$

where: c (t; q) Frequency of term t in query q;

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. In these cases, we construct the initial query model containing only the original terms, and a new model SRQ (state reformulated queries) containing the added terms. 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^A for the task model created from the main predefined tasks, θ_q^S for the contextual model created from the states of each main task, and θ_q^U for a user profile model. θ_q^0 can be created by MLE (Maximum Likelihood Estimation). Given these models, we create the following final query model by interpolation:

$$P(t\mid\theta_q)=\sum_{i\in X} a_i P(t\mid\theta_q^i) \qquad (3)$$

where: X= {0, A, S, U} is the set of all component models. a_i (With $\sum_{i\in X} a_i = 1$) are their mixture weights. Thus the (1) becomes:

$$Score(q,D)=\sum_{t\in V}\sum_{i\in X} a_i P(t\mid\theta_q^i)\log P(t\mid\theta_D)=\sum_{i\in X} a_i Score_i(q,D) \qquad (4)$$

where the score according to each component model is:

$$Score_i(q,D)=\sum_{t\in V} P(t\mid\theta_q^i)\log P(t\mid\theta_D) \qquad (5)$$

B. User Context Modeling

In this section, we will propose a new contextual analysis method which views the user context as the user's current task and its changes over time. The stages of the task performance are called task states and the transition from one stage to another means that the user has completed this stage of the current task. Thus, in this study, when we talk about the user context we talk about the task which the user is undertaking when the information retrieval process occurs and the states of this task. Therefore, we need to model the user's current task in order to expand the user query with contextual task terms that orientate the search to the relevant results.

1) Current Task Modeling

The task model is used to detect and describe the task which is performed by the user when he submits his/her query to the information retrieval system, as one of the contextual factors which surround the user during the information retrieval process.

Firstly, we have to distinguish between the activity and the task. In fact, an activity can be something you are just doing, and it may or may not have any purpose, it is the action actually performed, while a task is the purpose which is prescribed. Thus the activities are required to achieve the task. In other words, a task is a work package that may include one or more activities. Accordingly we can represent the user's task by a UML activity diagram which contains all the activities needed to perform this task. Each stage which is needed to accomplish the current task is called task state. Thus, the actual activity in the UML activity diagram expresses the actual state of the current task.

In our task model, we depend on study questionnaires [5] which were used to elicit tasks that were expected to be of interest to subjects during the study. In that study [5], subjects were asked to think about their online information seeking activities in terms of tasks, and to create personal labels for each task. They were provided with some example tasks such as "writing a research paper," "travel," and "shopping" but in no other way were they directed, influenced or biased in their choice of tasks. A generic classification was devised for all tasks identified by all subjects, producing the following nine task groupings:
1. Academic Research; 2. News and Weather; 3. Shopping and Selling; 4. Hobbies and Personal Interests; 5. Jobs/Career/Funding; 6. Entertainment; 7. Personal Communication; 8. Teaching; 9. Travel.

For example, the task labels "viewing news", "read the news", and "check the weather" would be classified in Group 2: "News and Weather".

We construct a UML activity diagram for each main task in order to detect the changes over time in the activities needed to accomplish this task and for describing all the sequences of the performed task. Each activity in the generated UML activity diagram expresses the task's actual state. This state can be explained by terms that are called state terms. Thus there is at least one term for each task state.

The task related to a specific query is selected (either manually or automatically) for each query.

- Manually: by the user who assigns one task from the proposed predefined tasks to his/her query. This method is effective when the user can determine exactly his/her current task.
- Automatically: in assigning one task to the user query automatically. For that, we will conceive an algorithm based on the vector space model and using advantages of existing linguistic resources (WordNet) and semantic resources (Ontology). this way can facilitate the process to users, we will explain this algorithm in the following:

At first, we construct an index of terms called *Task Terms Index*. This Task Index consists of:

- Terms of the predefined main tasks. $<t_1, t_2, \ldots, t_i>$. For example: {News, Weather, Shopping, Selling, Teaching…..}.
- State terms $<t_1, t_2, \ldots, t_j>$ for each predefined task: the terms that describe the actual task state. There is at least one term for each task state, for instance, if a user is currently in one activity "Find a Restaurant" to do one task at hand for example travel task, then the state term that explains the activity will be "Restaurant".
- Terms which represent the related-task concepts from ontology such as ODP (Open Directory Project) taxonomy $<t_1, t_2, \ldots, t_k>$.

This index consists of *r* terms. Table 1 shows an example of this task terms index. We will use this index when using the vector space model.

TABLE I. INDEX OF TASK TERMS

Term_Id	Term	tf	Occurrence (postings)
1	News	2	A2:1 A9:1
2	weather	2	A2:1 A9:1
3	Shopping	1	A3:1
4	Restaurant	2	A4:1 A9:1
….	….	….	….
r			

We suppose that each main predefined task can be considered as one document which includes the terms related to this task from the task index. This document can be represented by a terms vector $\vec{A}$. We treat weights as coordinates in the vector space. Term's weight is computed using the term frequency and the inverse document frequency "*tf * idf*" as follows:

$$Wa_{s_i} = tf_{a_{s_i}} * \log(\frac{|A|}{n_{a_{s_i}}})$$

where: A is a set of documents which represent the predefined tasks. Thus |A| is the total number of this set A. According to our proposition |A|=9.

a_{si}: state term that represent the state s_i of the current task A∗.

$n_{a_{si}}$: A number of documents that represent the predefined tasks in which term a_{si} occurs. $tf_{a_{si}}$: is the frequency of term a_{si} in the task A∗∈ A or number of times a term a_{si} occurs in a document that represents a task A∗.

Table 2 shows the weights of few terms in the task terms index. We present the terms related to the task A_2 "news and weather", as an example.

TABLE II. EXAMPLE OF CALCULATING TERM'S WEIGHTS Wa_{si}.

	Counts TFa_{si}						**Weights, Wa_{si}= TFa_{si}* IDFa_{si}**			
Terms	A_1	A_2	..	A_9	n_{asi}	IDFa_{si}	A_1	A_2	..	A_9
News	0	1		1	2	0.653	0	0.653		0.653
Weather	0	1		1	2	0.653	0	0.653		0.653
Tidings	0	1		0	1	0.954	0	0.954		0
Program	0	1		1	2	0.653	0	0.653		0.653
information	1	1		1	3	0.477	0.477	0.477		0.477
temperature	0	1		0	1	0.954	0	0.954		0
atmospheric	0	1		0	1	0.954	0	0.954		0
Meteorological	0	1		0	1	0.954	0	0.954		0
…										
r										

Now, let q <t_1, t_2, ….,t_n> be a query submitted by a specific user, during the performance of one task at hand denoted A_*. This query is composed of n terms; it can be represented as a single term vector $\vec{q}$.

We will use both a linguistic knowledge (*WordNet*) and a semantic knowledge (*ODP Taxonomy*) to parse the user query. Because linguistic knowledge doesn't capture the semantic relationships between terms and semantic knowledge doesn't represent linguistic relationships of the terms. The integration of linguistic and semantic knowledge about the user query into one repository will produce the so-called *query context* which is useful to learn user's task. The notion of query context has been widely mentioned in many studies of information retrieval [21]. The purpose is to use a variety of knowledge involving query to explore the most exact understanding of user's information needs.

Thus the initial query q is parsed using *WordNet* in order to identify the synonymous terms <t_{w1}, t_{w2}, ….,t_{wk}>.

The query and its synonyms q_w are queried against the ODP taxonomy in order to extract a set of concepts <c_1,c_2,…,c_m> (with m≥n) that reflect the semantic knowledge of the user query. These q_w concepts and its sub-concepts produce the query-context $C_{q=}$ <c_1, c_2, …, c_m > which is represented as a single term vector $\vec{C}_q$.

Next, to find out which task vector $\vec{A}$ is closer to the query-context vector $\vec{C}_q$, we resource to the similarity analysis introduced in [22]. The concepts in the query context C_q are compared with the previous predefined nine tasks including their task states terms, for that we use the cosine similarity to compare between the query context vector $\vec{C}_q$ and the vectors which represent the tasks $\vec{A}$ by finding the cosine of the angle between them depending on the task index which is previously explained. As the angle between $\vec{C}_q$ and the predefined nine tasks $\vec{A}$ is shortened, meaning that the two vectors are getting closer, meaning that the similarity weight between them increases. Thus we compute the similarity weights as follows:

$$SW(A_1) = Cos(\vec{C}_q, \vec{A}_1)$$

$$SW(A_2) = Cos(\vec{C}_q, \vec{A}_2)$$

…….

....

....

$$SW(A_9) = Cos(\vec{C}_q, \vec{A}_9)$$

Finally, the task A_* corresponding with the maximum similarity weight $(Max(SW(A_*)))$ is automatically selected as the current task. That means:

$$A_* = \arg\max_{i=1...9}(SW(\vec{C}_q, \vec{A}_i))$$

Thus the task related to a query q <t_1, t_2, ….,t_n> is A_* which is composed of few states S_1, S_2, …, S_i. State terms that represent the states S_1, S_2, …, S_i of the current task A_* are denoted a_{s1}, a_{s2}, ..., a_{si}. Fig. 2 illustrates the comparison between the different vectors which represent the query context $\vec{C}_q$ and the predefined tasks: $\vec{A}_1, \vec{A}_1, \ldots, \vec{A}_9$.

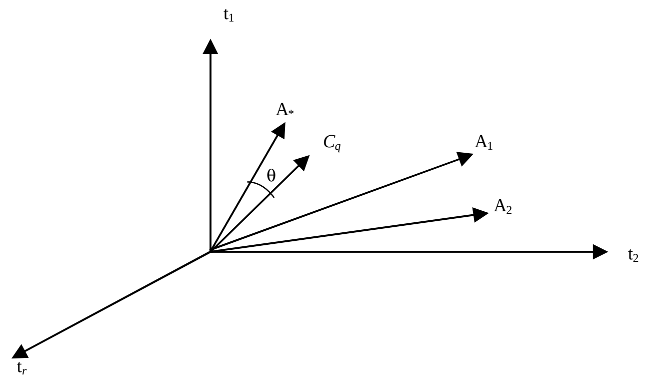


Figure 2. Representation of the tasks and the query as term vectors.

where: t_1, t_2, ….,t_r: terms of task index.

