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

Ounas Asfari

UNIVERSITE PARIS-SUD 11

ÉCOLE DOCTORALE *Informatique*
Equipe E3S (EA 4454) de *Supélec*

THÈSE
Pour obtenir le grade de
DOCTEUR DE L'UNIVERSITÉ - PARIS-SUD 11
DISCIPLINE *Informatique*
Présentée et soutenue publiquement le 19/09/2011
Par

Ounas ASFARI

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

Directeur de thèse: Jean-Paul SANSONNET Directeur de Recherche CNRS, LIMSI
Co-directrice de thèse: Bich-Liên DOAN Enseignant-chercheur, Supélec

Composition du jury

Rapporteurs:	Gilles FALQUET	Professeur, université de Genève
	Mohand BOUGHANEM	Professeur, université Paul Sabatier
Examinateurs:	Anne VILNAT	Professeur, université Paris-sud-11
	Yolaine BOURDA	Professeur, Supélec

Résumé

L'accès à une information pertinente, adaptée aux besoins et au contexte de l'utilisateur est un challenge dans un environnement Internet, caractérisé par une prolifération de ressources hétérogènes. Les travaux présentés dans cette thèse rentrent dans le cadre de la Recherche d'Information (RI) et s'intéressent à la prise en compte du contexte de l'utilisateur pendant la phase de requête.

Nous proposons un assistant d'aide à la reformulation de requêtes reposant sur l'utilisation d'une méthode hybride d'expansion de requêtes afin de fournir des résultats personnalisés en fonction du contexte. Cet assistant utilise le profil de l'utilisateur, qui contient les centres d'intérêts et les préférences des utilisateurs, et utilise également le contexte de l'utilisateur qui considère l'état actuel de la tâche courante de l'utilisateur pendant le processus de recherche.

Une implémentation de cette approche est réalisée, suivie d'une étude expérimentale. Nous proposons également une procédure d'évaluation qui tient compte l'évaluation des termes d'expansion, générés par notre système de reformulation de requêtes, et de l'évaluation des résultats retournés en utilisant les requêtes reformulées SRQ. Nous montrons sur plusieurs scénarios que notre approche, en particulier celle qui prend en compte la tâche actuelle de l'utilisateur est effectivement plus performante que les approches reposant soit uniquement sur la requête initiale, ou encore sur la requête reformulée en considérant uniquement le profil de l'utilisateur.

Mots-Clés: Recherche d'information, Reformulation de requêtes, Contexte de l'utilisateur, Modélisation des tâches, Personnalisation, Profil utilisateur.

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 increase of heterogeneous resources available on the web. Information needs are expressed via queries. Most often, these queries are short, ambiguous and don't grasp the neither the intention nor the implicit need of the user. For improving user query processing, we present a context-based hybrid method for query expansion that automatically generates 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. 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.

Acknowledgments

This research project would not have been possible without the support of many people.

I gratefully thank Professor Mohand Boughanem and Professor Gilles Falquet for their constructive comments on this thesis, whose detailed reading and feedback helped on the improvement of the final version of this PhD. I am thankful that in the midst of all their activities, they accepted to be members of the reading committee. Also I would like thank Professor Anne Vilnat for giving me the honor to be a member of the jury of my thesis.

I would like to express my gratitude to my supervisor, Mrs. Bich-Liên Doan who was abundantly helpful and offered invaluable assistance, support and guidance, whose expertise, understanding, and patience, added considerably to my graduate experience.

Deepest gratitude is also due to Professor Jean-Paul Sansonnet, my PhD supervisor, without his knowledge and assistance this study would not have been successful.

A special thank goes out to Professor Yolaine Bourda head of the Department of Computer Science, without her motivation and encouragement I would not have considered a graduate in this thesis. She had provided me with direction and technical support and she had given me the opportunity to realize this thesis in excellent conditions. I owe her my gratitude.

I would like to thank the members of the Department of Computer Science at Supélec for sharing the literature and invaluable assistance. I feel lucky to have had the chance to make this research with such a good team, which surely allowed me to progress on my research at a solid pace. I would like to thank the current and past members of this team for their company and guidance, and for making these years so enjoyable.

I would also like to thank my family for the support they have provided me through my entire life. No words would be sufficient to thank my parents for all they

have done for me. This thesis is dedicated to them, as well as my brother Ahmed and my two sisters and their husbands.

I am especially thankful to my best friends. Thank you for your ongoing friendship, support and all the good times we have had.

Table of Contents

List of Figures

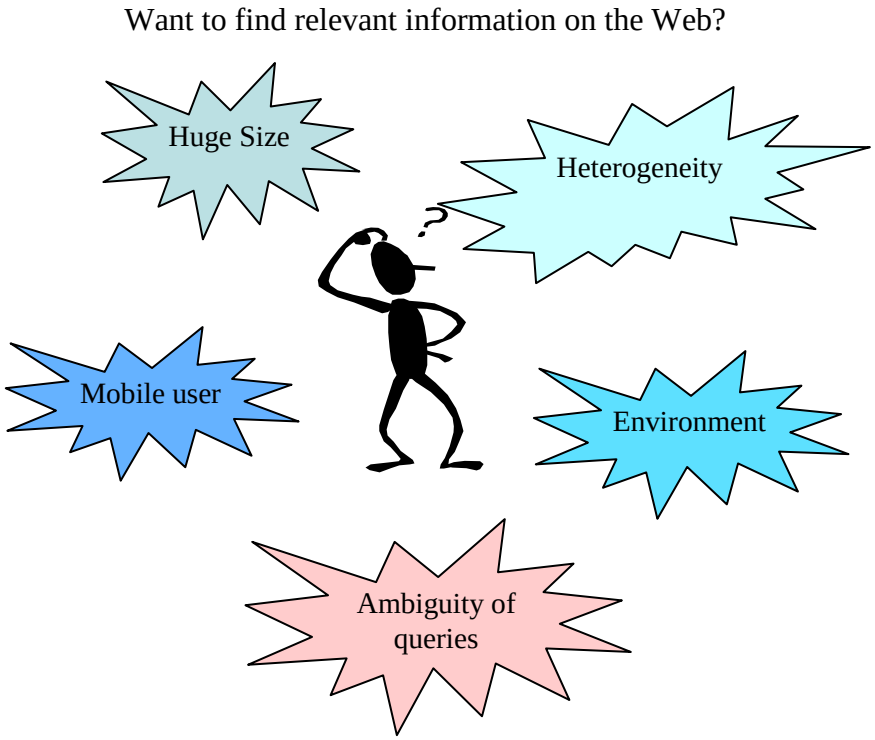
List of Tables

Chapter 1

Introduction

1.1. Motivations

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. Therefore it is difficult to obtain desired results from the returned results by IRS.

The example shown in Figure 1.1 below illustrates some of the difficulties. In this example, the query "*Trip Paris*" results (in September 2011) about 354 million documents, and there are five different themes in the top 10 documents selected by the search engine. Also the user queries are generally shorts and contain words with several meanings.

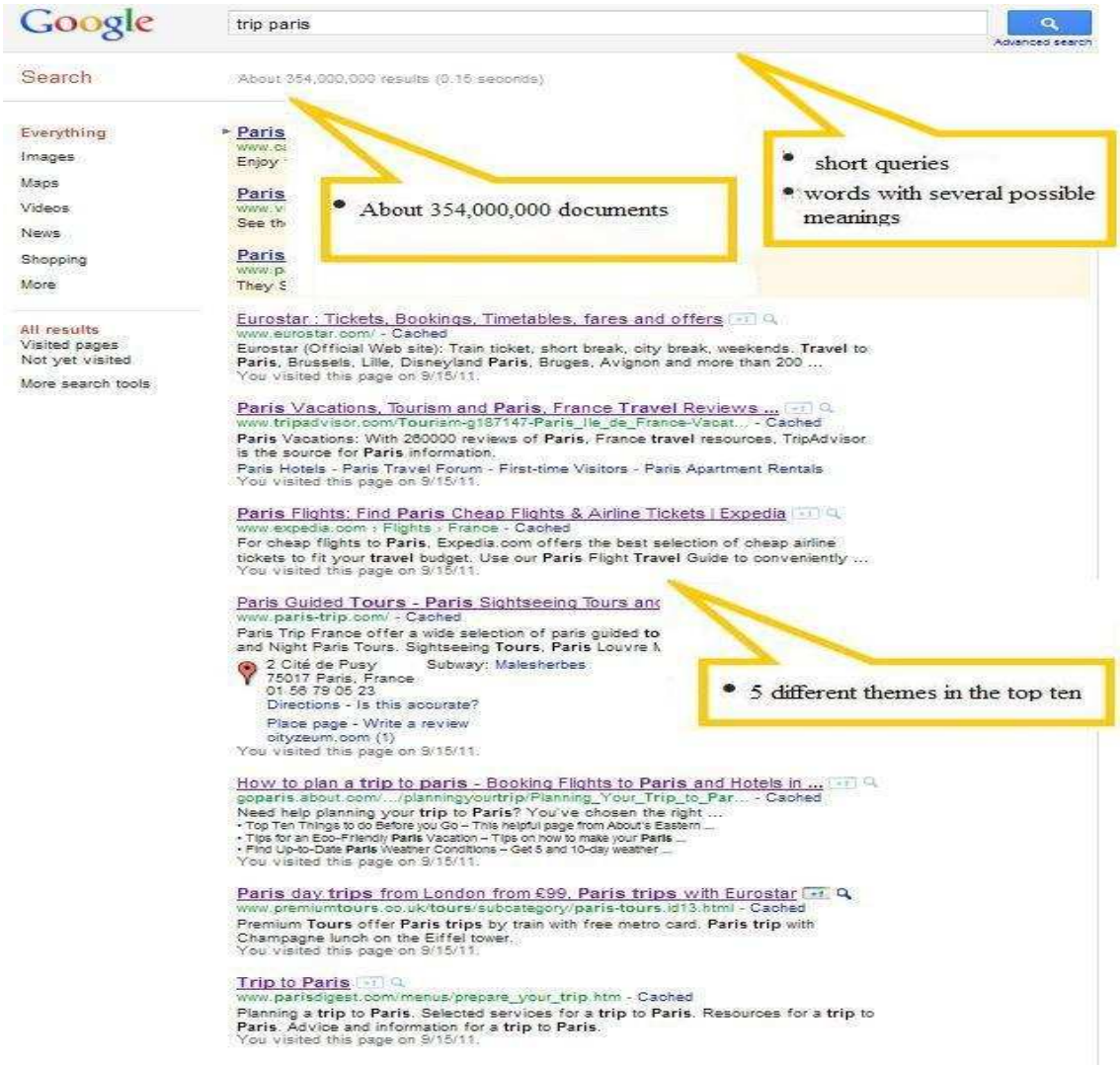


Figure 1.1. Example of a query and results.

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 have tried to enhance a user query with user's preferences, by creating a dynamic user profile, in order to provide personalized results (Micarelli et al., 2007). 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's 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 (Mylonas et al., 2008), (Sieg et al., 2007 a). The user context can be assimilated to all factors that can describe his intentions and perceptions of his surroundings (Mylonas et al., 2008). These factors may cover various aspects: environment (light, services, people…), spatial-temporal (location, time, direction…), personal (physiological, mental, professional …), social (friends, colleagues…), task (goals, information task), technical etc.

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 (White and Kelly, 2006). 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 (Dumais et al. 2003),

physical user location (Melucci, 2005), recently visited Web pages (Sugiyama et al. 2004), session interaction data (Shen et al. 2005a), etc.

In this thesis 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 thesis 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 (Figure 1.2). 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.

Figure 1.2. Using mobile devices to seek information needed to perform a task.

Here, the problem is that most of the classic information retrieval systems don't 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 (Jansen et al. 1998)) 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 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.

1.2. Our Proposed Solution

The research, presented in this thesis, 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 which 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 this thesis 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.

Therefore, this thesis 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. The objective of this new query SRQ is to provide the user with context-based personalized results; we will prove that these results are more relevant than the results 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.

In fact 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.

1.3. Contributions

The main original contributions of the research presented in this thesis include the following:

- Context is a broad notion in many ways. One of the aims of the research undertaken in this thesis is to identify a new kind of contextual analysis performed during the information retrieval process, which views the context as the user's current task, its changes over time and its states, i.e. we take into account the task which the user is undertaking when the information retrieval process occurs. We will show that it is possible to determine the user task automatically by exploiting both a semantic knowledge like Ontology (for example: ODP[1] taxonomy) and a linguistic knowledge (like WordNet), and we use UML activity diagram to represent the task. When the user identifies his/her actual task's state in the UML diagram then the system can follow this diagram to detect the next possible task states.

- We 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 purpose of *query context* is to use a variety of knowledge involving query to explore the most exact understanding of user's information needs.

- Our proposed approach involves an ontological user profile which is constructed by selecting interest concepts related to user's files collection, from existing concepts in domain ontology. We use the ODP (Open Directory Project) as ontology to generate the concepts. Also we propose a methodology to retrieve query-context-related terms from the ontological user profiles; these terms form the operational profile. The methodology depends on the semi-structured data retrieval.

- We propose a hybrid method to reformulate user queries depending on the user profile and his/her context for producing State Reformulated Queries SRQ, in order to provide the most appropriate answer for a user's information needs in the search time and at the actual state of the undertaken task.

[1] ODP: Open Directory Project: www.dmoz.org

- We construct a general architecture that combines several models for query expansion: Task model, User profile model and SRQ model.
- We 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 order to evaluate our proposed approach for query reformulation, three evaluation metrics are defined to cover the evaluation of the proposed expansion terms and the evaluation of returned results. The aim of the proposed experimental methodology is 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 user profile only.

1.4. Thesis Outline

This thesis is structured in five main Chapters.

In *chapter 2* we provide an overview of a related work on the State of the Art of personalized and context-aware retrieval systems. We also provide an overview of a several techniques of query reformulations, and a background on the task models.

In *chapter 3* we introduce the core part of this thesis, we describe our methodology, based on models to represent the user task and the user profile in order to generate relevant terms which are used to reformulate user query for providing personalized results in context. We also present the system architecture in this chapter.

In *chapter 4* we outline the implementation details of our system, and report the results of evaluating the proposed methodology empirically. We include some test scenarios and validate it by our methodology. We discuss the results of the experimental study in the context of prior work and related literature.

Finally in *chapter 5*, we state our conclusions and outline directions and ideas for future work.

Chapter 2

Background Research

The aim of this chapter is to gather existing techniques, approaches, and ideas from the fields of personalization, user modeling, context aware, and background in a number of related areas in order to provide a firm base for the techniques used and developed within our system. Various problems and techniques are discussed in preparation for the description of our own methods in chapter 3.

In order to solve the problems listed in the first chapter and provide personalized results in context to the user query, solutions are proposed. We can classify these solutions in two areas, personalized information retrieval and contextual information retrieval. Section 2.1 presents basics of *information retrieval* area. Section 2.2 reviews the existing approaches in the *personalization* area and the limitations of these approaches, while Section 2.4 presents the *context* aware systems and the contextual information retrieval approaches.

Our proposed solution, in this thesis, is an intelligent assistant that can generate 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 current task. Thus we need to review the different techniques in *query reformulation*; this will be shown in Section 2.6. As long as we depend on the user profile and the user task to reformulate the query, Section 2.3 describes the representation and the use of *user profile* and Section 2.5 reviews the task model and some approaches. Also, in Section 2.7, we review the systems that use an agent or an assistant in Information Retrieval this is because we use an assistant to reformulate user queries as it is previously explained.

2.1. Information Retrieval System IRS

2.1.1. Definitions

An Information Retrieval System (IRS) is a software tool for data representation, storage and information search. The aim is 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. We use the word *document* as a general term that could also include non-textual information, such as multimedia. Information retrieval (IR) deals with the representation, storage, organization, and access to information items, (Baeza-Yates et al., 1999).

Figure 2.1 shows a general architecture of an information retrieval system which is composed of three principle parts:

- Index: structure to organize and represent a collection of documents in order to make the search for information in these documents effective.
- Query: representation of the information user needs.
- Matching function: function that matches user queries to indexed documents. It is based on Information Retrieval Models developed to provide scores to documents in response to a query. The matching function is used by the Information Retrieval System to retrieve documents in decreasing order of relevance. The classic IR models include *Boolean* model, *vector space* and *probabilistic* models.

Evaluating the performance of Information Retrieval Systems is crucial. The measures of the evaluation require a collection of documents and queries. A document is relevant if it addresses the needed information (user relevance), not just because it is estimated as relevant by the system (system relevance). To evaluate classic IRS, standard collections such as TREC, GOV have been created and used until now.

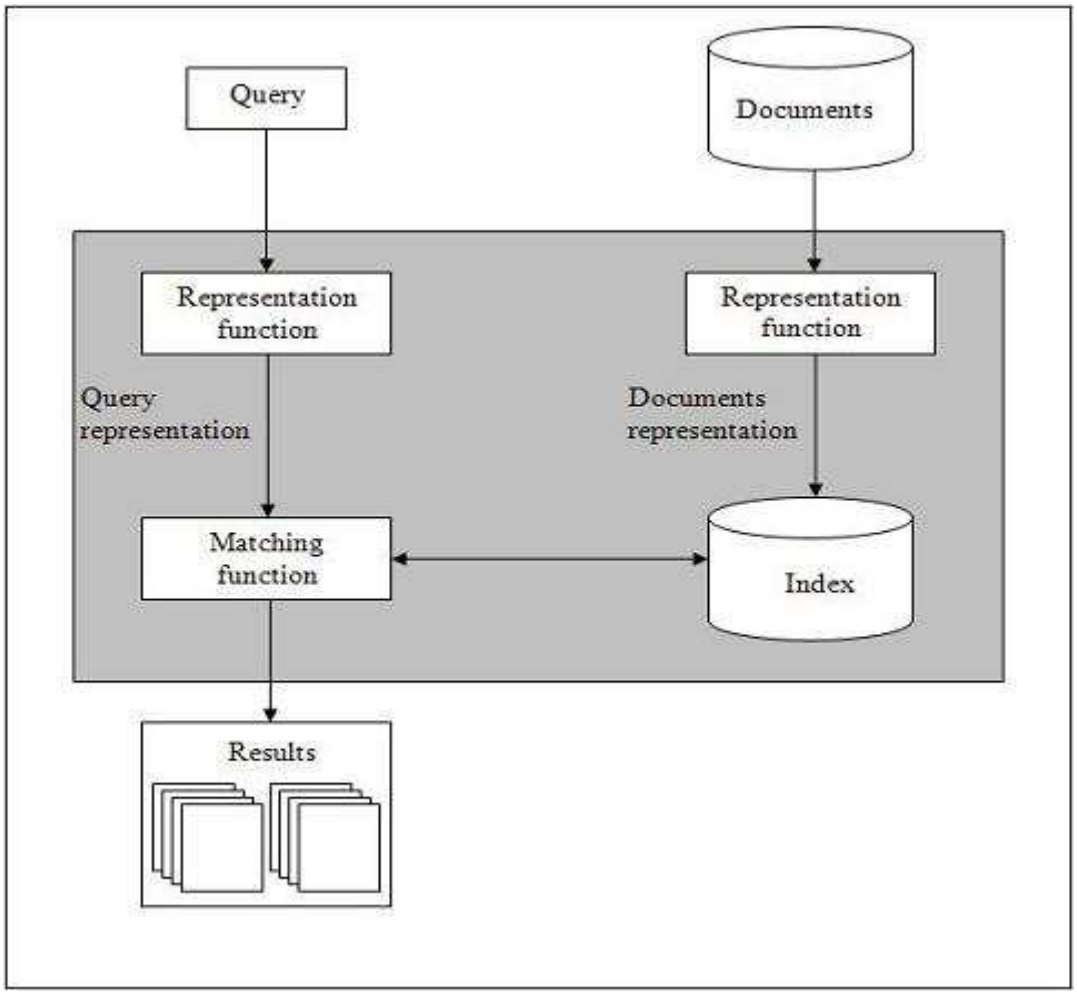


Figure 2.1. Information retrieval system architecture.

Due to the massive amount of information that is available now, the process of information retrieval tries to select numerous and heterogeneous documents as result of a single query. The reason is that the system cannot acquire adequate information concerning the user's wish. Traditionally, Information Retrieval Systems IRS allow the users to provide a small set of keywords describing their wishes, and attempt to select the documents that best match these keywords.

Thus, the main limitation of most existing information retrieval systems comes from the fact that they only depend on the query and document collection; information about the user and search context is largely ignored.

2.1.2. Taxonomy of Web Searches

On the web the need behind the query might be: informational, navigational, and transactional. We classify web queries according to their intent into 3 classes (Gabrilovich et al., 2009), (Broder, 2002):

A. Navigational queries. The purpose of such queries is to reach a particular site that the user has in mind, either because they visited it in the past or because they assume that such a site exists. For example: hp. Probable target: http://www.hp.com. Navigational queries have usually only one right result.

B. Informational queries. The aim is to acquire some information assumed to be present on one or more web pages. No further interaction is predicted, except reading. In any case, informational queries are closest to classic IR and therefore need less attention here.

C. Transactional queries. The aim is to perform some web-mediated activities. The purpose of such queries is to reach a site where further interaction will happen. This interaction constitutes the transaction defining these queries. The main categories for such queries are shopping, finding various web-mediated services, downloading various type of file (images, songs, etc), accessing certain data-bases (e.g. Yellow Pages type data), finding servers (e.g. for gaming) etc. The results of such queries are very hard to evaluate in terms of classic IR. Binary judgment might be all we have, say appropriate, non-appropriate, (Gabrilovich et al., 2009).

An understanding of this taxonomy is essential to the development of successful web search. Current search engines deal well with informational and navigational queries, but transactional queries are satisfied only indirectly and hence a third generation in search engines is emerging (Figure 2.2 shows this type of information retrieval); its main aim is to deal efficiently with transactional queries mostly via semantic analyses (understanding what the query is about) and blending of various external data bases. Figure 2.2 shows the process of deriving a query from an information need in the web context. We recognize that the information need is associated with some task. This need is verbalized and translated into a query posed to a search engine.

Because the intent of the transactional queries is to perform some activities, thus in our proposed system we assume that the user queries, which are submitted during the

performance of one task, belong to this transactional queries type. We will call them: queries related to task at hand.

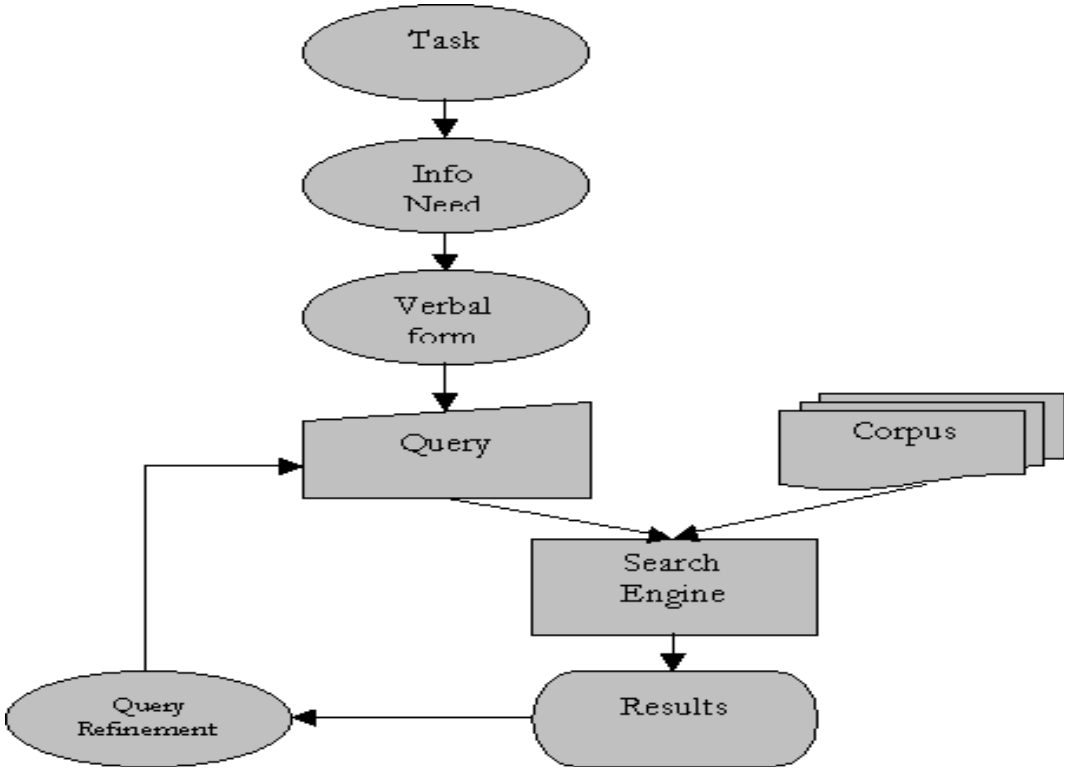


Figure 2.2. Model IR, deal efficiently with transactional queries.

2.2. Personalized Information Retrieval

The classic information retrieval systems generally handle search queries without considering the specific information (intention, context …) related to the users who submit the queries. As a result, it becomes more difficult to obtain desired results than ever due to the ambiguity of user's needs. These systems are inadequate for making a difference among the various information needs of the users. Studies on personalized search have focused on requiring users to explicitly enter their information including interest topics, preferences, etc., and using this information to expand users' queries or re-rank search results (Chirita et al., 2005). However forcing users to submit their interests would be a task that few users would be willing to do.

Furthermore, it is very difficult for users to define their own interests accurately. Much attention has been paid to learn user interests automatically by modeling user profiles or user representations (Qiu et al., 2006), (Micarelli et al., 2007), (Shen et al., 2005 a).

Some approaches emphasize on learning user profiles and utilizing the learned user profiles to re-rank search results, or base on the conceptual similarity between page and user profile (Speretta et al., 2005). Most studies on learning user profile have deemed user profile to be static. A problem occurs when user interests change over time. For instance, if a user changes her vocation from being an IT specialist to a lawyer, it is natural that her interests will shift with this change. It becomes important to keep the user profile up-to-date, and for a search engine to adapt accordingly. Therefore, suitable strategies are needed to capture the accumulation and degradation of changes of user interests, and then adapt the contents and structures of the user profiles to these changes (Li et al., 2007).

Thus, the problem of personalized information retrieval systems has two broad dimensions:

- Collect information about the user (explicitly or implicitly).
- Represent this information and use it to personalize the information retrieval systems.

2.2.1. ODP: The Open Directory Project

We can use ontology as the fundamental source of a semantic knowledge in our framework. Firstly we have to distinguish between taxonomy and ontology. 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, (taxonomy can be considered as a particular type 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 less formal 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. It covers 4 million sites filed into more than 590,000 categories (16 wide-spread top-categories, such as Arts, Computers, News, Sports, etc.). Currently, there are more than 65,000 volunteering editors maintaining it, (ODP contains topic classifications for about 0.05% of all Google indexed pages). 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 (Chirita et al., 2005).

A category in the ODP can be considered a concept that is defined by:

- Label of the concept, for example, ‘Microsoft Windows’.
- Web documents related to the category.
- Parent concepts, e.g. ‘Operating Systems’, ‘Computers’, and the children concepts, for example, ‘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.

Figure 2.3 shows an example of a tree structure that represents some of topics from ODP for the node “*Arts*”.

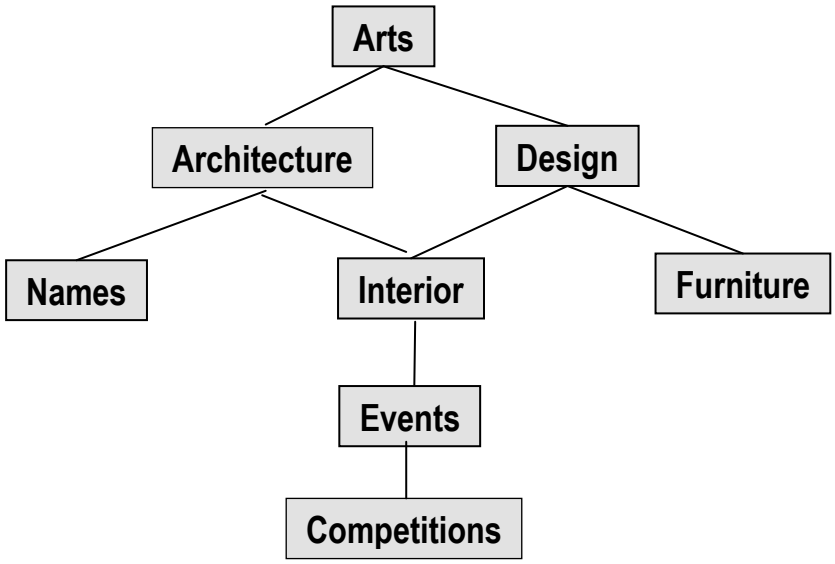


Figure 2.3 Example for tree structure of topics from ODP.

2.2.2. Example of a Personalized Information Retrieval Algorithm

Figure 2.4 (Shen et al., 2005b) presents an example of a possible architecture for a personalized access to information system.

General steps of the personalized search algorithm are:

- Collect information about the user's interests (ex: from the search history), Categorize representative texts into concept hierarchy (for example, we can use ODP, Open Directory Project, for concepts).
- Submit query to information retrieval system or Internet search engine (e.g., Google).
- Categorize each result into same concept hierarchy (e.g., ODP) to create result profiles.
- Conceptual match is calculated based on similarity between each result profile and user profile.
- Re-rank results based on conceptual match, the produced rank order is called "conceptual rank".
- The final rank of the document is calculated by combining the conceptual rank with Google's original rank using the following weighting scheme:

$$\text{Final Rank} = \alpha * \text{Conceptual Rank} + (1 - \alpha) * \text{Google Rank}$$

  - o α has a value between 0 and 1.
  - o The conceptual and search engine based rankings can be blended in different proportions by varying the value of α.

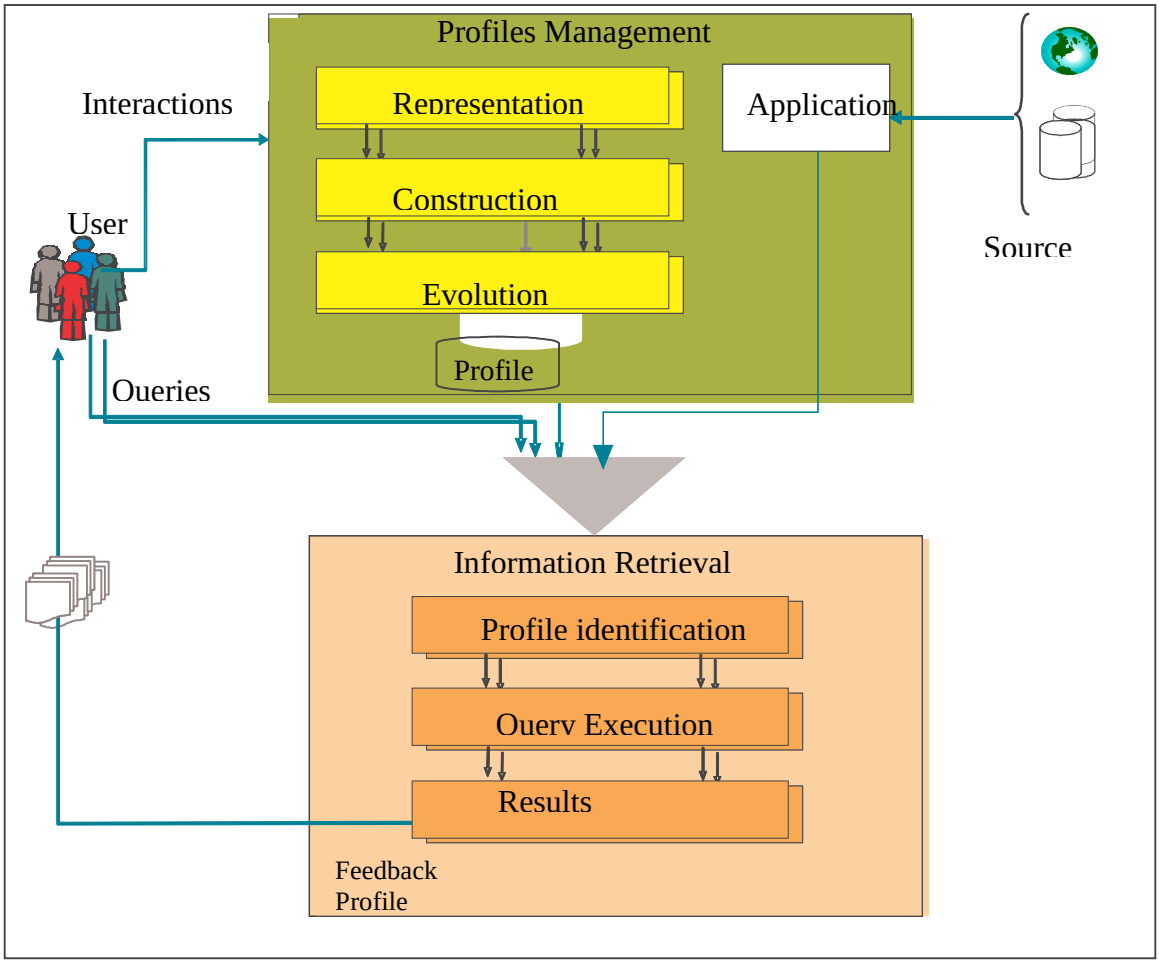


Figure 2.4. Example of an architecture for personalized information retrieval systems.