Each term's weight is computed using *tf * idf* as we previously mentioned, (Table 2).

For example, let's take the user query q= {weather}. We take again the table 2 and we determine the term counts TF_i for the query context C_q and their term's weights. That is shown in Table 3.

TABLE III. EXAMPLE OF CALCULATING TERM'S WEIGHTS FOR THE QUERY CONTEXT AND EACH TASK.

		Counts TFa_{si}					**Weights, Wa_{si} = TFa_{si}* IDFa_{si}**			
Terms	C_q	A_1	A_2	A_9	n_{asi}	IDFa_{si}	C_q	A_1	A_2	A_9
News	0	0	1	1	2	0.65	0	0	0.65	0.65
Weather	1	0	1	1	2	0.65	0.65	0	0.65	0.65
Tidings	0	0	1	0	1	0.95	0	0	0.95	0
Program	0	0	1	1	2	0.65	0	0	0.65	0.65
information	0	1	1	1	3	0.48	0	0.48	0.48	0.48
temperature	1	0	1	0	1	0.95	0.95	0	0.95	0
atmospheric	1	0	1	0	1	0.95	0.95	0	0.95	0
Meteorological	1	0	1	0	1	0.95	0.95	0	0.95	0
…										
r										

To find out which task vector is closer to the query vector, we calculate the cosine similarity. First for each task and query-context, we compute all vectors lengths (zero terms ignored). For instance the length vector of the task A_2 is computed as follows:

$$|A_2| = \sqrt{(0.65)^2 + (0.65)^2 + (0.95)^2 + (0.65)^2 + (0.48)^2 + (0.95)^2 + (0.95)^2 + (0.95)^2} = 2.27$$

We do same thing for the others tasks to compute $|A_1|$, $|A_3|$, … , $|A_9|$.

$$|C_q| = \sqrt{(0.65)^2 + (0.95)^2 + (0.95)^2 + (0.95)^2} = 1.78$$

Next, we compute all dot products (zero products ignored). For the task A_2:

$$C_q \bullet A_2 = 0.65*0.65 + 0.95*0.95 + 0.95*0.95 + 0.95*0.95 = 3.157$$

Now we calculate the similarity values:

$$\text{Cosine } \theta_{A_2} = \frac{C_q \bullet A_2}{|C_q| * |A_2|} = \frac{3.157}{1.78 * 2.27} = 0.78$$

Finally, the task corresponding with the maximum similarity value is automatically selected as the current task. In this example the task A_2 has the maximum similarity with the query context C_q.

Let's take an example to extract the query context C_q from the initial user query *q*= {Tourism in Toulouse}. The steps of our algorithm are shown in Table 4:

TABLE IV. APPLYING TASK MODEL TO THE QUERY Q= {TOURISM IN TOULOUSE}.

Description	Knowledge used	Result
Parsing the initial query *q* using WordNet	WordNet	A set of query terms (t_1,.., t_n) (tourism, Toulouse) and its synonym terms (that will be used as the baseline query: (services to tourists, touring, travel, city in France)
The concepts in ontology that represent the baseline query terms are identified, in order to identify the query-context C_q.	Ontological information from ontology (such as, ODP taxonomy).	Set of concepts: query-context (C_q= <C_1, C_2, …,.C_m> with $m \geq n$) relevant to the baseline query: (Travel Guides, Travel and Tourism, Vacations and Touring, Touring Cars, Weather, Food, Maps and Views, hotel, University of Toulouse, Commerce and economy, ….)

Thus, the assigned task to the user query *q* is: A_9= "*Travel*" as it has the maximum similarity weight with the query context C_q.

2) *Contextual Task State*

A task is a work package that may include one or more activities needed to perform this task. A task state is a stage of the task processing, or an efficient way of specifying a particular behavior. Thus the actual state of the current task expresses the actual activity needed to accomplish this task. Each main task consists of several states that can be sequential or parallel, the transition between the task states is related to the events that could occur in the state.

For instance, if we have a task "*shopping*", we can consider the task states for the user u_j as following:

- S_1: Tell you what parts you need.
- S_2: where to find them relative to your location in the store?
- S_3: What is on sale?
- S_4: Do comparative pricing.
- S_5: Use your previous profile information to customize shopping and delivery.

Once the user's task is detected (either manually or automatically), as mentioned in the previous section, it is important to determine the actual state of the current task in order to use the related contextual information in the task modeling. We can consider for each task state at least one term which describes this state and expresses the actual activity, this state term is denoted state attribute a_{si} for the state S_i. For example, if the actual state is "Find a Restaurant", then the state attribute will be "Restaurant". We will see later that related terms from the user profile (such as vegetarian, Italian, etc.) may be assigned to this state attribute.

Accordingly, we can represent the user task including their different states by a UML activity diagram which contains all the activities needed to perform this task. This diagram illustrates the changes in the task-needs over time and describes all the sequences of the performed task. For instance, for the task "*Travel*" (discussed in the previous section) we can design a UML activity diagram for the user u_j that contains all activities as shown in Fig. 3.

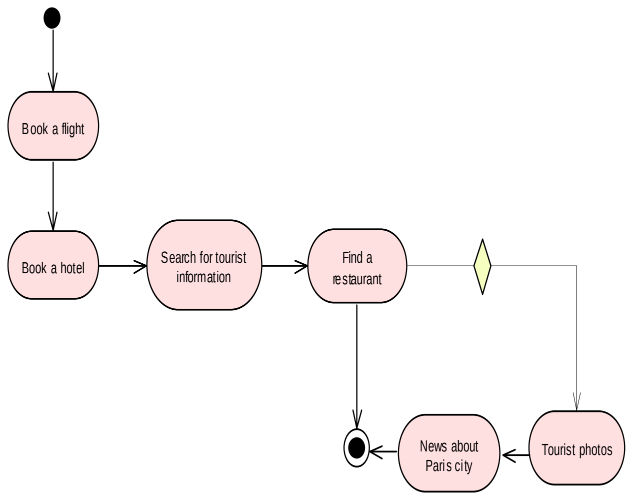


Figure 3. Example of a "travel task" that is modeled by UML activity diagram.

In fact, because a mobile device moves with the user, it is possible to take into account the actual task state in which the user is in when submitting certain queries to the information retrieval system IRS. Such contextual information may come automatically from various sources such as the user's schedule, sensors, entities that interact with the user; it may also be created by the user.

In our approach, according to our assumption that we have 9 main predefined tasks, thus for each user u_j we have one UML activity diagram for each main pre-defined task. After the user's query is submitted to our platform, the related task is assigned automatically to the user query. In this time the system can generate the suitable UML diagram that contains all task states. Set of State Reformulated Queries SRQ related to each state are presented to the user. The user is then asked to choose the appropriate query SRQ according to his state. Finally, from the selected task state, the system will follow the UML activity diagram to present the next query SRQ which is appropriate to the next task state. Thus we need a feedback from the user in order to determine exactly his actual state or his actual activity to perform the main task. This feedback is given by selecting the appropriate query related to the actual state of the user task.

Each query session is defined by the: q_s=<q, u_j, S_i, S_{i-1}>, where S_i: is the actual state of the current task for the user u_j. S_{i-1}: the previous task state. The change from one state to another is done over time when the user u_j complete the actual activity and start the next one. Fig. 4 shows the query session over times.

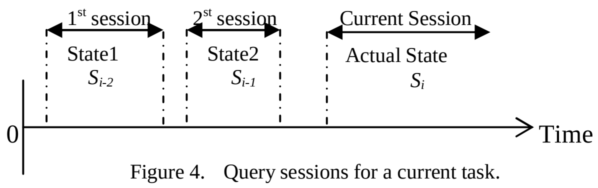


Figure 4. Query sessions for a current task.

In the implementation level, we can conceive that the change from one state to another is done when the user clicks on the "*Next*" button to start the next search session of the query q.

For instance, let's take the example in Fig. 3, if the user u_j is in the activity: "hotel reservation", and if the previous query session was about "book ticket to Toulouse" then the current query session will be about the hotels in Toulouse. At the next query session, if the user u_j submits the same query, thus for this user the query session will be about the "preparation the program to visit Toulouse" which is the next activity in his/her UML diagram shown in Fig. 3.

C. User Profile Modeling

We use ontology as the fundamental source of a semantic knowledge in our framework. Firstly, we have to distinguish between taxonomy and ontology.

1) Ontology and Taxonomy

Ontology is a formal representation of a set of concepts within a domain and the relationships between those concepts. Thus the basic building blocks of ontology are concepts and relationships. Ontology allows the definition of non-taxonomical relation. Concepts (or classes or categories or types) appear as nodes in the ontology graph. Whereas the taxonomy is a subset of ontology, it represents a collection of concepts that are ordered in a hierarchical way. People often refer to taxonomy as a "tree", and Ontology is often more of a "forest". Ontology might encompass a number of taxonomies, with each one organizing a subject in a particular way. Taxonomies tend to be a little casual about what relationship exists between parents and children in the tree. An example of taxonomy is ODP Open Directory Project which is a public collaborative taxonomy of the http://dmoz.org/.

The "DMOZ" Open Directory Project (ODP) represents some of the largest manual metadata collections, most comprehensive human-edited web page catalog currently available. ODP's data structure is organized as a tree, where the categories are internal nodes and pages are leaf nodes. By using symbolic links, nodes can appear to have several parent nodes [23]. A category in the ODP can be considered a concept that is defined by: label of the concept (e.g. 'Microsoft Windows'), Web documents related to the category, parent concepts (e.g. 'Operating Systems', 'Computers') and the children concepts, (e.g. 'Windows XP', 'Windows Vista').

Since ODP truly is free and open, everybody can contribute or re-use the dataset, which is available in RDF (structure and content are available separately), i.e., it can be re-used in other directory services. Google for example uses ODP as basis for its Google Directory service. Fig. 5 shows an example of a tree structure that represents some of topics from ODP for the node "Arts".

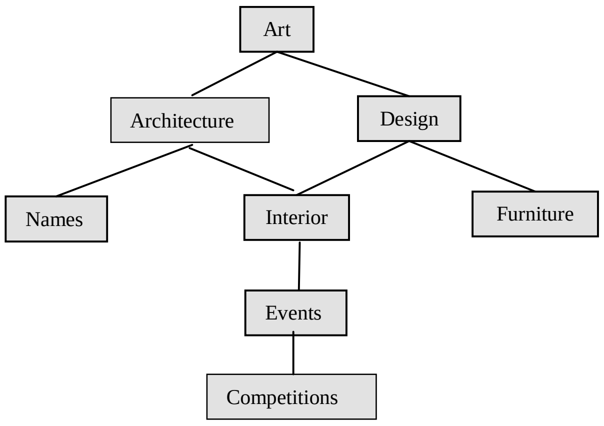


Figure 5. Example for tree structure of topics from ODP.

2) Phases of the User profile Representation

In our system exploiting user profile is carried out through three parts, each with a specific role:

a) The Library Observer

In the library observer phase the user documents, which exist in one library on the user machine, are represented and indexed. Also the library observer is responsible to track the library evolutions.

We assume that the user documents, that are used to construct the user profile, are represented as XML files in order to facilitate the matching between the user documents library and the ODP graph to infer the ontological user profile denoted $Prof_u$. We index these XML files, and consequently we have a XML corpus that will be used to construct the ontological user profile.

For tracking the evolutions of a user profile; when the user interacts with the system by adding new documents or removing others from the user indexed documents, the user profile will be updated based on these updated documents and the annotations for user profile concepts will be modified by spreading activation. Thus, the evolution of the user profile depends on the evolution of the library that supports it; that means when the user adds or removes documents, these modifications are propagated to the ontological profile, and the operational profile will certainly be affected.

b) The Ontological Profile

The ontological profile is a semantic hierarchical structure of the user profile. We use ODP taxonomy as a basis for concepts-based part of our system. As the dataset of ODP is available in RDF, and it is free and open, thus we can reuse it to infer the ontological user profile. Thus, the user profile is represented as a graph of ODP concepts related to the user information (indexed user documents in the library).

In consequence, we consider a dynamic ontological user profile as a semi-structured data in the form of attribute-value pairs where each pair represents a profile's property. The properties are grouped in categories or concepts. For example: global category (language, address, age, etc.) or preference categories (preferences of restaurants, hotel, travel, music, videos, etc.). This allows us to help users to understand relationships between concepts, moreover, to avoid the use of wrong concepts inside queries. e.g., for a query "looking for a job as a Professor", ODP concepts suggests relevant related terms such as teaching, research, etc.

From the ODP concepts, we annotate those related to the user documents. This is done by giving values to these ODP related concepts and weight to each value based on an accumulated similarity with the index of user documents [24], consequently an ontological user profile is created consisting of all concepts with non null value.

Thus, a graph of related concepts of the ODP (Open Directory Project) is inferred using the indexed XML documents, this is shown in Fig.6. Each leaf node in the ontological user profile is a pair, (concept, value), where the annotated value for that concept infer by the comparison with the user documents, this value will be also annotated by a score (*VS*) that reflects the degree of user interest. In Fig. 6, for instance, we consider the node "*Music*" and its children nodes from the ODP taxonomy nodes, we can infer the ontological user profile from these nodes based on the matching with the indexed user documents in the library. Next the concept "*Jazz*" is annotated with the value "*Dixieland*" from the user information because the user has shown interest in Dixieland Jazz, this value is annotated with a score (*VS*) which is "0.08". We can add another value for this concept "*Jazz*" and then score to this value if the user is also interested in another jazz type.

Now we will overview how we can compute the value score *VS*. The score of the concept value (*VS*) is computed using the term frequency and the inverse document frequency (*tf* ∗ *idf*) as follows:

$$VS = \sum_{d \in D} [tf_v * \log(\frac{|D|}{n_v})]$$

where: *D* is the set of user documents used to construct the user profile, |*D*|: is the total number of this set *D*.

n_v: is a number of documents in which value *v* occurs.

tf_v: is the frequency of value *v* in document *d* ϵ *D*, this is computed as follows:

$$tf_{v,d} = \frac{n_{v,d}}{N_d}$$

where $n_{v,d}$ is the number of occurrences of the considered term (value *v*) in document *d*, and the denominator is the sum of number of occurrences of all terms in document *d*, that is, the size of the document | *d* |.

Example:

Let's consider a set of user documents contains 40 documents, and the value "*Dixieland*" appear in 3 documents: *d*7, *d*24, *d*33, (2 times in *d*7, only once time in *d*24, *d*33), the size of documents *d*7, *d*24, *d*33 is 80, 50, and 35, sequentially.

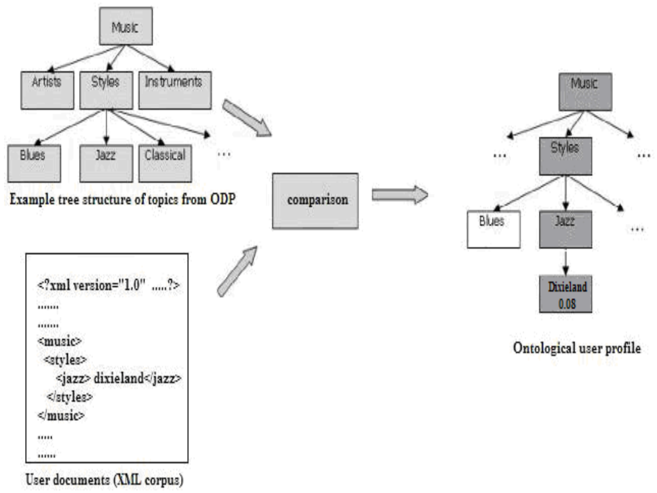
Thus: $tf_{v,7} = 2/80$, $tf_{v,24} = 1/50$, $tf_{v,33} = 1/35$.

We can calculate *VS* by the previous formula:

$$VS = [(0.025 * \log(40/3)) + (0.02 * \log(40/3)) + (0.0286 * \log(40/3))]$$

$$VS = 0{,}0828$$

Thus the value *V* of the leaf node concept in the ontological user profile will be annotated with a score (*VS*) or weight that reflects the degree of user interest for this concept value, in our example the score of the value "*Dixieland*" is *VS*=0.0828 as shown in Fig. 6.



```
<?xml version="1.0" .....?>
.......
.......
<music>
  <styles>
     <jazz> dixieland</jazz>
   </styles>
</music>
.....
......
```

Figure 6. Inferring the ontological profile from user documents and ODP.

Thus, the ontological profile for each user consists of a list of concepts and their current weighted values. For example, a user profile could look like this:

Profile = (<user>, <Concept>, <weighted value>)

E.g.: (Someone, sport, surf 0.8 - ski 0.2 -football 0.9)
(Someone, restaurant, Italian 0.7- French 0.2)
(Someone, cinema, action 0.6- horror 0.4)

In fact using ontology as the basis of the profile allows the user behavior to be matched with existing concepts in the domain ontology and relationship between these concepts. Based on the user's behavior over many interactions, the interest score of the concept values can be incremented or decremented based on contextual evidence. As a result, a graph of related ODP concepts is inferred by using the matching with the user library in order to represent the user profile.

c) The Operational Profile

The operational profile is derived from the ontological profile, as a list of related relevant terms that can be easily used by the other models.

Once the ontological profile is created, the query context-related concepts, from this ontological profile, must be activated in order to extract the operational profile. This is done by mapping the query-context $C_q[i]$ on this ontological user profile (note that, the query context C_q is computed during the construction of the task model). This allows activating for each query-context concept its semantically related concepts from the ontological user profile, following our algorithm, depending on the relevance propagation [25], which will discuss in the next paragraph. Hence, these previous activated user profile concepts with their values will form the operational profile which will be used to reformulate the user query.

Indeed, only an excerpt of the operational profile is used to reformulate the user query, in order to reduce and to focus the activated concepts. The split of the profile in two aspects (ontological/operational) allows a clear separation of concerns between understanding the available user information and taking into account that can be used to lead a search.