2.2.3. Web Personalization

Web personalization can be defined as the process of customizing the content and structure of Web pages to the specific and individual needs of each user taking advantage of the user's navigational behavior whether automatically or manually, explicitly or implicitly (Kim W., 2002). Most of the definitions given to Web personalization agree that the steps of the Web personalization process include ((Kim W., 2002), (Cingil et al., 2000)):

- Collection of Web data.
- Modeling and categorization of these data (pre-processing phase).
- Analysis of the collected data.
- Determination of the actions that should be performed.

The ways that are employed in order to analyze the collected data include content-based filtering, collaborative filtering, rule-based filtering and Web usage mining. The site is personalized through the highlighting of existing hyperlinks, the dynamic insertion of new hyperlinks that seem to be of interest for the current user, or even the creation of new index pages.

Content-based filtering systems are based on individual users' preferences. The system tracks each user's behavior and recommends them items that are similar to items the user liked in the past.

Collaborative filtering systems invite users to rate objects or divulge their preferences and interests and then return information that is predicted to be of interest for them. This is based on the assumption that users with similar behavior (for example users that rate similar objects) have analogous interests.

In rule-based filtering the user is asked to answer to a set of questions. These questions are derived from a decision tree, so as the user proceeds on answering them, what he finally receives as a result is tailored to their needs. Content-based, rule-based and collaborative filtering may also be used in combination, for deducing more accurate conclusions.

Web usage mining is process relies on the application of statistical and data mining methods to the Web log data, resulting in a set of useful patterns that indicate users' navigational behavior. The data mining methods that are employed are: association rule mining, sequential pattern discovery, clustering and classification. This knowledge is then used from the system in order to personalize the site according to each user's behavior and profile (Eirinaki et al., 2003).

2.2.4. Approaches

Personalized search could be either server-based or client-based. A server-based search system could keep track of a user's previous queries and selected documents, and use this information to infer user interests. The system in (Ferragina et al., 2005) is an available server-based search engine that unifies a hierarchical web-snippet clustering system with a web interfaces for the personalized search. Google and Yahoo! also supply personalized search services. With the cost of running a large

search engine already very high, however, it is likely that the server-based full-scale personalization is too expensive for the major search engines at present.

On a client-based personalized search, studies in (Shen et al., 2005 b) for example, focus on capturing all the documents edited or viewed by users through computation-consuming procedures. Allowing for scalability, the client-based personalized search could learn user contexts more accurately than the server based personalized search, while it is unavoidable that keeping track of user contexts has to be realized by middleware in the proxy server or client. Users, however, may feel unsafe to install such software even if it is guaranteed to be non-invasive, and may intend to enjoy the services provided by search engines instead. Moreover, if a user changes his/her computer from his/her office to home, keeping his/her contexts consistent becomes a problem.

As examples for the server-based or client-based approaches, here, we will describe some important approaches in the personalized information retrieval area:

(Speretta et al.,2005) creates user profiles by classifying information into concepts from the ODP taxonomic hierarchy and then re-ranks search results based on the conceptual similarity between page and user profiles. They have not taken the hierarchy structure of the ODP into account when calculating the conceptual similarity.

For re-ranking search results, (Chirita et al., 2005) propose a rank mechanism in which a semantic similarity measure is introduced for web page rank with consideration to the hierarchy of the ODP structure. But the technique proposed suffers from the problem of requiring users to select topics which best fit their interests from the ODP.

(Yu et al., 2002) propose a technique to map a user query to a set of categories, which represent the user's search intention. This set of categories can serve as a context to disambiguate the words in the user's query. A user profile and a general profile are learned from the user's search history and a category hierarchy respectively. These two profiles are combined to map a user query into a set of categories. Several learning and combining algorithms are evaluated and found to be effective.

(Challam et al., 2007) present their approach to personalizing search engines using ontology-based contextual profiles. In contrast to long-term user profiles, they construct contextual user profiles that capture what the user is working on at the time they conduct a search. These profiles are used to personalize the search results to suit the information needs of the user at a particular instance of time.

(Li et al., 2007) adapt strategies for modeling user profiles automatically. These strategies are based on click-history data while considering the accumulation and degradation changes of user interests. When user interests change, user profiles, not only in contents, but also in structures, are modified to adapt to the changes. They propose a novel rank mechanism by measuring hierarchy semantic similarities between up-to-date user profiles and web pages.

(Shen et al., 2005 b) study how to infer a user's interest from the user's search context and use the inferred implicit user model for personalized search. The user context, here, is the previous queries and click through information. This method presents a decision theoretic framework and develops techniques for implicit user modeling in information retrieval. They develop an intelligent client-side web search agent that can perform eager implicit feedback, e.g., query expansion based on previous queries and immediate result ranking based on click through information.

(Sieg et al., 2007 b) present an approach to personalized search that construct an ontological profiles by assigning implicitly derived interest scores to existing concepts in domain ontology. A spreading activation algorithm is used to maintain and incrementally update the interest scores based on the user's ongoing behavior.

2.2.5. Limitations

We can summarize some limitations and disadvantages of the personalized systems in the following points:

- Need to contextualization the user interests (the preferences that are out of focus for a given context are disregarded, and only those that are in the semantic scope of the ongoing user activity are considered for personalization).

For example a programmer submits the query "*java*", personalized systems will retrieve results on the Java programming language depend on his profile and will exclude results on the Java Island in Indonesia or on the Java Coffee, but in certain situations the programmer needs information about the Java Island to prepare a trip that is not specified in his profile.

Another example, a person being at beach submits the query "*sport*"; knowing that he is interested both in skiing and surfing, the personalized systems do not take into account the user location to improve search results by taking into account his interests for surfing and not for skiing given that he is at beach and not on a mountain.

- Need to look at combination of short-term, long-term user interests with the current task focus.
- Human preferences are multiple, heterogeneous, changing, even contradictory, and should be understood in context with the user goals and tasks at hand.
- As we mentioned previously, the server-based personalization is too expensive for the major search engines at present because the cost of running a large search engine already very high.
- Also we mentioned that the client-based personalized search keep track of user contexts by middleware in the proxy server or client. Thus users may feel unsafe to install such software even if it is guaranteed to be non-invasive. Moreover, if a user changes his/her computer from his/her office to home, keeping his/her short-term interests consistent becomes a problem.

2.3 User Profile

As long as we depend on a user profile in our approach, this section will describe the representation and the use of user profile in several techniques. Firstly we will describe in this section the concept of user model.

2.3.1 User Model

What is a user model?

- Most systems that interact with human users contain, even if only implicitly, some sort of model of the creatures they will be interacting with, (Elaine Rich) in (Kobsa et al., 2007).
- Systems that tailor their behavior to individual user's needs often have an explicit representation structure that contains information about their users; this structure is generally called a user model, (Robert Kass) in (Kobsa et al., 2007).
- A user model is a representation of the properties of a particular user. If a program can change its behavior based on something related to the user, then the program does (implicit or explicit) user modeling.
- What for? The major concern of user model is to improve the quality of human-computer interactions by inferring and predicting goals, preferences and context of users from the observed facts. In other words, user model is used to adapt to the known facts about the user and to infer properties of the user.
- The user modeling methods invest many areas relating to the implementation of intelligent systems such as systems which use the natural language analysis, system-enhanced learning, adaptive hypermedia systems and in general all customized systems.

Some challenges for a user modeling:

- Identify user goals from low-level interactions.
- Capture the larger context and what users are doing (especially beyond the direct interaction with the computer system).
- Reduce information overload by making information relevant to the task at hand and to the assumed background knowledge of the users.
- Support different descriptions (relate new information to information and concepts assumed to be known by the user).

Any system, that implements user modeling methods, includes a part of the following information package:

- Personal information which are associated with the user such as age, country, and language.
- Preferences: may be at different levels such as preferences of page style, document length, and preference domains to target the user's interests.

- User's history: previous user's interactions which represent a source to predict his/her intentions and to recommend his/her objects.

The user modeling approaches and techniques may be based on simple or complex models depending on the final objective or the system application domain.

2.3.2. User Profile

User profile is a collection of a personal data and stored knowledge which are associated to a specific user. Usually simple profile consists of keywords that describe user's area of long time interest. Extended profile is extended with information about the user such as name, location, mother tongue and so on. Advanced user profile contains rather than set of keywords a list of queries which characterize user's behavior and habits, (Suhail et al., 2005).

User profile can be exploited to make the search task more personalized. Information retrieval system which is equipped with user profiles could utilize user-specific information from the profile for retrieving documents satisfying stated query with special respect to individual user, her or his preferences, needs, abilities, history, knowledge and context. Keywords from the profile can be used for query extension, query reformulation and for other techniques such as for improving search (Snasel et al., 2010).

A user profile can be either static, when the information it contains is never or rarely altered (e.g., demographic information), or dynamic when the user profile's data change frequently, (e.g., all the visited pages can be considered as user interests to various degrees). Such information is obtained either explicitly, using on-line registration forms and questionnaires resulting in static user profiles, or implicitly, by recording the navigational behavior and the preferences of each user, resulting in dynamic user profiles.

In the latter case, there are two further options: either regarding each user as a member of a group and creating aggregate user profiles, or addressing any changes to each user individually. When addressing the users as a group, the used method is the creation of aggregate user profiles based on rules and patterns extracted by applying Web usage mining techniques to Web server logs. Using this knowledge, the Web site can be appropriately customized.

In the following sub-sections of user profile we will review how we can create the two forms of user profile explicit and implicit profile. Also we will review the types of user profiles and finally we will present various approaches for user profiles, these approaches can be classified in main groups such as bag of words, set of concepts, item-based, and user interaction information with IR system.

2.3.2.1. User Profile Creation

Explicit profile

Explicit user profiles are created by users or system administrators by using one of the following methods:

- Ask the user for static information (Name, age, residence location, hobbies, interests, etc).
- Require the user to manually create a profile on the topic by completing a form detailing the main constructs, key attributes of each construct, preferences about each attribute, and preferred instances.
- Present examples: movies, TV shows, music, blog topics, and asks the users to explicitly rate them.
- People specified literature as one of their interests but did not make a single related search (Google personalization).

The main advantage of explicit feedback systems is that there is a higher degree of confidence on the collected information; because the proper user who gives the interest information that system is adapting to. But in general, people do not like to give explicit information frequently. Moreover explicit profiles are not flexible enough and do not reflect dynamic changes of user preferences. Instead, various techniques for automated creation and maintenance of user profiles are being investigated (Cordon et al., 2004).

Implicit profile

Here, the user profile is automatically created and updated. It can be constructed by one of the following data sources without needing any extra interaction of the user:

- A log file that describes various information which are manipulated by different applications and the requests which are addressed to the system.
- Web pages, documents, search queries, location.
- Information from applications (Media players, Games, MSN Messenger …etc.).
- Expand practice queries using WordNet and DAML Ontology.
- Highly ranked snippets are parsed for additional terms to add to the construct list of the implicit user profile terms.
- Search for relevant elements by querying the ontology library and other existing profiles. These elements can be presented to the user and asks him to select relevant elements, and rate their importance.

In fact when we have more riche information, we will have better profile. All documents are better than only recent documents, better than only web pages, better than only search queries, better than no personalization. Drawback with implicit information is cannot collect information about user dislikes.

Types of User Profiles

A user profile may be a knowledge-based profile. That means it reflects the user's knowledge to one domain in the form of semantics. Using domain knowledge requires the system to identify firstly the domain (topic) a user's query pertains to and secondly the user's level of knowledge about this domain (topic). Also a user profile can be behavior-based profiles. That means the profile stores records of user's actions.

Another distinction among profiles is whether the preferences are: personal (i.e., individual), that means each specific user have individual preferences. Or the preferences held by a class of individuals (Stereotype) or held by an entire community (Community type). In the last type the user is addressed as a group, we can create aggregate user profiles based on rules and patterns extracted by applying Web usage mining techniques to Web server logs.

The Semantic Retrieval System chooses a suitable profile based on a user's level of domain knowledge.

2.3.2.2. Approaches

There are various approaches for user profiles representation such as bag of words, set of concepts, item-based, and user interaction information with IR system, (Vallet et al., 2007).

The most common approach for user profile representation is the bag of terms approach like (Widyantoro et al., 2001), where user interests are represented as a set of terms from user documents. Majority of systems, which use this type of approach, express user profiles as a set of weighted terms by using vector model as following:

- The profile is represented as sets of words $tf * idf$ weighted.
- Could use one long profile vector or different vectors for different topics (sports, health, and finance).
- Documents converted to same representation, matched with keyword vectors using cosine similarity.

One problem of these approaches is that they don't consider term correlations. Thus (Liu et al., 2004) link terms by correlation, but in this case the correlation is based on co-occurrence on a predefined set of categories, obtained from the Open Directory Project ODP. (Chirita et al., 2006) cluster the terms that are extracted from the user documents and the terms are only weighted by term frequency that means the number of the term appear in the documents. (Koutrika et al., 2005) link terms with logical operators, which indicate operations of negation, addition or substitution related to a given term.

To overcome the drawbacks of the bag of terms approaches, some approaches like (Schickel-Zuber et al., 2006) try to score user interests and concept similarity based on the structure of ontology. This is the second type approaches where user profile is represented with a set of concepts in order to add more semantics to this representation. These concepts have a background knowledge, which usually adds new relations between concepts. But here the problem is that the proposed approaches need users to express their interests by rating a given number of items explicitly. Another drawback is that these studies neglect that user interests could change over time. To resolve this last point, the study, in (Widyantoro et al., 2001) for example, proposes a multiple three-descriptor representation introduced to learn changes in

multiple interest categories, but it also needs positive and negative relevance feedbacks which are provided by users explicitly.

Other approaches are item-based, in these approaches the user profile is represented as a set of documents that the user has interest in (e.g. a set of bookmarks or documents). The personalization systems will try to extract interests from these documents' content or to use intra-document relations to find more interesting documents. (Zigoris et al., 2006) represent user profile as a graph, where users are nodes that link to the preferred documents. (Martin et al., 2004) define a workspace where the user can stored their interesting documents. In order to establish manual relations between documents, they define the concept of bundle, similar to the concept of folder in an Operating System, where the user can store related documents.

Another approach is collecting interaction information of the user with the information retrieval system. Typically this is done by collecting click through data of past interaction of the users that can provide useful information to extract user's interests. This hypothesis is existed in many systems, which exploit this kind of implicit information in order to construct the user profile, for example (Tan et al. 2006). Another example, the system in (Li et al., 2007), they propose a strategy to capture the accumulation and degradation changes of user interests automatically based on the visited pages that can be considered as user interests to various degrees because the users have accessed them. They suppose that sufficient contextual information is hidden in the web log.

2.4. Context Modeling

2.4.1. Notion of context

Context is a broad notion in many ways. In order to address some of the limitations of classic personalization systems, researchers have looked to the new emerging area defined by the so-called context-aware systems. In this scope, the term *context* can take on many meanings and there is not one definition that is felt to be globally satisfactory and that covers all the ways in which the term is used. The term has a long history in diverse areas of computer science, namely in artificial intelligence, IR,

image and video analysis, context-sensitive help, multitasking context switch, psychological contextual perception, and so on. We can mention some definitions:

In the early works that introduce the term *context-aware*, (Schilit et al., 1994) refer to context as location, identities of nearby people and objects, and changes to those objects. In a similar definition, (Brown et al., 1997) define context as location, identities of the people around the user, the time of day, season, temperature, etc. (Ryan et al., 1997) define context as the user's location, environment, identity and time.