3) Algorithm of the Operational Profile Retrieval

As we mentioned previously, the ontological user profile in our approach is represented as an instance of a reference domain ontology in which the concepts are annotated by interest value and scores derived and updated implicitly based on the user's information. In order to extract the operational profile, the query-context $C_q[i]$, which is computed during the construction of the task model, is mapped on the ontological user profile $Prof_u$ to activate for each query-context concept its semantically related concepts by applying our technique that is depended on the relevance propagation [25]. The execution of this technique is depicted in the following Algorithm:

Input: $Prof_u$: Profile for user u, given as a vector of concepts and weighted value.
C_q: Query-Context $C_q= <C_1, C_2, …,.C_i>$ to be answered by the algorithm.
Output: Res_u: Vector of sorted context-related user's concepts.

```
1: Send $C_q$ to a $Prof_u$
2:  For $j$ = 1 to Size ($Prof_u$)
        For $i$ = 1 to Size ($C_q$)
            Calculate: Weight ($C_q[i]$, $Prof_u[j]$)
        End
    End
    For $j$ = 1 to Size ($Prof_u$)
        For $i$ = 1 to Size ($C_q$)
            IF (Weight ($C_q[i]$, $Prof_u[j]$)) $\neq$ 0
            Then: Relevance Propagation
        End
    End
    For $j$ = 1 to Size ($Prof_u$)
        Calculate: Relevance ($Prof_u[j]$, $C_q$)
    End
3: $Res_u$ = Vector of user profile context-related concepts and its Relevance score for the query context $C_q$.
4: Sort $Res_u$ using the Relevance ($Prof_u$, $C_q$) as comparator.
```

We additionally need a function to estimate the weight of the query-context concepts C_q in the user profile concept $Prof_u$: (*Weight* ($C_q[i]$, $Prof_u[j]$)) and the relevance of the user profile concept $Prof_u$ for all query-context concepts C_q (*Relevance* ($Prof_u[j]$, C_q)). Let us inspect this issue in the following:

a) Relevance Propagation Technique

In our user profile modeling approach, we use a new contextual technique to select the context-relevant concepts from the ontological user profile that is represented as semi-structured data like RDF tree. RDF is metadata (data about data) to describe information resources, it is written in XML. As the dataset of ODP is available in RDF, and our ontological user profile is inferred from this RDF graph of ODP as shown in Fig. 6, thus we can imagine the representation of the user profile that is shown in Fig. 7, this graph contains the concepts and the leaf node in this graph is annotated by values and interest scores for this values.

We apply our technique, depending on relevance propagation, on this ontological profile graph to activate for each query-context concept $C_q[i]$ its semantically related concepts from the ontological user profile $Prof_u$. This method consists of computing the node weight, and the node relevance to the query-context concepts. This contextual method consists of three steps:

1. Calculate Weight ($C_q[i]$, $Prof_u[j]$): the weight of the query-context concepts C_q in the user profile concept $Prof_u$.

Each leaf node in the ontological profile is a pair, ($Prof_u[j]$, V($Prof_u[j]$)), where $Prof_u[j]$ is a concept in the reference ontology and V($Prof_u[j]$) is the interest value annotation for that concept. The weight of the query-context concept $C_q[i]$ in the user profile concept node $Prof_u[j]$ is 1, if this node contains the concept $C_q[i]$ and 0 otherwise.

$$Weight\ (\mathrm{C}_q[i], \mathrm{Prof}_u[j]) = \begin{cases} 1 & \text{If } C_q[i] \text{ is in } Prof_u[j] \\ 0 & \text{Otherwise} \end{cases}$$

2. Next we calculate the weight of query-context concept $C_q[i]$ in the ancestor nodes by the relevance propagating from this node to the ancestor node:

$$\mathrm{P}ropagation_i(\mathrm{Prof}_u[j],\ \mathrm{Prof}_u[n]) = Weight(\mathrm{C}_q[i], \mathrm{Prof}_u[j]) * \frac{1}{Max(Dist(\mathrm{Prof}_U[j],\ \mathrm{Prof}_u[n]) + 1)}$$

where: $Prof_u[j]$: user profile concept at j. $Prof_u[n]$: user profile concept at n which is one of the ancestor nodes of the node j (concept j).

$Dist(\mathrm{Prof}_u[j], \mathrm{Prof}_u[n])$: Semantic distance between the two user profile nodes.

3. Aggregation:

Once all the weights of query-context concepts C_q are calculated for all user profile nodes (contain the ancestors nodes), we have to calculate the relevance score of each user profile node for all concepts of context query $C_q= <C_1, C_2, …,.C_i>$ denoted N. This can be estimated in two methods, either "And method" or "OR method".

And method:

Here, the weight aggregation of nodes uses the following formula:

$$N = \mathrm{Re}\,levance\ (\mathrm{Prof}_u[n]\ , \mathrm{C}_q[i]) = \prod_{x_i \in \mathrm{C}_q[\mathrm{i}]} [Weight\ (\mathrm{Prof}_u[n], x_i)]_i$$

Thus, depending on the previous formula, the relevance score *N* is not null for only the nodes which contain all the query-context concepts directly or in their ancestor nodes. Thus this will give the smallest relevant sub tree contains the previous concepts C_q=<C_1, C_2,…, C_i>.

We use the formula *And*, only when we need user profile fragments that contain all the query concepts, and neglect those contain some of query concepts. This case is not appropriate to our system, so we will use the *OR* method for computing the relevance score of user profile nodes for the query-context concepts.

OR method:

The weight aggregation of nodes uses the following formula:

$$N^* = \mathrm{Re}\,levance\,(\mathrm{Prof}_u\,[n]\;, \mathrm{C}_q[i]) = \sum_{x_i \in \mathrm{C}_q[i]} [Weight\;(\mathrm{Prof}_u\,[n], x_i)]_i$$

The relevance score *N* is not null if the node contains one of the query-context concepts directly or in their ancestor nodes. So this will give fragments of user profile that are sorted by decreasing order of *N*.

Example:

Let's consider the initial query q, and the query-context C_q which is composed of three concepts: C_q= {C_1, C_2, C_3}.

We consider also the user profile *u*, which is composed of many concepts represented as RDF graph (metadata); Fig. 7 shows the user profile graph *u*.

The leaf nodes: n_3, n_6, n_9, n_{10}, n_{12} annotate by values, and interest score to these values. Now we calculate the relevance of the user profile nodes for the query-context C_q using the formulas of weight and propagation. For example we calculate the relevance score fore the node n_4:

$$Weight\,(c_1, n_8) = 1 \qquad Weight\,(c_2, n_5) = 1 \qquad Weight\,(c_3, n_7) = 1$$

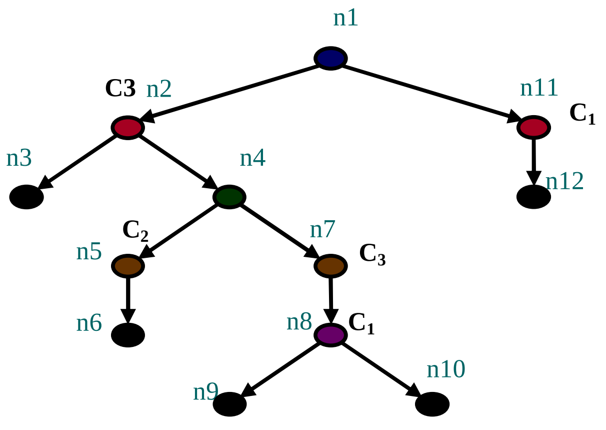


Figure 7. Example of a user profile graph Profu.

Then we follow the algorithm to compute the relevance score of the node n_4 for the concepts C_1, C_2, C_3. We have to propagate the weight not null to n_4:

$$\mathrm{Pr}\;opagation_{\;C_3}(n_7, n_4) = \frac{Weight\;(n_7, C_3)}{Max\;(Dist\;(n_7, n_4) + 1)} = \frac{1}{2}$$

$$\mathrm{Pr}\;opagation_{\;C_2}(n_5, n_4) = \frac{Weight\;(n_5, C_2)}{Max\;(Dist\;(n_5, n_4) + 1)} = \frac{1}{2}$$

$$\mathrm{Pr}\;opagation_{\;C_1}(n_8, n_4) = \frac{Weight\;(n_8, C_1)}{Max\;(Dist\;(n_8, n_4) + 1)} = \frac{1}{3}$$

And:

$$\mathrm{Re}\,levance\;(n_4, \mathrm{C}_q) = \prod_{i=1,2,3} Weight\;(n_4, C_q^{\;i}) = \frac{1}{3} * \frac{1}{2} * \frac{1}{2} = \frac{1}{12}$$

OR:

$$\mathrm{Re}\,levance\,(n_4, \mathrm{C}_q) = \sum_{i=1}^{3} Weight\,(n_4, C_q^{\;i}) = \frac{1}{3} + \frac{1}{2} + \frac{1}{2} = \frac{4}{3}$$

We do the same steps to compute the relevance score of the other user profile nodes, the results are shown in Table 5 for the "And method" and the "Or method".

If we consider the "*And*" method then the smallest relevant sub tree that contains all query concepts is the sub-tree that is presented by the node n_4 and its descending nodes to leafs, because the node n_4 has the most relevance score as shown in Table 5.

But if we consider the "*OR*" method then the node n_7 has the most relevance score, as shown in Table 5 below. In this case the most relevant result is the sub-tree which is presented by the node n_7 and its descending nodes until the leaf nodes.

As we mentioned previously, the leaf nodes may be annotated by many values, and each one annotates with score *VS*, so we select the value that has the greater score *VS*. As a result the concepts of the user profile related to the query-context concepts are: n_7, n_8, n_9, n_{10} and the values of n_9, n_{10} which have greater score *VS*.

These concepts and their values constitute the operational profile; we will depend on this operational profile to generate the reformulated queries SRQ, based on the user profile and his/her context, those queries can be easily used in the search process to get relevant results which are needed to accomplish the task at hand.

TABLE V. RELEVANCE SCORE OF USER PROFILE CONCEPTS *PROF*$_U$ USING BOTH "AND METHOD" N_N, "OR METHOD" N^*_N RESPECTIVELY.

Node	C1	C2	C3	N_n
n1	0.2	0.25	0.25	0.0125
n2	0.25	0.333	0.333	0.0277
n3	0	0	0	0
n4	0.333	0.5	0.5	0.0833
n5	0	1	0	0
n6	0	0	0	0
n7	0.5	0	1	0
n8	1	0	0	0
n9	0	0	0	0
n10	0	0	0	0
n11	1	0	0	0
n12	0	0	0	0

Node	C1	C2	C3	N^*_n
n1	0.2	0.25	0.25	0.7
n2	0.25	0.333	0.333	0.916
n3	0	0	0	0
n4	0.333	0.5	0.5	1. 333
n5	0	1	0	1
n6	0	0	0	0
n7	0.5	0	1	1.5
n8	1	0	0	1
n9	0	0	0	0
n10	0	0	0	0
n11	1	0	0	1
n12	0	0	0	0

D. *SRQ Model (State Reformulated Queries)*

Short queries usually lack sufficient words to capture relevant documents and thus negatively affect the retrieval performance, and thus fail to represent the information need. Query expansion is a technique where original query is supplemented with additional related terms. Existing query expansion frameworks have the problem of poor coherence between expansion terms and user's search goal, For instance, if the query *jaguar* be expanded as the terms {*auto, car, model, cat, jungle*,...} and user is looking for documents related to car, then the expansion terms such as cat and jungle are not relevant to user's search goal.

1) *SRQ Definition*

In the following, we will introduce a new notion State Reformulated Queries (SRQ) which are provided by the reformulation of the initial user queries *q*, related to the current task, depending on the actual state of this task and the user profile. The states of the current task are expressed by activities which are required to accomplish this task and grouped in UML activity diagram including the relations between them, each state represents one search session. The change from one state to another is done over time when the user u_j complete the actual activity and start the next one. Thus for two different task states, submitting the same query the relevant results will not be the same.

Let q= {t_1, t_2..., t_n} be an initial query which is related to the task at hand. The state reformulated query at the task state S_i and for a specific user profile P_j is: $S_i RQ<Q,P_j,S_i>$, this query contains the initial query *q* and the expansion terms $E^{(q)}=\{t_{q,1}, t_{q,2}, t_{q,3}, ...\}$. Thus we have to get the expansion terms $E^{(q)} = \{t_{q,1}, t_{q,2}, t_{q,3},...\}$ which are relevant to user's search goal by exploiting user's implicit feedback at the time of search. The relevant results D_i at the states S_i are produced by applying $S_i RQ<Q,P_j,S_i>$ on an information retrieval system. We expect that the results D_i at the task state S_i are more relevant than those produced by using the initial query *q* at the same state S_i.

A search is handled as follows: the user expresses his/her query, our assistant identifies the context of this search, and it creates the context description and proposes relevant terms to be added to the initial query. The initial user query will be reformulated depending on these relevant terms in order to generate SRQ (State Reformulated Query) to improve the retrieval performance. The assistant then submits the new reformulated query SRQ to a search engine on the Web and gets the results. The documents are then presented to the user in the order of decreasing estimated relevance.

2) *Query Reformulation Phases*

The two phases to generate the State Reformulated Queries (SRQ) are: query expansion and query refinement.

a) *Query expansion*

The initial query is expanded with two types of generated terms which are denoted expansion terms $E^{(q)} = \{t_{q,1}, t_{q,2}, t_{q,3},..\}$:

- Terms which represent the actual state of the current task A_* (a_{s1}, a_{s2}, …,a_{si}). There is at least one term for each task state which describes this state, this state term is denoted state attribute a_{si}. These attributes are computed using the Task model which was previously explained.
- Terms which represent the query-relevant concepts from the ontological user profile with its values (operational profile). The algorithm of extracting these terms from the ontological user profile was previously explained. These terms are denoted user profile attributes (a_{u1}, a_{u2}, …, a_{uj}).

b) *Query Refinement*

After the user query is expanded by new terms, the tool of query refinement must be applied in order to consider only the terms that are related to the actual task context, and disregard those are out of focus for the given context. Thus Query refinement is the incremental process of transforming an initial query into a new reformulated query SRQ that reflects the user's information need in more accurate way.

Sometimes irrelevant attributes may be presented in the retrieved user profile concepts, and thus irrelevant terms are recommended by the operational profile, in order to keep only the relevant user profile attributes for the current task state S_i, we compare between these generated attributes and the actual state attributes, next we consider the attribute of the previous task state, and then we exclude from the generated user profile attributes those non similar with the state attributes. Also we have to exclude the duplicated terms if they exist in the resulting SRQ.

Another method for filtering the previous terms is by asking the user to choose the relevant terms before adding them to the final reformulated query.

Finally, state reformulated queries SRQ are built according to the syntax required by the used search engine in order to submit the queries SRQ and to retrieve relevant results to the user at the actual state of the current task. Boolean operators can be used to construct the final query and adequate care is taken to ensure that the final query meets the syntax requirements, after each step, the user is asked if the query reflects his intension. If so, the final query is constructed using the appropriate syntax and submitted to the search engine.

For the Boolean operator, we use "*And*" with the terms that are extracted from the actual state of the current task, and "*Or*" with the terms that are extracted from the operational profile, because the task state terms are always required while the operational profile terms can be sometimes abandoned. For example, we can imagine the state reformulated query as follows:

SRQ: *q* AND *hotel* OR *2 stars* OR *single*

where:

- *q* is the initial user query.
- "*hotel*": the state term that represents the task actual state (state attribute).
- "*2 stars*" and "*single*" are the relevant terms from the operational profile.

E. *System Architecture*

Fig. 8 illustrates the system architecture. It combines the three models which are described in the previous sections: The task model, the user profile model and the SRQ model.

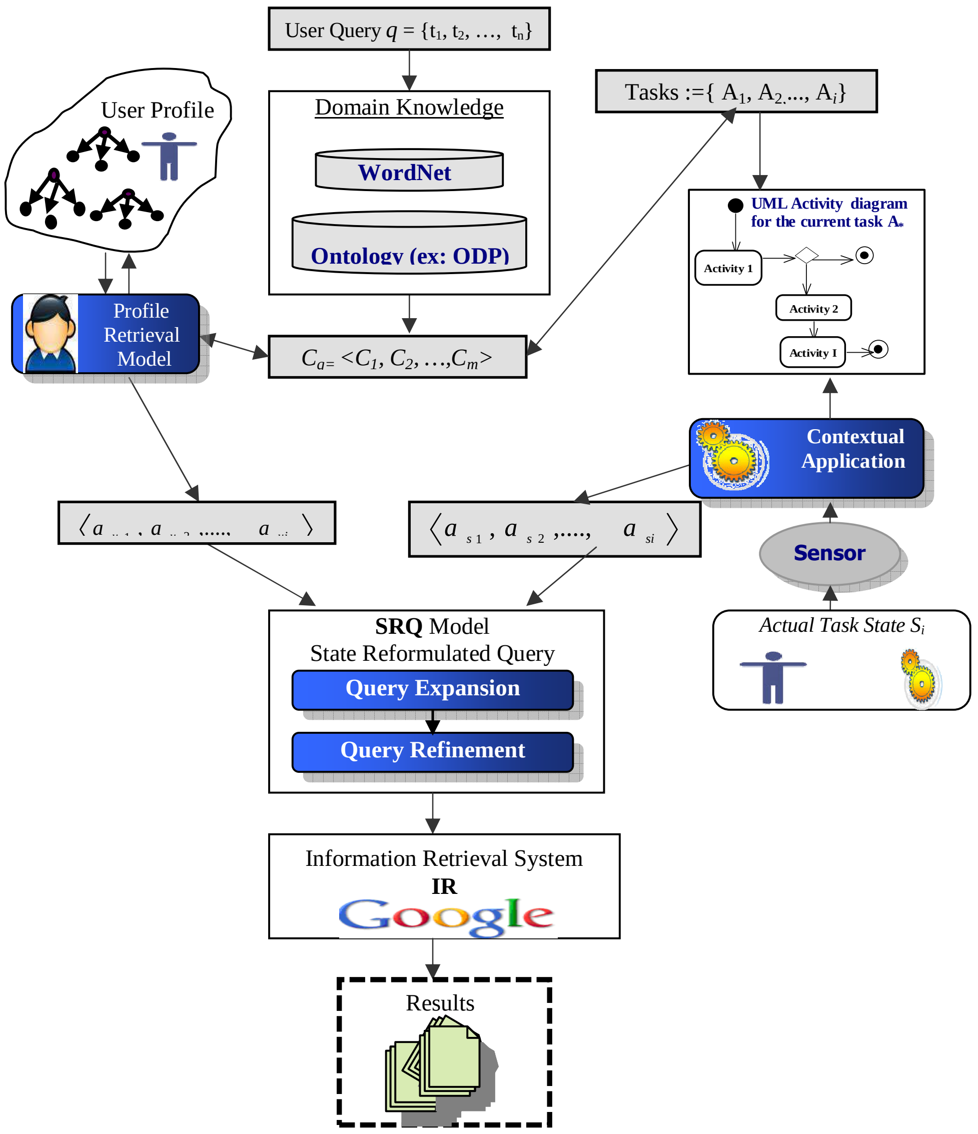


Figure 8. System Architecture.

IV. Evaluation

The evaluation of the personalized information retrieval in context systems is known to be a difficult and expensive [26] due to the dynamic aspect of the system environment and its strongly adaptive properties. A formal evaluation of the contextualization techniques requires a significant amount of extra feedback from users in order to measure how much better a retrieval system can perform with the proposed techniques than without them. Our proposed approach which was described in this paper have been implemented in an experimental prototype, and tested by real users. Evaluation in the context of an evolving real-world system is always a challenge. In order to evaluate and to quantify the improvement provided by our system compared to the direct querying of a search engine without reformulation, or more generally to the use of other assistants, we should verify that using a user context improves the search results, by focusing the system on the most relevant part of the profile. The standard evaluation measures from the Information Retrieval field require the comparison between the performances of retrieval:

- Using the initial user query without any personalization and contextualization.
- Using the user query with simple personalization, depending only on the user profile, (i.e., regardless of the user context, more precisely regardless of his/her task at hand).
- Using the state reformulated queries SRQ which are generated depending on the user context and his/her

profile, (i.e., constrained to the context of his/her current task).

Currently, to compare different configurations (corresponding to different profile, context, query); several agents are used simultaneously by the assistant when handling user query. Thus our experiments have been done with three agents: the « default » agent simply linked to Google, and a « personalized » agent which uses the user profile to rank the results without taking the context into account. A third agent « personalization with context» is also used.