(Dey et al., 1999) define context as any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

In other studies, the user context can be assimilated to all factors that can describe the user intentions and perceptions of his surroundings (Mylonas et al., 2008). These factors, called contextual factors, may cover various aspects such as:

- Environment (light, services, people…).
- Spatial-temporal (location, time, direction…).
- Personal (physiological such as "physical ability, age", Mental such as: "interests, expertise", professional …).
- Social (friends, colleagues…),
- Task (goals, information task), technical…etc.

Figure 2.5 shows these factors and examples for each one (Kofod-Petersen et al., 2006).

The effective use of contextual information in computing applications still remains an open and challenging problem. Several researchers have tried over the years to categorize context-aware applications and features, including contextual sensing, contextual adaptation, contextual resource discovery and contextual augmentation (the ability to associate digital data with a user's context).

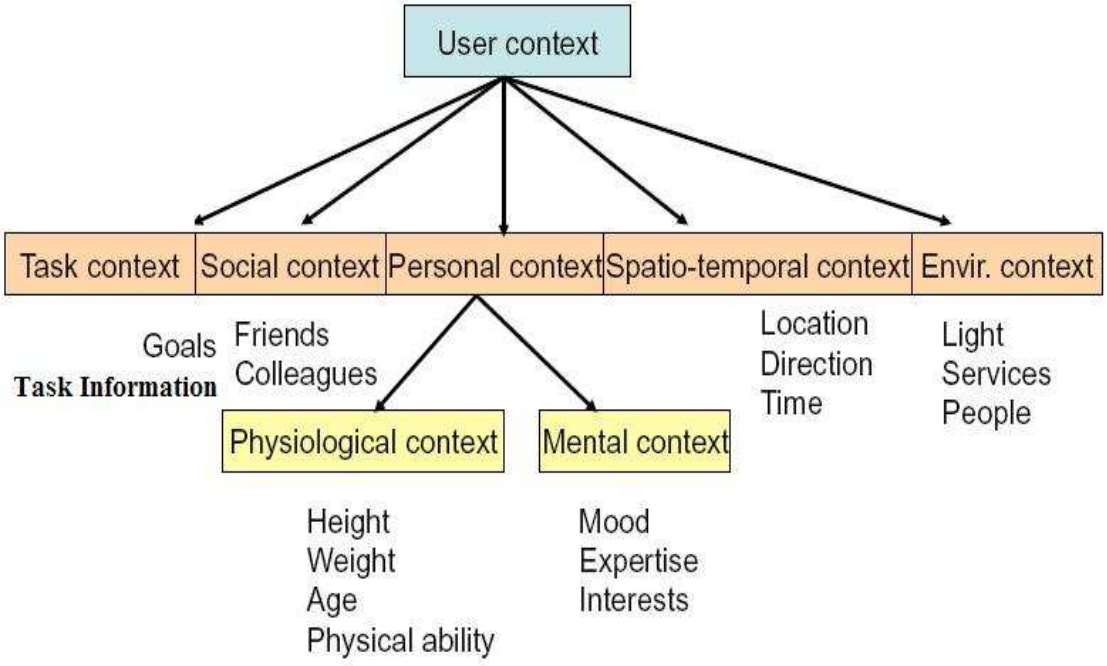


Figure 2.5. Context Model.

2.4.2. Context in Information Retrieval

Most existing Information retrieval systems depend, in their retrieval decision, only on queries and documents collections; information about actual users and search context is largely ignored, and consequently great numbers of irrelevant results occur. Towards the optimal retrieval system, the system should exploit as much additional contextual information as possible to improve the retrieval accuracy, whenever this is available.

The context notion can be applied in information retrieval area; this will lead to the *contextual information retrieval* systems which combine a set of technologies and knowledges on the query and the user context, in order to deliver the most appropriate answers to the user's information needs.

In the contextual information retrieval, context has a wide meaning; we can classify some of them in the following groups:

- User Behaviour
  - Visited Web pages (Sugiyama et al., 2004).
  - Recently accessed documents (Bauer et al., 2001).

  - Past queries and click-through data (Shen et al. 2005b).
  - Recent selected items or purchases on proactive information systems (Billsus et al. 2005).
  - Information that are previously processed or accessed by the user via various forms: email, web page, desktop document…etc. Stuff I've Seen SIS, (Dumais et al., 2003).

- Surrounding elements
  - Text surrounding a query, Text highlighted by a user (Finkelstein et al., 2001).
  - Surrounding elements in an XML retrieval application (Hlaoua and Boughanem, 2005), (Sauvagnat and Boughanem, 2004).
  - Broadcast news text for query-less systems (Henzinger et al. 2003).
- Implicit feedback (Kelly and Teevan, 2003).
- Query context
  - Several sources of knowledge involving query to explore the most exact understanding of user's information needs (Allan, 2003), (Conesa et al., 2006).
  - Needs behind the user query and user profile (Daoud et al., 2009).

Thus, contextual information retrieval systems can be classified by: The source of context; how can we extract the contextual information; how the context information is represented and how the context representation is used to adapt the system (Vallet et al., 2007).

2.4.3. Query Context

The notion of query context has been widely mentioned in many studies of information retrieval like (Allan, 2003), (Daoud et al., 2009). The objective is to use a variety of knowledge involving query to explore the most exact understanding of user's information needs. A query context will minimize the distance between the information need, I, and the query q, (Conesa et al., 2006).

Distance (I to q) is minimized by Min (DL, DC, DP) where:

- DL: lack of precision in the language used in the query terms.
- DC: use of the wrong concepts in the query to represent the information need.
- DP: lack of preferences in the query to constrain the concepts requested.

Prior research suggests three techniques for minimizing DC, DL, and DP:

- Lexicons: Comprise the general vocabulary of a language, Can minimize DL by identifying terms with minimal ambiguity.
- Ontology's: Consist of terms, definitions, and axioms relating them, Can minimize DC by helping users understand relationship between concepts.
- User Profiles: Way of stating preferences about a concept. We can minimize DP by serving as a constraint on the range of instances that will be retrieved by the query.

(Daoud et al., 2009) exploit the query context for predicting the user intent as being informational related to the content retrieval, navigational related to the web site retrieval or transactional related to the online service retrieval. Predicting the user intent consists of combining morphologic query characteristics and the session context defined by the user intent held by the associated queries. They construct the query context by associating it with ontology concepts from the ODP (Open Directory Project) taxonomy.

2.4.4. Limitation

The problems to be addressed include 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 difficult to find a contextual information retrieval system that takes into account all the contextual factors (environment, spatial-temporal, personal, task …etc.) at the same time. The considered contextual dimensions may be more or less relevant according to the actual performed research.

Thus the researchers in the context-based information retrieval area often take into account some of these contextual factors (location for example).

In this thesis, we view a user context as a user's current task and its changes over time until accomplish this task at hand. The user context, according our proposition, is the task that the user is undertaking when the information retrieval process occurs. That means we consider from the contextual factors which surround the user during his/her search the *user task*. Therefore, in our approach, we need to model the user task, for this reason we will present in the following section a background research for the task model in the information retrieval area.

2.5. Task Model

Web information retrieval has been studied in the light of request-response for a relatively significant period of time. The user submits a query trying to convey their information need to the Web and in return, they receive a response from the search engine in the form of document hits. But in many occasions, a search activity may necessitate that the user continues interacting with the search engine to achieve a higher-level Web task (Kules, et al., 2008). Research has studied user tasks in order to identify a task framework that would help with understanding user interactions with the Web (Byström and Hansen, 2005). Web tasks have been classified into fact finding, navigation, performing a transaction, and information gathering (Kellar, et al., 2007).

Task models are logical descriptions of the activities to be performed in reaching user's goals. They have shown to be useful for designing, analyzing and evaluating interactive software applications. This section introduces the main concepts underlying task models and discusses how they can be represented and used.

Because the aim of any information system is to be able to answer effectively different search tasks, it is important to understand the nature of these tasks. 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. Because now, most existing interactive systems do not integrate user needs with the characteristics of the relevant task states as the execution of the task progresses.

2.5.1. Task Definition

Each task is represented by several characteristics (features): the identification, the elements and task attributes (Tricot et al., 1998). The elements of the task are:

- The Goal: what is wanted in the execution of the task?
- The initial state: list of objects for representing, at time t_0, a part of the world in which the task will be performed.
- The task body: expression of how the task is executed. It can be either an elementary action or a structure of sub-tasks; the last one is defined by a set of subtasks that can be sequential, alternative, parallel or simultaneous.
- Post conditions: constraints on the objects in the final state.
- The final state: list of objects which represent the part of the world that has been modified by the task.

Finally, the task attributes are the particular characteristics of certain sub-tasks (optional, iterative, priority, interruptible).

The task can be either long-term task (trip for some days) or short-term task (shopping), as shown in Figure 2.6:

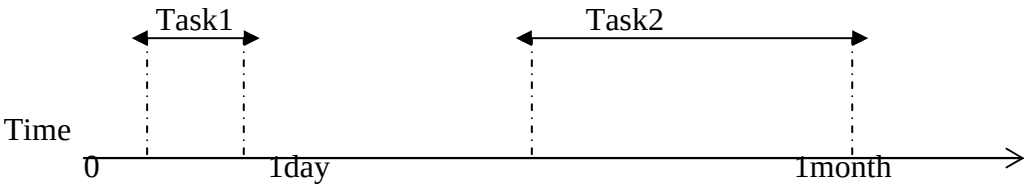


Figure 2.6. Task types.

We need flowchart to describe the progression of some task from beginning to end and to detect the transition between the task states interval with the time changes. For that, we can use a UML activity diagram or contextual graph, as we will present in the following sections.

In fact, there is complex interaction between all task components. In Figure 2.6 we consider task goals, the task process proper, information acquisition, information used in the task, and information systems.

Very schematically, Figure 2.7 shows that the components affect each other. Whenever one changes, the change requires and/or causes changes in the other. For instance, information systems may affect the information that is available for the task. In some cases this may allow setting more demanding goals for the task and changes to the task process proper. This may bounce back new requirements on information systems: there are constant repercussions. In fact, in modern times of rapid technological change, a dynamic imbalance dominates the scene with only relatively short periods of relatively stable practices. Information seeking and retrieval is intimately connected to task performance indeed. The changes may be classified from simple to very pervasive, the simple ones being just the change of implementation like replacing pen and paper by a pocket calculator without touching anything else, and complex ones such as changes in the ultimate goals of work whereby also the process may totally change, information seeking processes included. Technology may be the cause of such changes but also the recipient of new requirements, (Järvelin and Ingwersen, 2004).

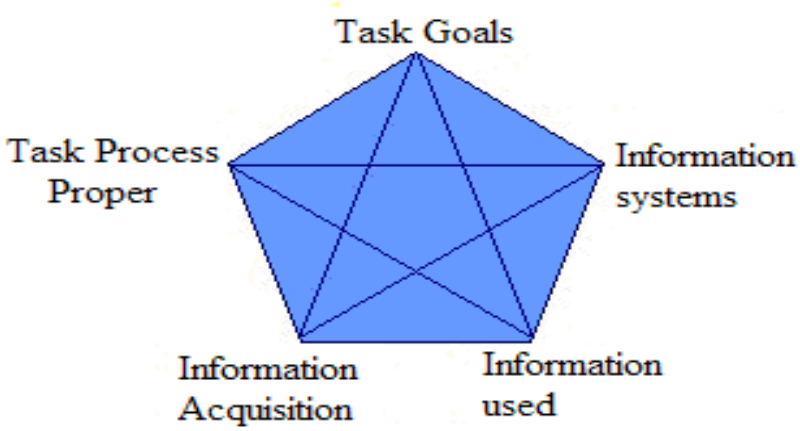


Figure 2.7. The interaction of task components (Järvelin and Ingwersen, 2004).

2.5.2. The different types of task scenarios

For the different types of the task, we can use two principal scenarios; outside scenario (for example user walks around the city, looking for interesting places/buildings to visit/look at, this is the case of the mobile user) and inside scenario (for example the user tries to organize a workshop).

In the *outside scenario*, we can take the example of shopping assistant; let suppose the user is at the shopping center trying to figure out what he needs to finish his shopping, a shopping assistant, which is applied in PDA, can:

- Tell the user what parts he needs.
- Where to find them relative to his location in the store.
- What is on sale?
- Do comparative pricing.
- Use his previous profile information to customize shopping and delivery.

In the *inside scenario,* we can take the example of manage the process of selecting articles of a scientific conference electronically.

2.5.3. Task Model Approaches

The task modeling consists of describing of an optimal procedure to achieve the goal, a sequence of actions or operations in a given environment.

Task models are explicit representations of user tasks that can help support certain rigorous forms of task analysis. They are recognized as useful constructs that can be exploited throughout the Software Development Life Cycle (SDLC) (Balbo et al., 2002), either on their own or as components of full interface design models. For example, designers and implementers of interactive systems can use them to assess the complexity of the tasks that the user is expected to perform, to anticipate the amount of time required to perform the tasks, to simulate the behavior of the resulting system, to automatically build interfaces based on the model, and to generate user-oriented instructions, (Paris et al., 2004). In addition, task models can be used to facilitate communication between the various stake-holders in the design process (e.g. Balbo et al., 2002). Though there are a variety of task modeling languages, they can all support each of these goals to some degree.

Watson's "Just-in-time" information retrieval system (Leake, et al. 1999) 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.

According to (Terai et al., 2008) 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 (Freund et al., 2005) proves that the nature of the task has an impact on decisions of relevance and usefulness.

In the approach (Luxenburger et al., 2008) 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 (W. White and Kelly, 2006) describes a study on the effect on retrieval performance of using additional information about the user and their search tasks when developing IRF algorithms (Implicit Relevance Feedback).

Some approaches try to detect the user task automatically, by capturing content from Internet Explorer and Microsoft Word applications (Leake et al. 1999), or by using a relevance feedback in the Web search system, (TaskSieve) (Jae-wook Ahn et al., 2008). Other approaches combine semantic technologies with machine learning in order to detect the user task (S. Rath et al., 2010).

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.

2.5.4. Task Representation

2.5.4.1. Contextual graph

For the last few years, the use of context has grown more and more widespread in certain areas. This tool now plays a fundamental part in the study of human behavior. Recently an Artificial Intelligence approach has emerged: contextual graphs. These

have many applications in various areas: psychology, accident management, medical diagnostic, etc. In order to fulfill this need to integrate context in knowledge management.

A contextual graph is a tool or a formalism that gives a consistent representation of thinking processes and contextual elements. Thus a contextual graph consists of the representation of the context of the execution (and not only a description of the task). A contextual graph represents the different ways to solve a problem; it is an oriented directed graph, with only one input and one output and a general structure of spindle (Brezillon, 2005).

For making context explicit in order to use it, contextual graphs are used to capture the effective behaviors of users in an activity of information retrieval on a scientific website.

Contextual graphs represent:

- A temporal sequence of diagnosis and actions.
- The different ways to reach a goal.
- The elements for choosing the right action sequence.

Components of the contextual graph:

The elements of a contextual graph are actions, contextual elements, sub-graphs, activities and parallel action groupings, (Figure 2.8).

Action is an elementary task. A contextual element is a pair of nodes, namely a contextual node (1, N) where N is the number of instances of the contextual element. A sub-graph is itself a contextual graph CG, and the activity is a particular type of sub-graph identified by human actors. The temporal branching expresses the fact that several groups of actions must be accomplished, but the order in which action groups must be considered is not important, or even could be done in parallel, however all actions must be accomplished before continuing. This is a kind of complex contextual element, in the same way that an activity is a kind of a complex action, (Brezillon, 2005).

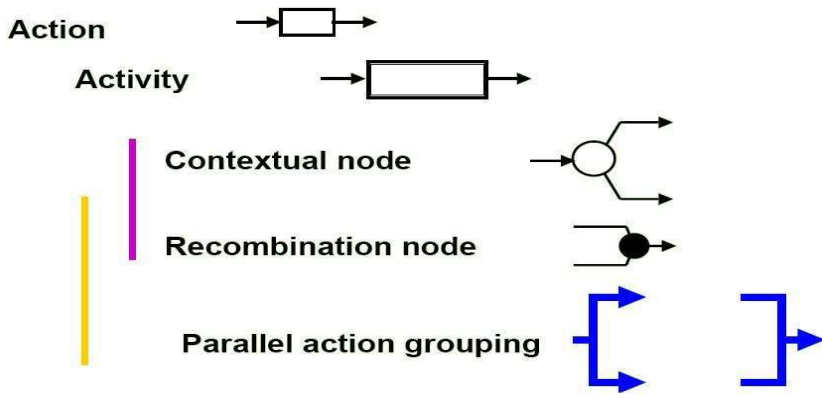


Figure 2.8. Contextual Graph Components.