A. *Experimental Study*

In order to evaluate the use of the task context together with the user profile to contextualize returned results, a prototype around the search engine, Google for example, is built using the Google API. This program builds a log of the initial user queries, the returned results by Google, the result on which the user clicked, and the summaries, titles and ranks of the returned results from Google. This log information is used to compute the evaluation metrics at the experimental queries and to evaluate the performance of our system. To conduct the experiments and calculate the evaluation metrics, 10 users are asked to use our system to perform similar tasks by submitting initial queries. The 10 users are classified in three groups, novice, medium and expert, depending on their experience levels in computer science and search engine. Each one is asked to submit queries on 3 different scenarios, where we put the users in specific scenarios to make them thinking about writing appropriate queries for these scenarios. We depend on scenarios such as travel, shopping, restaurant searching, etc. we will illustrate an example of these scenarios in the next section. Consequently a total of 30 queries are selected as experimental queries. The prototype records results on which the users clicked, which we use as a form of implicit user relevance in our analysis.

After the data is collected, we remove from the experimental queries that were no contextual information available for that particular query, and thus we had a log of 30 queries averaging 3 queries per user. We will calculate, at each experimental query, the evaluation metrics in the three cases: using classic search engine Google, using only personalized search without user context, and using our system based on user context and his/her profile.

1) *Example of the experimental scenarios*

Here we will take an example of the scenarios that are used in the experimental study. We consider the task "*Travel*" which was discussed in the section *task modeling* (section 3). We have illustrated in Fig. 3, a UML activity diagram for the user u_j that contains all activities needed to perform this task. Now when the user submits his initial query *q*, which is related to the current task, in our platform, let it be *q*: "*Trip to Paris*", the task model will assign the task "*Travel*" to this query as the first step. Next, the UML activity diagram for this task which is shown in Fig. 3 is retrieved. The system then uses the attributes associated with each task state and the user profile attributes for producing the relevant terms (query expansion phase), next the irrelevant terms are excluded (query refinement phase), finally, the system generate the appropriate state reformulated query SRQ for each task state:

S_1: Book a flight $\Rightarrow S_1$RQ :{ *trip Paris* + "*Flight*" OR *Ticket* + OR *Inexpensive*}.

S_2: Book a hotel $\Rightarrow S_2$RQ :{ *trip Paris* + *"hotel"* +*2 star* OR *single*}.

S_3: Search for tourist information $\Rightarrow S_3$RQ :{ *trip Paris* + "*Monuments*" OR *Weather* OR *plan* OR *Metro*}.

S_4: Find a restaurant $\Rightarrow S_4$RQ :{ *trip Paris* + "*restaurant*" + *Italian* OR *Vegetarian*}.

S_5: Tourist photos $\Rightarrow S_5$RQ :{ *trip Paris* + "*Photos*"}.

S_6: News about Paris city $\Rightarrow S_6$RQ: {*trip Paris*+ "*News"* OR *Weather*}.

where:

"*Flight*", *"hotel"*, "*Monuments*", "*restaurant*", "*Photos*" and "*News"* are the terms that represent task state attributes.

"Ticket", *"Inexpensive"*, *"2 star"*, *"single"*, *"Weather"*, "*plan"*, *"Metro"*, *"Vegetarian"* and *"Italian"* are the relevant terms from the user operational profile.

To evaluate our proposed framework we have to compute the evaluation metrics based on the experimental scenarios.

B. *Evaluation Metrics*

There are many evaluation metrics in the literature for the classic information retrieval evaluation, these metrics often depend on relevance judgments for the returned results, one of the most known of them is the "Precision and Recall" (PR), this metric takes into account the rate of relevant retrieved documents (precision) and the quantity of relevant retrieved documents (recall). Another metric is the Precision at *n* (P@N) [27], P@N is the ration between the number of relevant documents in the first *n* retrieved documents and *n*. The P@N value is more focused on the quality of the top results, with a lower consideration on the quality of the recall of the system. These evaluation metrics for classic IR can be also applied by IIR (Interactive Information Retrieval) authors [9], but IIR system authors must incorporate human subjective judgments, either implicitly (analyzing interaction logs) or explicitly (asking the users to rate the results to provide a best order).

The classic IR evaluation metrics are not sufficient to evaluate our system due to the contextual aspect of the system and the need to provision a real user judgement. Thus to evaluate our proposed framework, the used metrics must cover on one hand the evaluation of the proposed expansion terms which are used to reformulate the initial user query, and on the other hand they must cover the evaluation of returned results. Thus we will use three metrics:

- Quality: measures the quality of expansion terms.
- Precision@k: measures the retrieval effectiveness.
- Dynamics: measures the capability of adapting to the changing needs of users and the changing states of his/her task at hand.

Now we will compute these three evaluation metrics: quality, precision@k, and dynamics, based on the experimental scenarios.

1) *Quality*

Let *q* be an initial user query, given an IR system, $D_c^{(q)}$ is the set of documents actually visited by the user for *q*. Thus $D_c^{(q)}$ represents the relevant results which are evaluated by the user at his/her actual context and taking into account his/her profile using *q*. Therefore, the ideal information retrieval system should retrieve these documents $D_c^{(q)}$ in the foreground and present them to the user at the specific context.

Given a query expansion system, let $E^{(q)}$ be the set of expansion terms for the query *q*, i.e.:

$$E^{(q)} = \{\tau_{q,1}, \tau_{q,2}, \tau_{q,3}, ...\}$$

Then the quality of the expansion terms is defined as follows:

$$Quality = \frac{\left|\rho\,(E^{(q)}, D_c^{(q)})\right|}{\left|E^{(q)}\right|}$$

where:

$\rho\,(E^{(q)}, D_c^{(q)})$: The matching terms between $E^{(q)}$ and $D_c^{(q)}$, that's mean:

$$\rho\,(E^{(q)}, D_c^{(q)}) = \{\tau \mid \tau \in E^{(q)}, \exists d \in D_c^{(q)} \text{ s.t. } \tau \in d\}$$

For example, if we take the scenario presented in the previous section and the user query *q=''trip Paris''*, during this scenario, we take the second state S_2 which is searching a hotel in Paris, at this actual task state we execute the query *q* by using Google and we present the returned results to the user, then the user visits the relevant documents at S_2. If the user visits 5 documents then $\left|D_c^{(q)}\right| = 5$. At this actual state S_2, our system proposes set of expansion terms $E^{(q)}$, this set contains 5 terms which are: trip, Paris, hotel, 2 star, single. Thus: $E^{(q)} = 5$. From these 5 terms, if there are 3 terms existing in the 5 visited documents $D_c^{(q)}$ at S_2, then:

$$\rho(E^{(q)}, D_c^{(q)}) = 3$$

Thus the quality of the expansion terms over this query *q* is:

$$Quality = \frac{\left|\rho\,(E^{(q)}, D_c^{(q)})\right|}{\left|E^{(q)}\right|} = 0.6$$

We do the same steps for the other queries at the different states of this task and then we can compute the average quality of the expansion terms over 10 queries submitted by 10 different users. In consequence, the average quality of the expansion terms by our system is 0.73 for this scenario. Finally we can compute the average quality of the expansion terms over all experimental queries (30 queries) at the different scenarios.

If we depend only on the user profile to generate the expansion terms $E^{(q)}$ for the same user's queries at the same context and the same conditions, thus the $E^{(q)}$ will be different from the first case. In the same steps we calculate the average quality of expansion terms $E^{(q)}$ which are extracted from the user profile and do not taking into account the user context at the same user's queries for the previous scenario (*trip Paris*). In consequence the average quality is 0.34 in this case.

We notice that the average quality of the generated expansion terms, depending on user profile and user context (first case), is higher than that generated depending only on the user profile. Thus our system has an improvement of about 39% in the average quality of the generated expansion terms compared with that of standard personalized systems.

2) *Precision@k*

The second metric is the *Precision@k*, Let $D_n^{(q)}$ be the set of top *n* documents retrieved by the IR system using the query *q*. To define retrieval effectiveness, we determine the number of documents in $D_n^{(q)}$ which are closely related to the documents in $D_c^{(q)}$. We use cosine similarity (previously explained) to define the closeness between two documents. Let $D_r^{(q)}$ be a set of documents from $D_n^{(q)}$ for which the cosine similarity with at least one of the document in $D_c^{(q)}$ is above a threshold Θ_{sim}, that's mean:

$$D_r^{(q)} = \{d_i \mid d_i \in D_n^{(q)}, \exists d_j \in D_c^{(q)} \text{ s.t. } Sim(d_i, d_j) \geq \Theta_{sim}\}$$

Thus, to measure the retrieval effectiveness, we define the *Precision@k* as follows:

$$precision@k = \frac{\left|D_r^{(q)}\right|}{k}$$

To facilitate the experiments, let's *n*=20, then $D_{20}^{(SRQ)}$ represents the first 20 documents from the retrieved results by the IR system (Google for example) by using the state reformulated query SRQ which contains the expansion terms $E^{(q)}$. In the previous section, we mentioned that $D_c^{(q)}$ represents the relevant results for the initial user query *q*, these $D_c^{(q)}$ are evaluated by the user at his/her actual context and taking into account his/her profile. In order to define the closeness between $D_{20}^{(SRQ)}$ and $D_c^{(q)}$ we compute the cosine similarity between the documents of the two sets. We determine the number of documents from $D_{20}^{(SRQ)}$ which are closely related to the documents in $D_c^{(q)}$. Let $D_r^{(SRQ)}$ be a set of documents from $D_{20}^{(SRQ)}$ for which the cosine similarity with at least one of the document in $D_c^{(q)}$ is above a threshold Θ_{sim}. In this study we define $D_r^{(SRQ)}$ with the threshold value [Θ_{sim} = 0.5], because as we know the value of cosine similarity is in the range of [0, 1], we consider the middle point as the threshold value, thus:

$$D_r^{(SRQ)} = \{d_i \mid d_i \in D_{20}^{(SRQ)}, \exists d_j \in D_c^{(q)} \text{ s.t. } Sim(d_i, d_j) \geq 0.5\}$$

Thus: $$precision@K = \frac{\left|D_r^{(SRQ)}\right|}{K}$$

Note that, the set of relevant documents $D_c^{(q)}$ is obtained from the query log or from the user exploring at the snippets of the returned results whereas the set $D_{20}^{(SRQ)}$ is obtained from our experimental retrieval system after simulating the query sequence and submitting the reformulated queries.

Now we compute the retrieval performance (*precision@k*) of our proposed query reformulation system based on user profile and his/her context for all experimental queries of the experimental scenarios. We give the values 5, 10, 20 to k, in order to compute the *Precision@*5, *Precision@*10 and *Precision@*20.

We consider again the scenario "*travel*" in the previous section and the query *q*=''*trip Paris*". We take, as an example, the second state S$_2$ which is searching a hotel in Paris, at this task state the $\left|D_c^{(q)}\right| = 5$ and the S$_2$RQ is: {*trip Paris* + *"hotel"* + *2 star* OR *single*}. We execute this S$_2$RQ by using Google and then we compute $D_r^{(S_2RQ)}$ in the three cases (k=5, k=10, k=20) by calculating the cosine similarity between $D_c^{(q)}$ and $D_5^{(S_2RQ)}$ for k=5, $D_{10}^{(S_2RQ)}$ for k=10 and $D_{20}^{(S_2RQ)}$ for k=20. Thus:

$$precision@\ 5 = \frac{\left|D_r^{(S_2RQ)}\right|}{5} = \frac{3}{5} = 0.6$$

where:

$D_r^{(S_2RQ)} = \left\{ d_i \middle| d_i \in D_5^{(S_2RQ)}, \exists d_j \in D_c^{(q)} \text{ s.t. } Sim(d_i, d_j) \geq 0.5 \right\}$

$D_5^{(S_2RQ)}$ is the set of top 5 documents retrieved by IR system using S$_2$RQ.

For K=10: $$precision@10 = \frac{\left|D_r^{(S_2RQ)}\right|}{10} = \frac{5}{10} = 0.5$$

where: $D_r^{(S_2RQ)} = \left\{ d_i \middle| d_i \in D_{10}^{(S_2RQ)}, \exists d_j \in D_c^{(q)} \text{ s.t. } Sim(d_i, d_j) \geq 0.5 \right\}$

For K=20: $$precision@\ 20 = \frac{\left|D_r^{(S_2RQ)}\right|}{20} = \frac{8}{20} = 0.4$$

where: $D_r^{(S_2RQ)} = \left\{ d_i \middle| d_i \in D_{20}^{(S_2RQ)}, \exists d_j \in D_c^{(q)} \text{ s.t. } Sim(d_i, d_j) \geq 0.5 \right\}$

Otherwise we can compute $D_r^{(S_2RQ)}$ based on the user judgment of relevant results from the top k returned results by using SRQ. That means the user evaluates the relevant results himself without using the cosine similarity, but this will require more feedbacks from the user.

In the same method, we can calculate the precision of our system for the other task states in the actual taken scenario and for the others task states in the three experimental scenarios.

In order to quantify the improvement provided by our system compared to the direct querying of a search engine without reformulation or with simple personalization, depending only on the user profile, we calculate the retrieval performance of the standard Google search system and the retrieval performance of the query reformulation system based only on the user profile, by using the same experimental queries in the same experimental scenarios and the same users. Fig. 9 shows a comparison between the Precision@5, Precision@10, Precision@20 averages of our proposed system and those of the standard search without any reformulation and personalized search based only on the user profile. We notice that the precision average of our proposed framework is more precise than the precision average of the standard Google search in the specific task state, and more precise than that of the query reformulation system based on the user profile in the same task state. Thus our retrieval system is more effective at a specific context than that of the classic information retrieval systems and the personalized retrieval systems at the same context.

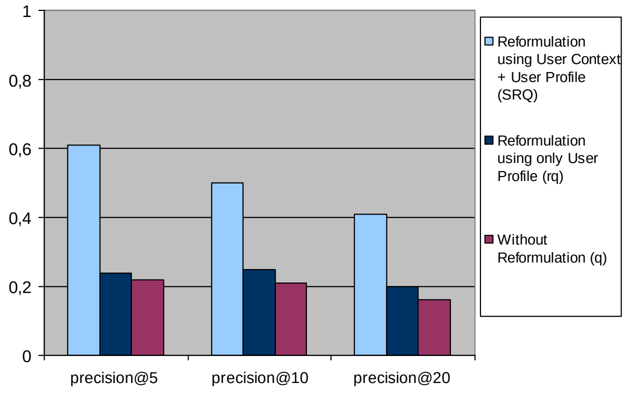


Figure 9. Comparison between the Precision@k averages of the different systems.

3) *Dynamics*

The third evaluation metric is the dynamics in query expansion. For a query *q*, our system of query reformulation returns different expansion terms at different search sessions of the task at hand. Let $E_i^{(q)}$ and $E_j^{(q)}$ be the set of expansion terms for a query *q* at two different task states *i* and *j*, we define the dynamics between the two states *i*, *j* as follows:

$$\delta^{(q)}(i, j) = 1 - Sim\,(E_i^{(q)}, E_j^{(q)})$$

For example, to calculate the dynamics in query expansion terms for the two states S$_1$, S$_2$ of the previous experimental scenario (*travel*) and the query *q= trip Paris*, we have to calculate the similarity between the expansions terms proposed in the two states. The all expansion terms in this two states S$_1$, S$_2$ are 9 terms, there are 2 common terms, and thus the similarity between these two states is 2/9, and the dynamics will be:

$$\delta^{(srq)}(s_1, s_2) = 1 - Sim\,(E_{s_1}^{(srq)}, E_{s_2}^{(srq)}) = 1 - (\frac{2}{9}) = 0.78$$

In the same method we can calculate the dynamics in query expansion terms of the other states and for the three experimental scenarios. Fig. 10 shows the average of the dynamics in query expansion over the experimental queries which are submitted during the three proposed scenarios.

In fact the personalization-based query expansion systems have a dynamics of zero in all cases, because these systems always return the same expansion terms in all task states irrespective of user's search goal or task states, because the expansion terms, in this case, are based on the user's profile only.

We notice from Fig. 10 that our proposed system has a small dynamics in the expansion terms among the states of the simple tasks, such as scenario 2 (*shopping*), and it has a high dynamics in expansion terms among the states of the complex tasks, such as task in scenario 1 (*travel*). Thus our proposed framework is able to adapt to the changing needs of the users and generate expansion terms dynamically.

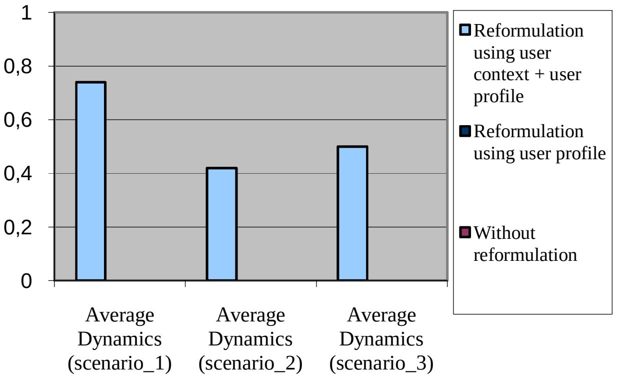


Figure 10. The average dynamics in query expansion terms for our system in the three experimental scenarios.

C. Discussion

From the various experiments, we observed that our proposed framework provides more relevant expansion terms compared with the query expansion mechanisms based on user profile only. Most importantly, our system can dynamically adapt to the changing needs of the user by generating state reformulated queries for the initial user query q in each search session. These generated queries SRQ will be different from one task state to another for the same user and the same initial query q. Consequently these queries SRQ provide more relevant results, in a specific context, compared with the results returned by the standard information retrieval system IRS using the initial user query q or the results returned by the personalized information retrieval systems.

In fact we notice from the experiments that our system is more effective when the user is not expert in computer science because he/she needs an aide to formulate the query that reflects his/her needs. Also our system is effective when the user needs are vague, especially when he is in the context of performing one task. Our system is also effective when the user query is short, so the query expansion will lead to disambiguate the query and to provide relevant results. Because the queries of mobile users are often short, and their information needs are often related to contextual factors to perform one task, thus our system is more effective in providing relevant results for mobile users. In addition, we notice that our proposed system is more effective when the task has many clear and different states (such as the travel task). In this case our system has high dynamics in expansion terms among the states of this task. Whereas the proposed system is less effective with the simple tasks (such as shopping task), in this case our system has small dynamics in the expansion terms among the states of this task types.

One of the system disadvantages, which has emerged during the experiments, that when the expansion terms increase greatly the precision of our system will decrease, but we cannot determine a specific ideal number of expansion terms. Indeed the limitation of our experiments is the manual relevance judgments by several users; this is due to the dynamic aspect of our system and the absence of a standard test collection for the context-based personalized information retrieval systems.

However the experiments show that our approach of context-based information retrieval can greatly improve the relevance of search results.

V. CONCLUSION AND FUTURE WORKS

We have proposed a hybrid method to reformulate user queries depending on an ontological user profile and user context, with the objective of generating a new reformulated query more appropriate than that originally expressed by the user. The objective of the new reformulated query denoted State Reformulated Queries SRQ is to provide the user with context-based personalized results, we proved in an experimental study that these results are more relevant than the results provided by using the initial user query q and those provided by using the user query with simple personalization, depending only on the user profile, in the same context, because the user profile is not relevant all the time, thus we consider only the preferences that are in the semantic scope of the ongoing user activity for personalization, and disregard those are out of focus for a given context.