A path in a contextual graph corresponds to a specific way for the problem solving. It is composed of contexts. A contextual graph is an acyclic graph because the user's tasks are generally in ordered sequences. Figure 2.9 presents a contextual graph for an example of the task “organizing a visit to a city”.

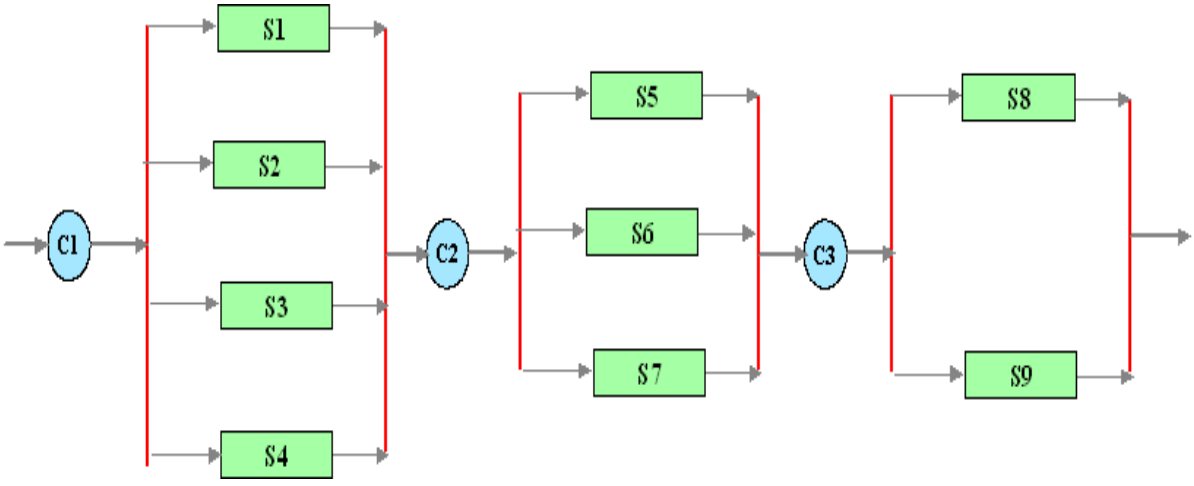


Figure 2.9. A contextual graph that represents the task “*organizing a visit to a city*”.

Where: C1, C2, C3: contextual nodes (before the visit, during the visit, after the visit).

The rectangles represent the actions. The two red bars that frame actions are a group of parallel actions. This means that actions between these two bars may be executed in any order or in parallel.

2.5.4.2. UML Activity Diagram

Activity diagrams are graphical representations of workflow behavior of a system. It shows the flow of activities through the system. Activity diagrams are similar to state diagrams because activities are the state of doing something. The diagrams describe the state of activities by showing the sequence of activities performed. Activity diagrams can show activities that are conditional or parallel (Rumbaugh et al., 1999).

A task is a work package that may include one or more activities needed to perform this task. An activity is the action actually performed while a task is the purpose which is prescribed. 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 task.

For instance, we can take the task "*organize a Trip to Paris*". This task will be presented by the UML activity diagram as shown in Figure 2.10. It has several task states or activities needed to perform this task such as: book a flight, book a hotel, search for tourist information, etc. the task state is a stage needed to accomplish the current task. Figure 2.10 shows the UML activity diagram which contains the activities needed to perform the current task:

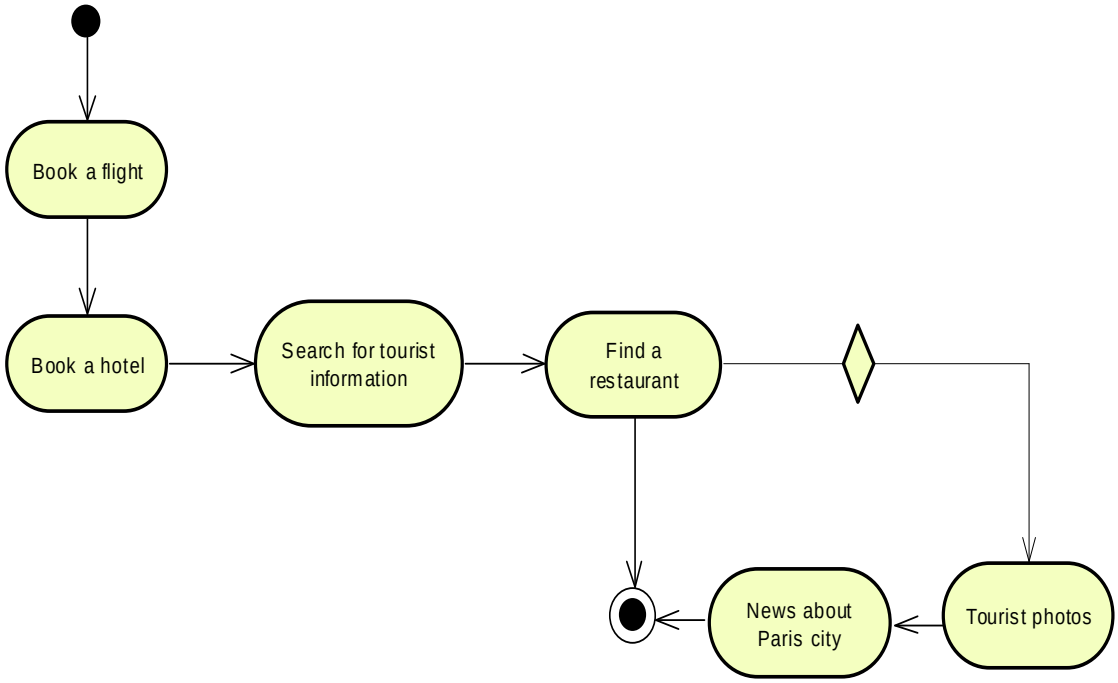


Figure 2.10. UML Activity Diagram for the task "*organize a Trip to Paris*".

2.6. Query Reformulation

It is not always easy for users to formulate effective queries to search engines. One reason for this is the ambiguity that arises in many terms of a language. Queries having ambiguous terms may retrieve documents which are not what users are searching for. On the other hand, users typically submit short queries to the search engine, and short queries are more likely to be ambiguous. From a study of the log of a popular search engine, (Jansen et al, 1998) conclude that most queries are short (around 2 terms per query) and imprecise.

In order to overcome these problems, methods to reformulate user queries are suggested. Their aim is to help the users to specify alternative related queries in their search process. In this thesis we will present an algorithm to suggest related queries to a query submitted to a search engine.

(Efthimiadis, 1996) identifies two query formulation stages: the initial query formulation stage in which the search strategy is constructed and the query reformulation stage in which the initial query is adjusted manually or with the assistance of a system.

It is often argued that query reformulation is not any easier than initial query formulation given that information retrieval (IR) systems provide very little assistance.

Several studies have investigated patterns of query reformulation on the Web. We can classify it in the following groups:

2.6.1. Query Reformulation Systems based on user profile

A profile is a user model which specifies the user domain of interest and the most general preferences that distinguish this user from the others. All queries issued by the same user are evaluated with respect to his specific profile. The same query issued by different users may have different results as it is evaluated using different profiles.

Two main approaches based on the user profile to reformulate a query have been proposed: query enrichment process which consists in integrating elements of the user

profile into the user's query and the second approach based on a user profile is the query rewriting process which translates the query to access the real data sources.

- *The query enrichment* process consists in integrating elements of the user profile into the user's query. The user profile is defined as a list of disjunctive predicates, including selections and joints. Given such a profile, the query enrichment process consists in reformulating the initial user query by adding predicates from this profile. The first step of query enrichment consists in selecting the top K profile predicates which will be used to enrich the user query. In order to be selected, each predicate has to be related to the user query and not to conflict with it. The second step of query enrichment consists in integrating the top K profile predicates to the query. Two strategies can be followed to do this: the generation of a single query or the generation of multiple queries. (Koutrika et al., 2004)
- *The query rewriting* process consists in transforming the user query expressed on the virtual schema so that it can be evaluated on data sources. It aims to determine contributive data sources for query execution and to use their definitions to reformulate the query. The query rewriting process translates the query to access the real data sources (Vidal et al., 2006).

The limitation of these approaches is that they do not take into consideration the user context for activation the elements from the user profile.

2.6.2. Queries Reformulation by Relevance feedback

Rocchio's method (Rocchio, 1971) is a classic algorithm for relevance feedback. Queries reformulation by injecting relevance feedback is an interactive process, led by the user with the objective of generating a new query more appropriate than that originally expressed by the user. Its fundamental principle is to use the initial query in order to begin the search and then modify it from judgments of relevance and/or no relevance by the user. The new complaint obtained in each iteration feedback, can rectify the direction of the search in the meaning of the relevant documents within the meaning expressed explicitly by the user (Baeza-Yates et al., 2004). Indeed, the retrieval becomes effective after a high number of iterations feedbacks, but that will cause to the user an overload. Thus because relevance feedback requires the user to

select which documents are relevant (relevance judgment), it is quite common to use pseudo-relevance feedback and therefore lack reliability.

2.6.3. Query Disambiguation

The disambiguation techniques 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 (Wakaki et al., 2006). These techniques are usually based on an explicit user or exploitation of resources such as thesaurus and ontology's.

But this disambiguation may cause the query to move in a direction away from the user's intention. For example the query "windows" might be about actual windows in houses or the Microsoft Windows operating system. A system might choose an interpretation different from the user's intention and augment the query with terms related to the wrong interpretation.

2.6.4. Query Expansion using external resources of terms

Query expansion is the process of adding terms to the original query in order to improve results by including terms that would lead to retrieving more relevant documents. However, in a more general sense, it also refers to methods of query reformulation, i.e., any kind of transformation applied to a query to facilitate a more effective retrieval. In this group of approaches the initial query is expanded by using external resources of terms, such as thesauri or ontology, that contain the vocabulary used in the query enrichment.

Many approaches like (Storey et al., 2004) try to reformulate the web queries based on a semantic knowledge about different application domains from Research-Cyc for example, others use sense information (WordNet in general) to expand the query (Navigli et al., 2003).

Many approaches, for example (Bhogal et al., 2007), expand the user initial query 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 not related to query

terms. More precisely they are related to the query but only under a particular context of the specific query.

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.

This thesis presents a new approach for improving user query processing. We propose a hybrid query expansion method that automatically generates query expansion terms from the user profile and the user context.

2.7. Agents (Intelligent Assistant)

2.7.1. Definition

- Agents are software programs that implement user delegation. They can accomplish complex tasks by dividing into sub-problems and they can adapt behavior to changes. Each mental agent can only do small process, joining these agents in society's leads to true intelligence (Society of agents).
- Agent is a personal assistant who is collaborating with the user in the same work environment; Information filtering is one of the many applications an agent can assist (Maes, 1994).
- One of the applications that the agent can do is reformulation a user query in order to assist the information retrieval. Whereas a software agent can provide active and beneficial assistance to a user for query reformulation during a search session, (Jansen, 1999).

In this thesis, we propose an intelligent assistant that can generate new reformulated queries in order to guide the information retrieval system to provide context-based personalized results.

2.7.2. Using an agent or an assistant in Information Retrieval

In information retrieval systems, several authors, like (Jansen, 2005) propose to implement an assistant that could be a mediator between the user and the search site. Because of performance and confidentiality issues, this assistant should run on the user's workstation.

The researchers, in the information retrieval area, use the intelligent agent to be efficient without increasing the user workload, whereas the agent must automatically understand the aims of a query from the few words that compose it. It must be able to clear up any term ambiguity and to filter the results according to criteria that are specific to the user. In order to do this, it must build a model of the user and maintain it automatically. Furthermore, as a user may have several interests and as he may switch from one to another at any time, the assistant must be able to track the active center of interest in the on-going work.

Few systems however use contextual information not directly linked to the current search (i.e., information other than a direct or indirect feedback on documents retrieved by the query) to help the search. Remembrance Agent (Bradley et al., 1996) uses an analysis of an e-mail being created to retrieve archived mails that are similar and present them to its user. (Rhodes et al., 2000) proposed Just-In-Time Retrieval Agents, generalizing the concepts of the remembrance agent to other contexts.

(Subercaze et al., 2007) define two types of agent: *Context Agent*, responsible for contextual knowledge capture and dissemination, and the *Personal Agent* which possesses knowledge about its own user (the user's profile) and which is in charge of delivering recommendations to the user. The personal agent is implemented on a mobile device (cell phone, PDA) and it plays the role of a personal assistant. The personal agent gathers context information delivered by context agents and provides a service to the user, (Subercaze et al., 2007).

In our approach we consider a personal agent which uses a hierarchical model of the user's interests (user profile) and uses information gathered while observing the ongoing user's activities (user context) in order to interpret the queries and reformulate them to assist the information retrieval and filter the results returned by the Web search engines.

Chapter 3

State Reformulated Query (SRQ) Model

3.1 Introduction

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 an ontological 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 related terms to the current search session and 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.

As we mentioned previously, the queries of mobile users are often short, and their information needs are often related to contextual factors to perform task at hand, therefore our system is more useful in providing relevant results for mobile users. Here we will describe our approach which contains three models: Task model, user profile model and SRQ model which is used to generate State Reformulated Queries.

Our system has the following characteristics:

- It builds a user profile, from the analysis of documents or files collection that are managed by this user.
- During the performance of one task at hand, the user submits a query related to this task to the information retrieval system. Our system identifies a task context from the predefined tasks (task ontology) based on the initial user query and its linguistic and semantic knowledge.
- The user's task is represented by UML activity diagram which contains all task states and the transition between these states.
- To combine the user profile and the task context, our system proposes a context-based hybrid method for user query expansion. For an initial user query, relevant terms are proposed to reformulate this query, 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 don't contain stop words, duplicated terms and out of context terms. (query refinement)

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).

Figure 3.1 presents the main components of the system. Three main parts may be identified: user profile, user context, and context manager.

Exploiting ***user profile*** involves using information contained in profile in order to adapt the retrieved results to this user. In our system exploiting user profile is carried out through three parts, each with a specific role:

- The documents observer is responsible for indexing and handling the user's documents, which exist in one library on the user machine, and then tracking its evolutions.
- The ontological profile is a semantic hierarchical structure of the user profile. It organizes the user information in categories using ontology (like Open Directory Project ODP taxonomy).

- The operational profile is derived from the ontological profile, as a list of related relevant terms that can be easily used in the search process.

The ***task model*** is responsible for defining the current working context by assigning one task to the initial query from the predefined tasks.

The ***context manager*** 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.

The ***query manager*** uses the context to interpret a query and adapt results to the user (query refinement).

The several models will be described in the following sections.

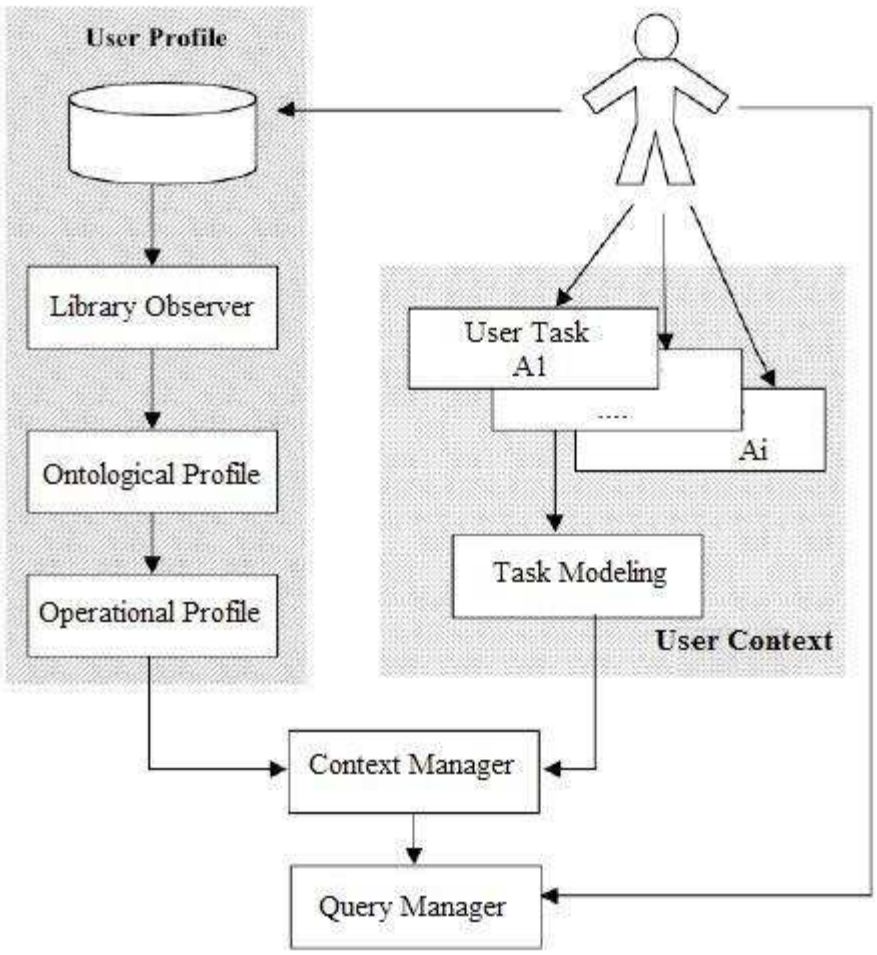


Figure 3.1. Main Components of our System.