In this paper, the user context describes the user's current task, its changes over time and its states, i.e., to define the user context; we define the task which the user is undertaking when the information retrieval process occurs and the states of this task. The stages of the task performance are called task states and the transition from one stage to another means that the user has completed this stage of the current task.

Consequently the user queries which are submitted during the task at hand are related to this task, indeed that are part of it. Because the queries of mobile users are often short, and their information needs are often related to contextual factors to perform task at hand, thus an intelligent assistant that can propose new reformulated query before submitting it to the information retrieval system is more effective in the case of a mobile user. Therefore our system is more useful in providing relevant results for mobile users.

On the other hand, we initialize a user profile by using mass of information existing on his/her workstation (personal files), and next we retrieve relevant elements from this profile to use them in query reformulation. In our system the user profile is ontological because it is constructed by considering related concepts from existing concepts in domain ontology (such as ODP taxonomy). Our proposed approach involves new methodology to retrieve query-related elements from the ontological user profile. This methodology has been applied successfully to retrieve information from the semi-structured data.

We have constructed a general architecture that combines several models: task model, user profile model and SRQ model. And we have constructed a new general language model for query expansion including the contextual factors and user profile in order to estimates the parameters in the model that is relevant to information retrieval systems.

We use both a semantic knowledge (ODP Open Directory Project taxonomy) and a linguistic knowledge (WordNet) to improve web querying processing because the linguistic knowledge doesn't capture the semantic relationships between concepts and the semantic knowledge doesn't represent linguistic relationships of the concepts. Parsing the user query by the two previous types of knowledge generate the so-called query context. We proved that the integration of linguistic and semantic information into one repository was useful to learn user's task.

UML activity diagram is used to represent the user's current task in order to detect the changes over time in the activities needed to accomplish this task and for describing all the sequences of the performed task. Each activity in the generated UML activity diagram expresses the task's actual state.

Our "State Reformulated Query" system has been implemented in a prototype and applied to web queries. We had achieved an experimental study using few scenarios by several users; the preliminary results from the prototype are encouraging. Also we proposed an evaluation protocol which uses three evaluation metrics to cover the evaluation of the expansion terms and the evaluation of returned results. The aim is to quantify the improvement provided by our system compared to the personalized reformulation query systems and the standard search without reformulation. From the various experiments, we have proved that the proposed framework provide more relevant results compared to the standard information retrieval system and the baseline query expansion mechanisms based only on the user profile. Thus, the experiments showed that our proposed context-based approach for information retrieval can greatly improve the relevance of search results.

A. Future Works

This research can be extended in several directions. Firstly to optimize the quality of generated terms and then the precision of results, secondly to optimize the detection of the user's task and its states by improving the task model.

To facilitate the use of the contextual model, we can use the contextual graph [28], instead of UML activity diagram to represent the user's current task. In our future work we plan to use this contextual graph.

In future work for this research, we propose to use a Markov models to select the actual task state implicitly by predicting from a number of observed events, the next event from the probability distribution of the events which have followed these observed events in the past. For example, when the task at hand consists of predicting WWW pages to be requested by a user, the last observed event could be simply the last visited WWW page or it could contain additional information, such as the link which was followed to visit this page or the size of the document.

In perspective we can also improve the assistant of generating reformulated queries (SRQ) to be more intelligent by using the ChatBot technique; that means the assistant can chat with a user in order to focus on the actual task state.

Further validation by using different types of queries and domains is required to provide more conclusive evidence. Further work is also needed to determine the circumstances under which the approach may not yield good results.

We plan also to evaluate this method by using another evaluation protocol by constructing a test collection and determining relevant results for several queries in a particular context, and next comparing between these relevant results and the results that are returned by our system for the same queries in the same context.

References

[1] O. Asfari, B-L. Doan, Y. Bourda and J-P. Sansonnet, "Context-based Hybrid Method for User Query Expansion", Proceedings of the fourth international conference on Advances in Semantic Processing, SEMAPRO 2010, pp. 69-74, Italy, Florence, 2010.

[2] A. Micarelli, F. Gasparetti, F. Sciarrone and S. Gauch, "Personalized Search on the World Wide Web", The Adaptive Web 2007, P. Brusilovsky, A. Kobsa, and W. Nejdl (Eds.), LNCS 4321, pp. 195-230, Springer-Verlag Berlin Heidelberg 2007.

[3] Ph. Mylonas, D. Vallet, P. Castells, M. Fernández, and Y. Avrithis, "Personalized information retrieval based on context and ontological knowledge", Knowledge Engineering Review 23(1), special issue on Contexts and Ontologies, pp. 73-100, March 2008.

[4] A. Kofod-Petersen and J. Cassens, "Using Activity Theory to Model Context Awarenessa", American Association for Artificial Intelligence, Berlin, 2006.

[5] R. W. White and D. Kelly, "A Study on the Effects of Personalization and Task Information on Implicit Feedback Performance", CIKM'06, USA, 2006.

[6] S. Dumais, E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins, (Stuff I've Seen) "A system for personal information retrieval and re-use", Proceedings of 26th ACM SIGIR 2003, pp. 72-79, Toronto, July 2003.

[7] M. Melucci, "Context modeling and discovery using vector space bases", CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management. ACM, Bremen, Germany, pp. 808-815, 2005.

[8] K. Sugiyama, K. Hatano and M. Yoshikawa, "Adaptive Web Search Based on User Profile Constructed without Any Effort from Users", WWW 2004, pp.17-22, New York, USA, May, 2004.

[9] X. Shen, B. Tan and C. Zhai, "Implicit User Modeling for Personalized Search", CIKM'05, Bremen, Germany, 31 November, 2005.

[10] B.J. Jansen, A. Spink, J. Bateman and T. Saracevic, "Real life information retrieval: a study of user queries on the Web", SIGIR Forum 32(1): pp. 5-17, 1998.

[11] G. Koutrika and Y. E. Ioannidis, “Personalization of Queries in Database Systems”, Proceedings of the 20th International Conference on Data Engineering, USA, 2004.

[12] Y. Lv and C. Zhai, “Adaptive Relevance Feedback in Information Retrieval”, CIKM, 2009, Hong Kong.

[13] H. Wakaki, T. Masada, A. Takasu, and J. Adachi, “A New Measure for Query Disambiguation Using Term Co-occurrences”, Lecture Notes Computer Science, NUMB 4224, pp. 904-911, 2006.

[14] J. Bhogal, A. Macfarlane and P. Smith, “A review of ontology based query expansion, Information Processing and Management”, International Journal, v.43 n.4, pp. 866-886, July, 2007.

[15] R. Navigli and P. Velardi, “An Analysis of Ontology-based Query Expansion Strategies”, Workshop on Adaptive Text Extraction and Mining at the 14th European Conference on Machine Learning, Cavtat-Dubrovnik, Croatia, 2003.

[16] H. Terai, H. Saito, M. Takaku, Y. Egusa, M. Miwa and N. Kando, “Differences between informational and transactional tasks in information seeking on the web”, Proceedings of the Second Symposium IIiX, 2008.

[17] L. Freund, E.G. Toms and C. Clarke, “Modeling task-genre relationships for IR in the workplace”, SIGIR 2005, Salvador, Brazil, 2005.

[18] D. Leake, R. Scherle, J. Budzik and K. Hammond, “Selecting Task-Relevant Sources for Just-in-Time Retrieval”, Proceedings of the AAAI-99 Workshop on Intelligent Information Systems, Menlo Park, CA, 1999.

[19] J. Luxenburger, S. Elbassuon and G. Weikum, “Task-aware search personalization”, Proceedings of the 31st annual international ACM SIGIR, Singapore, 2008.

[20] H. Bouchard and J. Nie, “Modèles de langue appliqués à la recherche d'information contextuelle”, Proceedings of CORIA 2006 Conf en Recherche d’Information et Applications. pp. 213-224, Lyon, 2006.

[21] J. Allan, “Challenges in information retrieval and language modeling”, report of a workshop held at the center for intelligent information retrieval, University of Massachusetts Amherst, SIGIR Forum, 37, 1, pages 31-47, 2003.

[22] E. Garcia, “Cosine Similarity and Term Weight Tutorial”, http://www.miislita.com/information-retrieval-tutorial/cosinesimilarity-tutorial.html, pp.187-219, 2006.

[23] P. A Chirita, W. Nejdl, R. Paiu and Ch. Kohlschutter, “Using ODP Metadata to Personalize Search”, Proc. of the 28th Annual Int’l ACM SIGIR Conf., pp. 178-185, Salvador, Brazil, 2005.

[24] A. Sieg, B. Mobasher and R. Burke, “Ontological User Profiles for Representing Context in Web Search”, Proceedings of the Workshop on Web Personalization and Recommender Systems in conjunction with the ACM International Conference on Web Intelligence, Silicon Valley, CA, November 2007.

[25] O. Asfari, Modèle de recherche contextuelle orientée contenu pour un corpus de documents XML, The fifth Francophone Conference on Information Retrieval and Applications, CORIA ET RJCRI 2008, pp. 377-384, Trégastel, France, 2008.

[26] Y. Yang and B. Padmanabhan, “Evaluation of Online Personalization Systems: A Survey of Evaluation Schemes and a Knowledge-Based Approach”, Journal of Electronic Commerce Research, pp. 112-122, 2005.

[27] R. Kraft, C. Chang, F. Maghoul and R. Kumar,“Searching with Context”, WWW'06: Proceedings of the 15th international conference on World Wide Web. ACM, Edinburgh, Scotland, pp. 367-376, 2006.

[28] P. Brézillon, “Task-realization models in Contextual Graphs”, 5th International and Interdisciplinary Conference on Modeling and Using Context, Lectures Notes in Artificial Intelligence, Vol 3554, pp. 55-68, Springer-Verlag, 2005.