3.2 Definitions

In this section, we will provide some definitions that we will use in our approach.

3.2.1 Cosine Similarity

Cosine similarity is a measure of similarity between two vectors of *n* dimensions by finding the cosine of the angle between them, often used to compare documents in text mining. Given two vectors of attributes, A and B, empirically, cosine similarity can be expressed as follows, (Garcia, 2006).

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}.$$

In the case of information retrieval, the cosine similarity of two documents will range from 0 to 1, since the term frequencies (*tf. idf* weights) cannot be negative. The angle between two term frequency vectors cannot be greater than 90°. As the angle between the vectors shortens the cosine angle approaches 1, meaning that the two vectors are getting closer, meaning that the similarity of whatever is represented by the vectors increases.

To do this we need to construct a term space. The term space is defined by a list of terms (index). These terms are extracted from the collection of documents to be queried. The coordinates of the points representing documents and queries are defined according to the weighting used scheme (Garcia, 2006).

In *our approach*, as we will mention later, we will represent the query and the predefined tasks by terms vectors. Thus we construct an index of terms that consists of:

- Terms of the predefined tasks.
- Terms of the predefined subtasks (or the states of each main task) including the task state attributes. Because we will see that each main task consists of several states and one attribute at least for each state.
- Terms of related-task concepts from ODP (Open Directory Project).

Thus this index consists of *r* terms. We will use this index when using the term vector model.

3.2.2 Kullback-Leibler Divergence (KLD)

Given two probability distributions P_i and P_j of a random variable, the similarity between P_i and P_j can be defined by the non-symmetric measure Kullback-Leibler divergence as follows (Kullback et al., 1987):

$$KLD \quad (P_i \| P_j) = \sum_i P_i \cdot \log \left(\frac{P_i}{P_j} \right)$$

3.3 General Language Model

Here, we construct a new general language model for query expansion terms including the contextual factors and user profile in order to estimates the parameters in the model that is relevant to information retrieval systems.

The main idea of language models in IR is to order each document *D* in the collection C according to their ability to generate the query q. Thus, it is estimate the generation probability $P(q| D)$ (Bouchard and Nie, 2006).

For a query $q = t_1 t_2 \ldots t_n$, the generation probability is estimated as follows:

$$P(q \mid D) = \prod_{t \in q} P(t \mid \theta_D)^{c(t;q)} \qquad (1)$$

Where:

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

θ_D is a language model created for a document D.

$P(t|\theta_D)$: The probability of term *t* in the document model.

In the language modeling framework the similarity between a document D and a query q (a typical score function) can be also defined by measuring the Kullback-Leibler (KL-divergence) (Lafferty et al., 2001) as follows:

$$Score\ (q, D) = -KL\ (\theta_q \| \theta_D) = \sum_{t \in V} P(t \mid \theta_q) \log \frac{P(t \mid \theta_D)}{P(t \mid \theta_q)}$$

$$= \sum_{t \in V} P(t \mid \theta_q) \log P(t \mid \theta_D) - \sum_{t \in V} P(t \mid \theta_q) \log P(t \mid \theta_q)$$

$$\propto \sum_{t \in V} P(t \mid \theta_q) \log P(t \mid \theta_D) \qquad (2)$$

Where: θ_q a language model for the query q, generally estimated by relative frequency of keywords in the query, and V the vocabulary.

$P(t|\theta_q)$: The probability of term *t* in the query model.

Note that the last simplification is done because $\sum_{t \in V} P(t \mid \theta_q) \log P(t \mid \theta_q)$ depends only on the query, and does not affect the documents ranking.

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 formula (2) becomes:

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

Where the score according to each component model is:

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

The remaining problem is to construct contextual model and user profile model and to combine all the models.

3.4 User Context Modeling

In this section we will study how we can extract the contextual information, how they are represented and how the context representation is used in our system of context-based query reformulation, in other words, we will present the user context modeling for our system. It is very difficult to take into account all the contextual factors during the information retrieval process (see section 2.4.4). Thus we 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.

To learn user's task, we exploit both a semantic knowledge (*ODP*) and a linguistic knowledge (*WordNet*) as we will mention in the next section.

3.4.1 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, (see section 2.4.1).

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 a set of predefined tasks A_1, A_2,…, A_i, and each one needs a set of predefined activities to perform this task. The activities of each predefined task are represented by using UML activity diagram in order to provide the succession between the activities. The objective of the task model is to assign one task A_* to the user query from the set of predefined tasks and then provide the UML activity diagram for this current task.

To implement our proposed task model, we exploit study questionnaires (W. White and Kelly, 2006) which were used to elicit tasks that were expected to be of interest to subjects during the study. In that study (W. White and Kelly, 2006), 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 generate 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 and the succession between these sequences. 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 selects one task from the proposed predefined tasks and assigns the selected task 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. Here, we will show that it is possible to determine the user's task automatically in order to facilitate the process to users. For applying this method, we will conceive an algorithm to assign automatically one task to the user query by taking advantages of existing linguistic resources (*WordNet*) and semantic resources (*ODP*) as shown in Figure 3.2. We will explain this algorithm in the following:

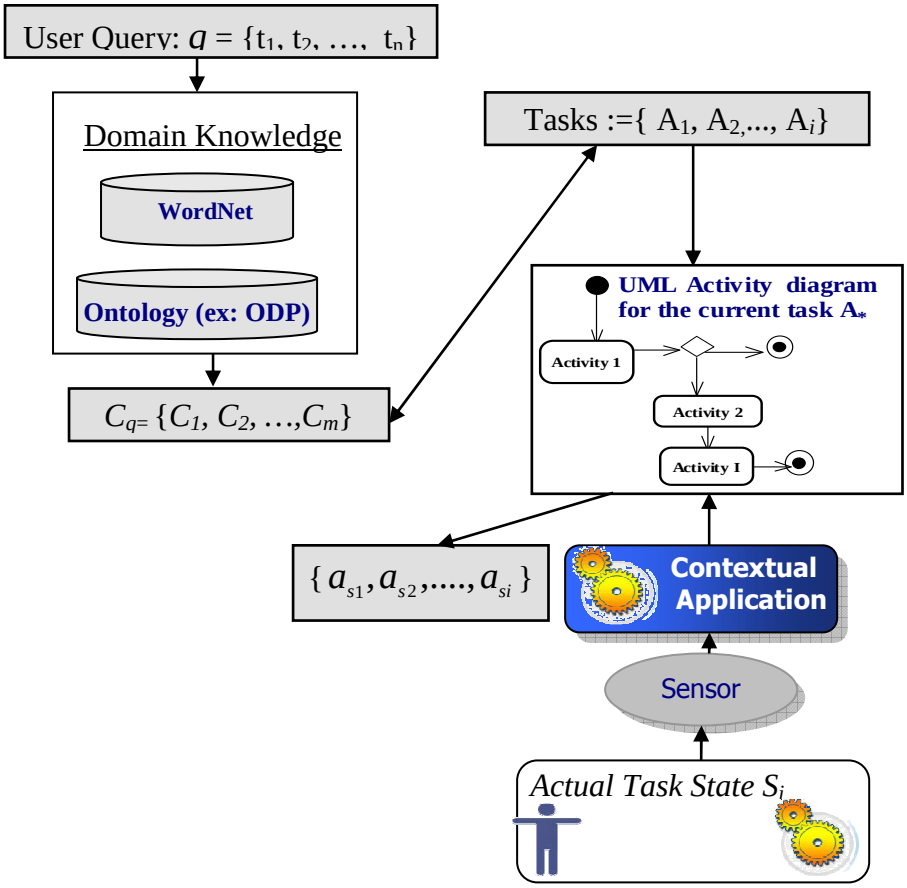


Figure 3.2. Task Model.

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

- Terms of the main predefined 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, because each main task consists of several activities, each one expresses a 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*" and the related terms from the user profile (such as vegetarian, Italian, …etc.) will be assigned to this state in order to personalize the query.

- Terms which represent the related-task concepts from ODP (Open Directory Project) $\{t_1, t_2, \ldots, t_k\}$. They are identified by querying the state attributes of the predefined tasks on the ODP taxonomy.

This index consists of r terms. Table 3.1 shows an example of this task terms index. We will use this index when using the vector space model.

Table 3.1. Index of task terms

Term_Id	Term	*tf*	Occurrence (postings)
1	*News*	2	A_2:1 A_9:1
2	*weather*	2	A_2:1 A_9:1
3	*Shopping*	1	A_3:1
4	*Restaurant*	2	A_4:1 A_9:1
….	….	….	….
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 $\vec{A}_*$. That means each predefined task is represented by a term vector.

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:

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

Where: A is a set of 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 the predefined tasks in which term a_{si} occurs in the documents that represent them.

$tf_{a_{si}}$: is the frequency of term a_{si} in the task $A_* \in A$ or number of times a term a_{si} occurs in a document that represents a task A_*.

Table 3.2 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 3.2. Example of calculating term's weights Wa_{si}.

	Counts TFa_{si}							Weights, $Wa_{si} = TFa_{si} * IDFa_{si}$			
Terms	$\mathbf{A_1}$	$\mathbf{A_2}$	**…**	$\mathbf{A_9}$	n_{asi}	$\|A\|/n_{asi}$	$IDFa_{si}$	$\mathbf{A_1}$	$\mathbf{A_2}$	**…**	$\mathbf{A_9}$
News	0	1		1	2	9/2	0.653	0	0.653		0.653
Weather	0	1		1	2	9/2	0.653	0	0.653		0.653
Tidings	0	1		0	1	9/1	0.954	0	0.954		0
Program	0	1		1	2	9/2	0.653	0	0.653		0.653
information	1	1		1	3	9/3	0.477	0.477	0.477		0.477
temperature	0	1		0	1	9/1	0.954	0	0.954		0
atmospheric	0	1		0	1	9/1	0.954	0	0.954		0
Meteorological	0	1		0	1	9/1	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 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 does not capture the semantic relationships between terms and semantic knowledge does not 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 like (Allan, 2003). 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. The concepts of the terms set q_w and their sub-concepts produce the query-context $C_q = \{c_1, c_2, ..., c_m\}$ which is represented as a term vector $\vec{C}_q$. Thus the elements of C_q are the concepts extracted from the ODP taxonomy by querying the initial query and its synonyms against it.

Next, to find out which task vector ($\vec{A}$) is closer to the query-context vector $\vec{C}_q$, we use the similarity analysis introduced in Section 3.2.1. 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 $q\ \langle t_1, t_2, ..., t_n \rangle$ 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}.

Figure 3.3 illustrates the comparison between the different vectors which represent the query context $\vec{C}_q$ and the predefined tasks: $\vec{A}_1, \vec{A}_2, \ldots, \vec{A}_9$.

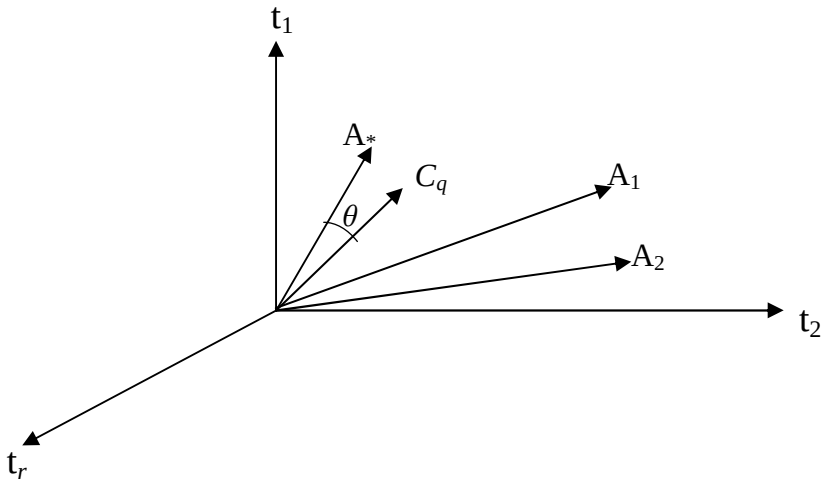


Figure 3.3. Representation of the tasks and the query as term vectors.

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 3.2).

For example, let's take the user query q= {*weather*}. We take again the table 3.2 and we determine the term counts TF_i for the query context C_q and their term's weights. That is shown in Table 3.3.

Table 3.3. Example of calculating term's weights for the query and each task.

		Counts TFa_{si}							Weights, $Wa_{si=}$ TFa_{si}∗$IDFa_{si}$				
Terms	C_q	A_1	A_2	…	A_9	n_{asi}	$\|A\|/n_{asi}$	$IDFa_{si}$	C_q	A_1	A_2	…	A_9
News	0	0	1		1	2	9/2	0.653	0	0	0.653		0.653
Weather	1	0	1		1	2	9/2	0.653	0.653	0	0.653		0.653
Tidings	0	0	1		0	1	9/1	0.954	0	0	0.954		0
Program	0	0	1		1	2	9/2	0.653	0	0	0.653		0.653
information	0	1	1		1	3	9/3	0.477	0	0.477	0.477		0.477
temperature	1	0	1		0	1	9/1	0.954	0.954	0	0.954		0
atmospheric	1	0	1		0	1	9/1	0.954	0.954	0	0.954		0
Meteorolo-gical	1	0	1		0	1	9/1	0.954	0.954	0	0.954		0
…													
r													

To find out which task vector is closer to the query vector, we calculate the cosine similarity which is previously described in Section 3.2.1. 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:

$$|A_2| = \sqrt{(0.653)^2 + (0.653)^2 + (0.954)^2 + (0.653)^2 + (0.477)^2 + (0.954)^2 + (0.954)^2 + (0.954)^2}$$
$$= 2.269$$

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

$$\left|C_q\right| = \sqrt{(0.653)^2 + (0.954)^2 + (0.954)^2 + (0.954)^2} = 1.777$$

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

$$C_q \bullet \mathrm{A}_2 = 0.653 * 0.653 + 0.954 * 0.954 + 0.954 * 0.954 + 0.954 * 0.954 = 3.157$$

Now we calculate the similarity values:

$$\mathrm{Cosine}\theta_{A2} = \frac{C_q \bullet \mathrm{A}_2}{\left|C_q\right| * \left|\mathrm{A}_2\right|} = \frac{3.157}{1.777 * 2.269} = 0.783$$

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 3.4:

Table 3.4. 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≥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.

3.4.2 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:

A: You are at the shopping center trying to figure out what you need.

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. 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*) 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 (Figure 3.4). There is at least one attribute as_i for each state S_i of the current task which represents by the UML activity diagram. For instance, for the task "*Travel*" (discussed in the previous section) we can design an

UML activity diagram for the user u_j that contains all activities as shown in Figure 3.4.

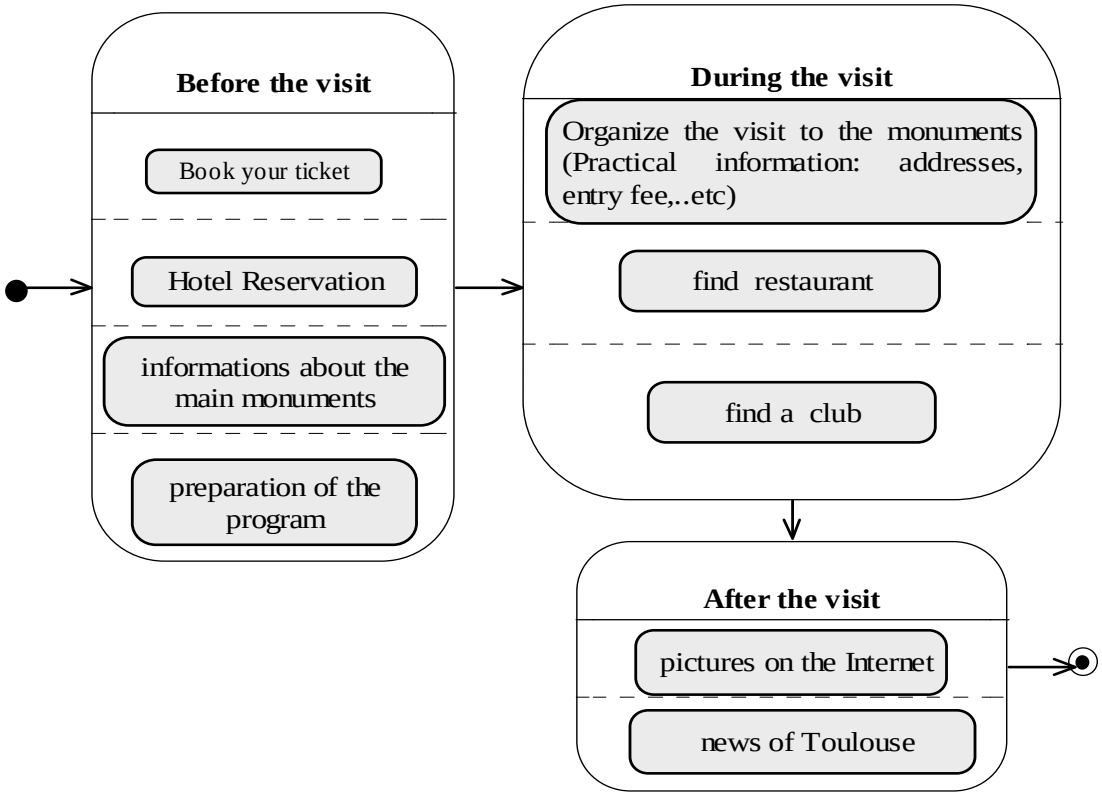


Figure 3.4. Example of a task "*travel*" which is represented by UML activity diagram.

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 (see Figure 3.2); 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 predefined task. After the user's query is submitted to our platform, the related task is assigned automatically to the user query according to the previous method. In this time the system can define the UML diagram related to this user that contains all task states. If the user profile hasn't the UML diagram for the current task, then the system will use a predefined UML diagram related to this current task. 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. Figure 3.5 shows the query session over times.

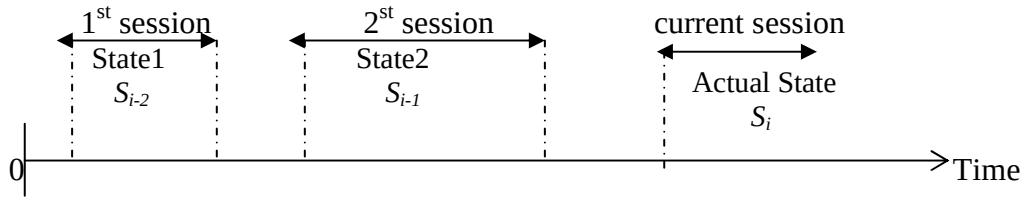


Figure 3.5. Query sessions for a current task.

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 us take the same example in Figure 3.4, if the user u_j is in the activity: "*find restaurant*", and if the previous query session was about an "*organization a visit to Toulouse"* then the current query session will be about the restaurants of 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 "*clubs in Toulouse*" which is the next activity in his/her UML diagram shown in Figure 3.4.

In the UML diagram, when the next task state has two probabilities, that means the user is in an activity and when he completes this activity he has two possibilities, thus that will require a feedback from the user, for instance in the UML diagram shown in Figure 3.4, if the user is in the context "*after the visit*", thus there are two probabilities for the user u_j, either "*watch the news*" or "*view photos of Toulouse*". This will depend on the user feedback.

In the following section we will present our approach to create a user profile and then extract task-related attributes from this profile and finally add these attributes to the query terms and the terms which are extracted from the task state attributes. All these terms, that represent the contextual information, are combined to create the context description.

3.5 User Profile Modeling

The life of the user profile modeling involves three phases (see Figure 3.1). In the first phase, user documents are indexed and represented to construct an ontological user profile (*library observer*). Next the indexed documents are used to create taxonomy which contains a collection of concepts that are ordered in a hierarchical way and inferred from the ODP (Open Directory Project) (*ontological profile*). This taxonomy is finally used to build an *operational profile* which is a list of relevant terms related to the query context that can be easily used by the other models.

3.5.1 Phases of the User profile Representation

As we mentioned previously, we propose three phases to construct the user profile from his/her library (documents or files collection), each phase has a specific role, (see Figure 3.1).

3.5.1.1 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 (Figure 3.6), 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.

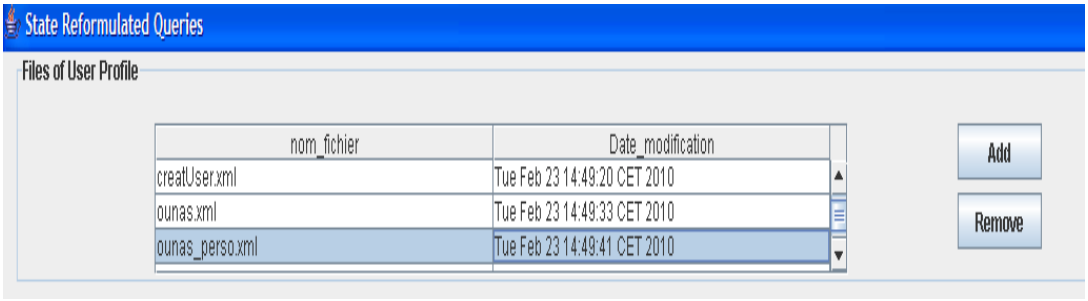


Figure 3.6. Interface of indexed user documents.

3.5.1.2 The Ontological Profile

We use ODP taxonomy as a basis for concepts-based part of our system (see Section 2.2.1). As the dataset of *ODP* (*Open Directory Project*) 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 indexed user documents (the library observer).

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 using ODP taxonomy. 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 (Sieg et al., 2007 c), 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 Figure 3.7. 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 Figure 3.7, 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 as shown in Figure 3.7. 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* | .

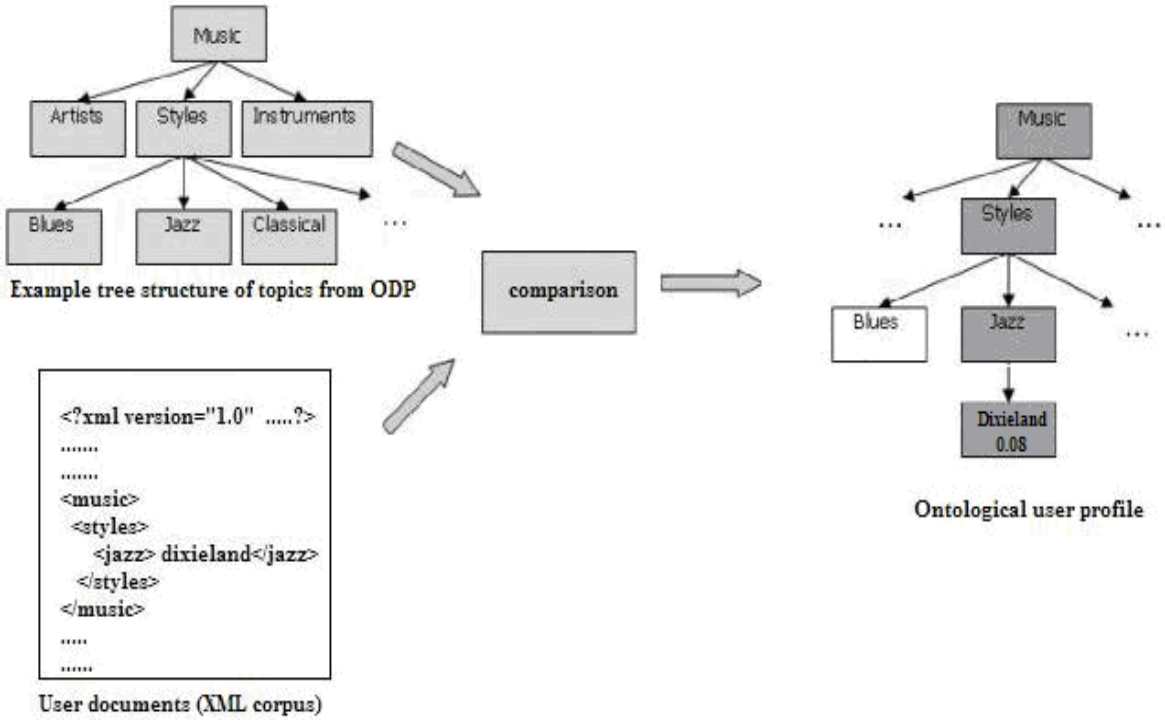


Figure 3.7. Inferring the ontological profile from user documents and ODP.

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 Figure 3.7.

Thus, the ontological profile for each user consists of a list of concepts and their current weighted values. In this way, the profile will adapt to changing user interests as the trial progresses.

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. Once an ontological user profile is constructed, the query context C*q* can be used to activate concepts that will form the operational profile.

3.5.1.3 The Operational Profile

The operational profile is derived from the ontological profile, as a list of related relevant terms that can be easily used in the search process.

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*, in Section 3.4). This allows to activate for each query-context concept its semantically related concepts from the ontological user profile, following our algorithm, depending on the relevance propagation (Asfari et al., 2008), that 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.

3.5.2 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 depended on the relevance propagation (Asfari et al., 2008) as shown in Figure 3.8.

The execution 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 $C_{q=} <C_1, C_2, \ldots,.C_i>$ to be answered by the algorithm.

Output: Res_u: Vector of sorted context-related user u'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:

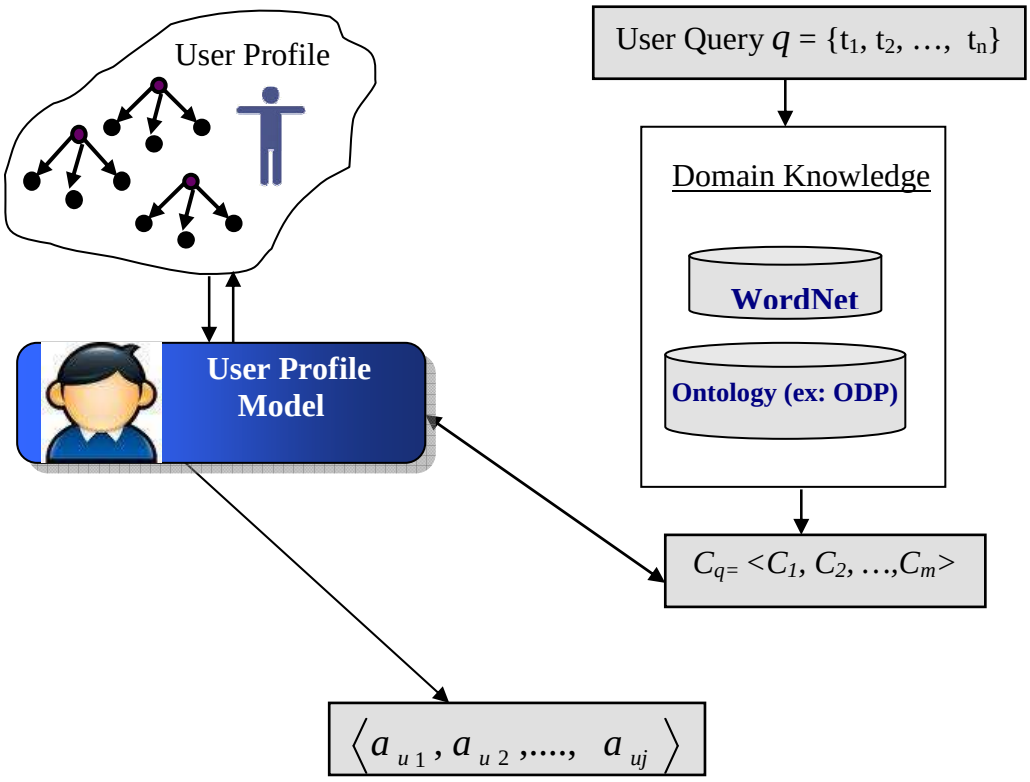


Figure 3.8. User Profile Model.

3.5.2.1 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, so to represent our ontological profile hierarchy, we can imagine the sub graph shown in Figure 3.9.

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 Figure 3.7, so we can imagine the representation of the user profile that is shown in Figure 3.9, this graph contains the concepts, the leaf node in this graph is annotated by values and interest scores for this values.

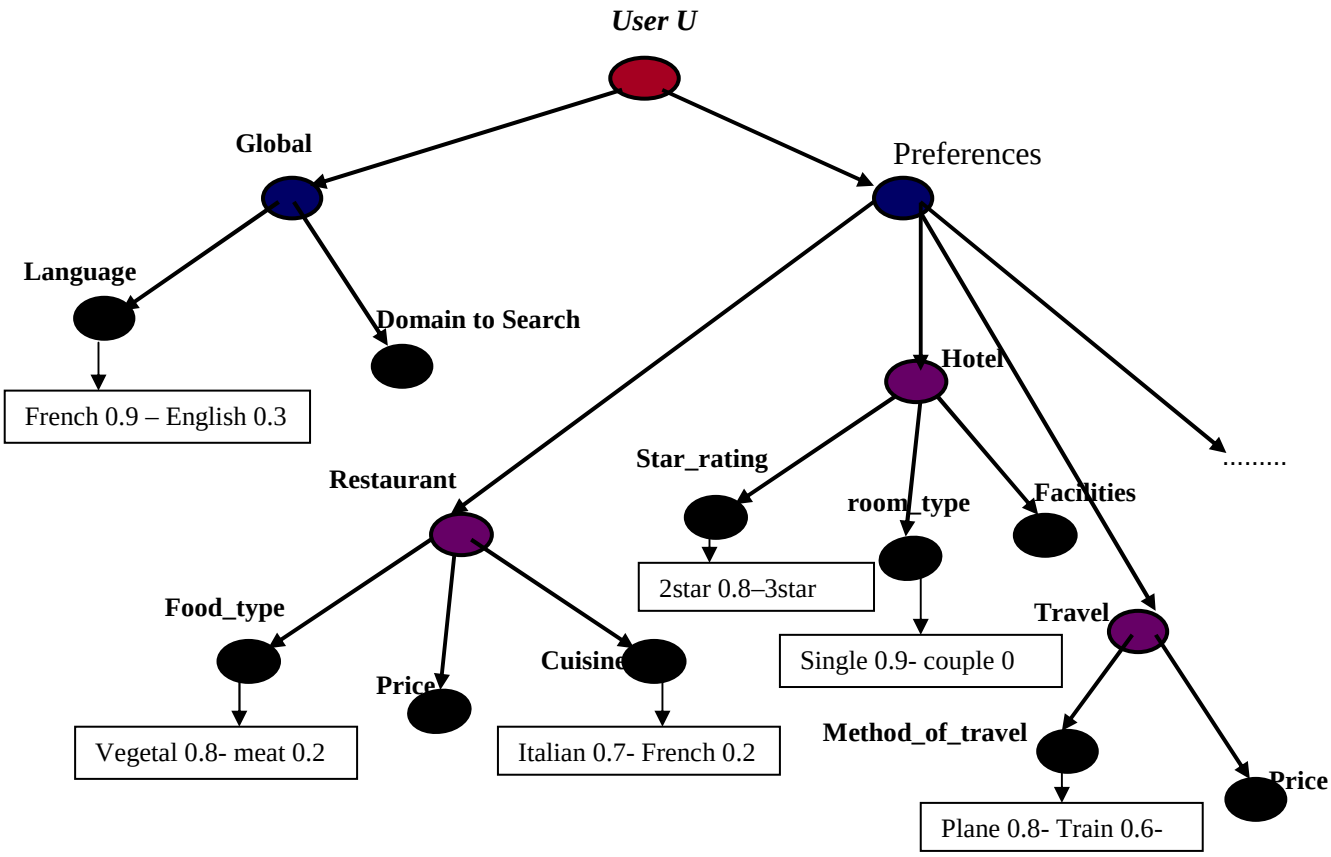


Figure 3.9. Graph of user profile's concepts $Prof_u$.

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:

Relevance Propagation method 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.

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

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:

$$Propagation_i(Prof_u[j],\ Prof_u[n]) = Weight\ (C_q[i], Prof_u[j]) * \frac{1}{Max(Dist(Prof_U[j],\ Prof_u[n]) + 1)}$$

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).

$Dist(Prof_u[j], Prof_u[n])$: 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 $C_{q=} <C_1, C_2, \ldots,.C_i>$ 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:

$$N = Relevance\ (Prof_u[n]\ , C_q[i]) = \prod_{x_i \in C_q[i]} [Weight\ (Prof_u[n], x_i)]_i$$

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. So this will give the smallest relevant sub tree contains the previous concepts $C_{q=}<C_1,C_2,\ldots,.C_i>$.

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); Figure 3.10 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:

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

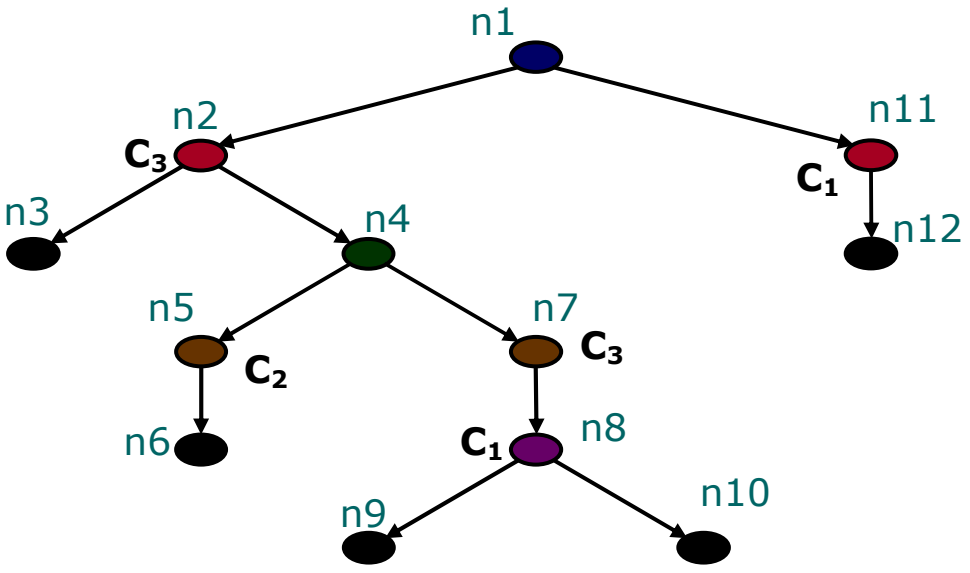


Figure 3.10. Example of a user profile graph $Prof_u$.

We then 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:

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

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

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

And:

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

OR:

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

We do the same steps to compute the relevance score of the other user profile nodes, the results are shown in Table 3.5 for the “And method” and in Table 3.6 for 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 3.5 below.

But if we consider the “*OR*” method then the node n_7 has the most relevance score, as shown in Table 3.6 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 3.5. Relevance score of user profile concepts $Prof_u$ using "*And* method".

Node	C_1	C_2	C_3	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

Table 3.6. Relevance score of user profile concepts $Prof_u$ using "*Or* method".

Node	C_1	C_2	C_3	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

3.6 State-based Query Reformulation (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.

3.6.1 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. These queries can be handled by studying the various states of the current task. The states 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. 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_iRQ<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_iRQ<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) which will aid to provide context-based personalized results. 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.

As we explained previously, each query session is defined by the tuple < 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. Thus, each query session in a task state is affected by the previous task state, except of the first query session.

3.6.2 Query Reformulation Phases

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

Figure 3.11 illustrates the SRQ Model, (State Reformulated Query).

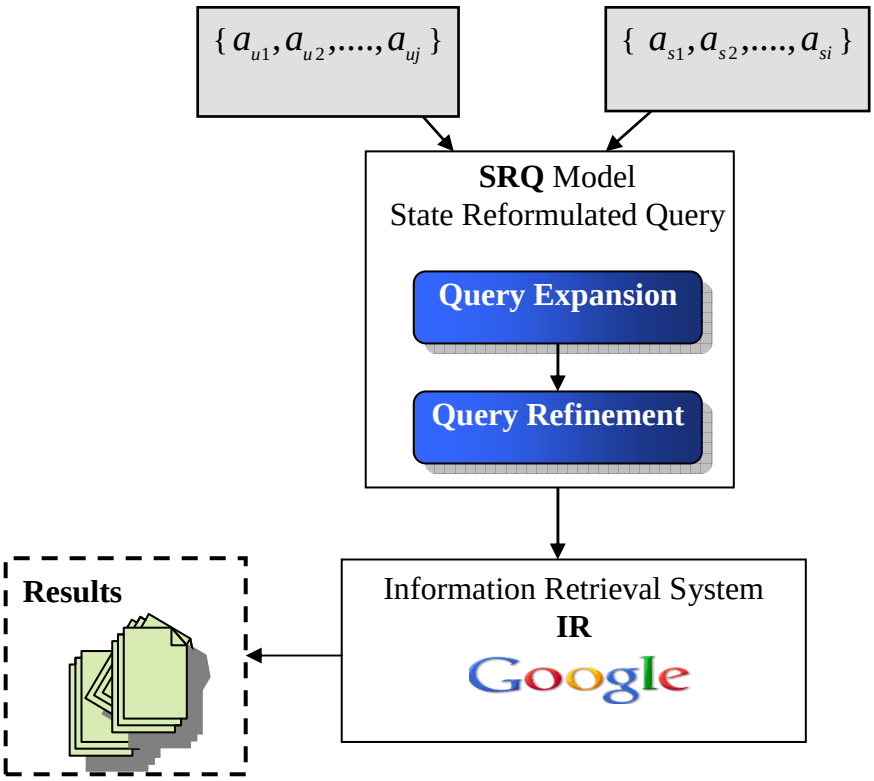


Figure 3.11. SRQ Model.

3.6.2.1 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}. Knowing that each main task consists of several activities, each one expresses a task state. These attributes are computed using the *Task model* which was explained in Section 3.4.
- Terms which represent the query-relevant concepts from the ontological user profile with its values (operational profile). (<a_{u1}, v_{au1}>, < a_{u2}, v_{au2}>, …, <a_{uj}, v_{auj}>). The algorithm of extracting these terms from the ontological user profile was explained in Section 3.5. These terms are denoted user profile attributes (a_{u1}, $a_{u2,}$…, a_{uj}).

3.6.2.2 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 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.

3.6.3 System Architecture

Figure 3.12 illustrates the system architecture. It combines the three models which are described in the previous sections:

- The task model.
- The user profile model.
- The SRQ model.

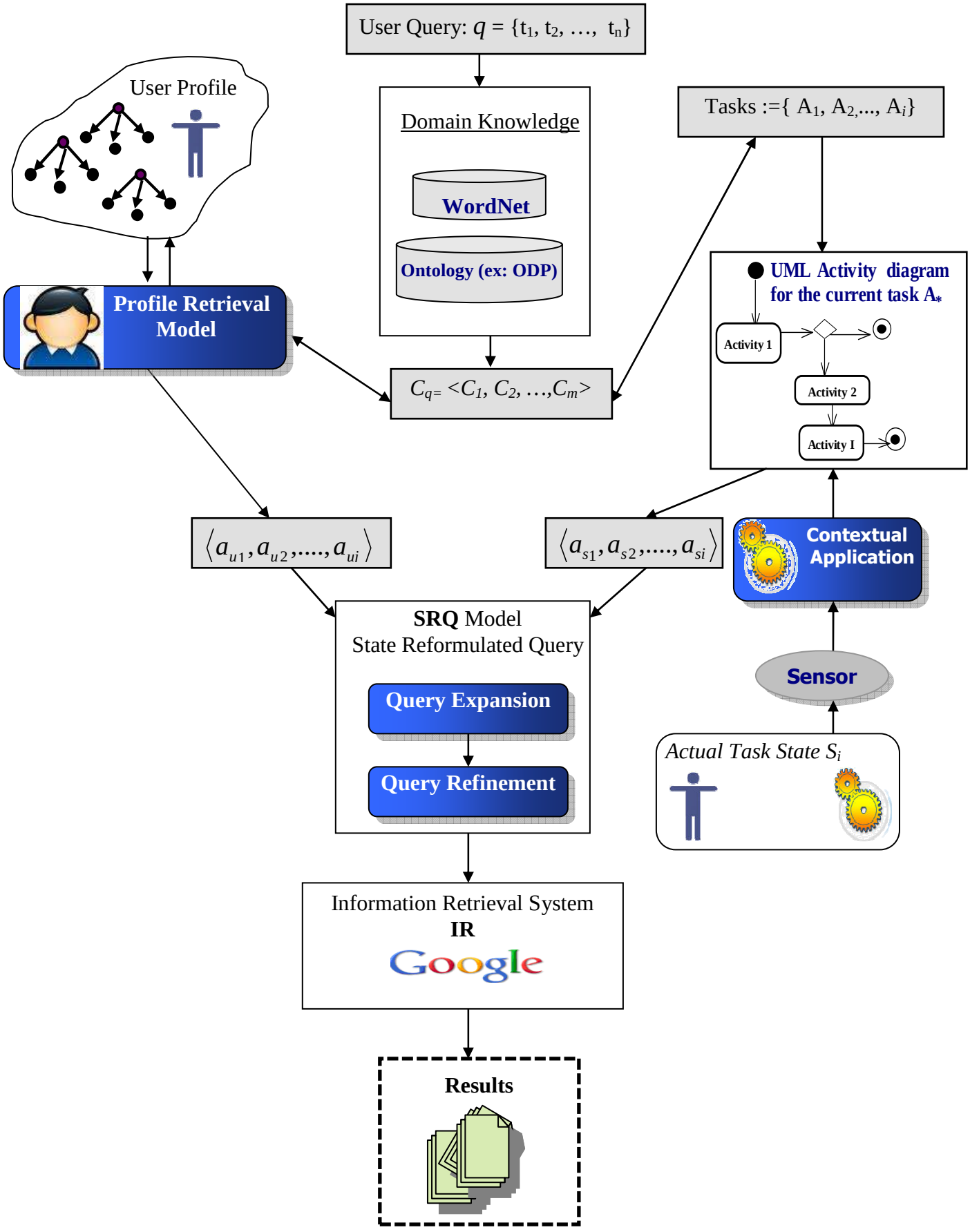


Figure 3.12. System Architecture.

3.6.4 Example

Here, we will summarize the example that we have taken in Section 3.4.1, the initial submitted query was q = {*Tourism in Toulouse*}, we presented in Table 3.1 the steps of our algorithm for detecting the user's task, the task that assigned to the user query q was: "*travel*" as it has the maximum similarity weight with the query context C_q.

We also presented in Section 3.4.2 that the task model allows the proposition of few task states for the user *U*, and these task states are expressed by user activities and represented using UML activity diagram, we presented this diagram in Figure 3.4. Next the system can generate one state reformulated query SRQ for each task state.

Let's consider a user *U*, and the actual state S_i = "*book a hotel in Toulouse"* for the detected task "*Travel*" at the time *t*. We propose that the query session in this state is the first one, so there is no impact of the previous state, i.e. it's the first attempt of the user in submitting this query.

The steps of our methodology to generate the state reformulated query S_iRQ for the user *U* at the task state S_i are summarized in Table 3.7.

Table 3.7. Description of the S_iRQ generating phases.

Description	Knowledge used	Output
Parsing the initial query q using *WordNet*	*WordNet*	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 *ODP* Taxonomy.	Set of concepts: query-context ($C_{q=}$ <C_1, C_2, …,.C_m> with m≥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, ….*)

Compare Cq $_<C_1$, C_2, …,C_m> with the user profile concepts $Prof_u$ in order to retrieve the relevant user profile attributes that will form the *operational profile*.	Ontological user profile which contains concepts and their interest values.	Set of common concepts (*travel, restaurant, food, hotel, vacation, outing*) and their interest values (*operational profile*): <travel> <Method_of_travel>**Airline**</Method_of_travel> <Price>**Inexpensive**</Price> </travel> < restaurant > <Food_type>**Vegetarian**</Food_type> <Cuisine>**Italian**</Cuisine> <Price>**Inexpensive**</Price> </restaurant> <Hotel> <Star_rating>**2 star**</Star_rating> <room_type>**single**</room_type> </Hotel> <Vacations> <vacation_type>**monuments**</vacation_type> </Vacations> <outing>**club**</outing>
Extend the query q with the actual state attributes (a_{s1},..a_{si}) : *Query Expansion*	UML activity diagram, at least one relevant attribute a_{si} for each task state: e.g. the actual state is looking for a hotel, thus: a_{si}:="*hotel*"	Extended query:= *tourism, Toulouse + hotel +* ***<travel>*** *Method of travel :* ***Airline*** *Price :=****Inexpensive*** ***< restaurant >*** *Food type:=* ***Vegetarian*** *Cuisine :=* ***Italian*** *Price :=****Inexpensive*** ***<Hotel>*** *Star_rating :=* ***2 star*** *room_type :=* ***single*** ***<Vacations>*** *vacation_type :=* ***monuments*** *outing :=****club***
Exclude the irrelevant terms to the actual state, and exclude duplicated terms: *Query Refinement*	A similarity comparison between the actual state attribute as_i and the operational profile.	S_iRQ: = *Tourism + Toulouse + hotel +* *Star_rating :=* ***2 star*** *room_type :=* ***single***

Construct the final reformulated query using the appropriate syntax	Search engine syntax. And ≈ ” “	Terms that represent the S_iRQ query: S_iRQ := *Tourism + Toulouse + “hotel” +2 stars OR single*
Submit S_iRQ to the search engine, Google for example, and provide the results back to the user	none	Results

Now, if the same user *U* submits the same query q at the time $t+\varepsilon$, and the actual task state at this time ($t+\varepsilon$) was $S_{i+\varepsilon}$ = “*find a restaurant* “, then Our system can propose the reformulated query (denoted $S_{i+\varepsilon}RQ$), at this task state $S_{i+\varepsilon}$. To generate this $S_{i+\varepsilon}RQ$, Table 3.7 will be changed as shown in Table 3.8: (steps 1, 2, and 3 don’t change but the results of the steps 4, 5, 6 and 7 will change depending on the actual state $S_{i+\varepsilon}$).

Table 3.8. Description of the $S_{i+\varepsilon}RQ$ generating phases.

Description	**Knowledge used**	**Output**
Extend the query q with the actual state attributes $(a_{s1},..a_{si})$: *Query Expansion*	UML activity diagram, relevant attribute $as_{i+\varepsilon}$ for the actual task state $S_{i+\varepsilon}$: $as_{i+\varepsilon}$ =“*Restaurant*”	Extended query*:= Tourism, Toulouse + restaurant +* <***travel***> *Method of travel :* ***Airline*** *Price :=****Inexpensive*** **< *restaurant* >** *Food type:=* ***Vegetarian*** *Cuisine :=* ***Italian*** *Price :=****Inexpensive*** <***Hotel***> *Star_rating :=* ***2 star*** *room_type :=* ***single*** <***Vacations***> *vacation_type :=* ***monuments*** *outing :=****club***

Exclude the irrelevant, duplicated terms: *Query Refinement*	A similarity between the actual state attribute $as_{i+\varepsilon}$ and the operational profile.	$S_{i+\varepsilon}RQ$ = *Tourism + Toulouse + restaurant +* *Food type:= **Vegetarian*** *Cuisine := **Italian*** *Price := **Inexpensive***
Construct the final reformulated query using the appropriate syntax	Search engine syntax. And ≈ ” “	Terms that represent the $S_{i+\varepsilon}RQ$ query: $S_{i+\varepsilon}RQ$ = *Tourism + Toulouse + “restaurant” + Italian OR Vegetarian*

Thus, in the same way, the system can generate the other reformulated queries SRQ at the different states of the task “*Travel*” which is represented by UML activity diagram (Figure 3.13). These reformulated state queries SRQ for the user *U* are:

- S_1 (book a flight): S_1RQ: {*Tourism + Toulouse + “Flight” OR Ticket + OR Inexpensive*}.
- S_2 (book a hotel): S_2RQ: {*Tourism + Toulouse + “hotel” +2 star OR single*}.
- S_3 (Preparation of the program): S_3RQ: {*Tourism + Toulouse + “Monuments” OR Weather OR plan OR Metro*}.
- S_4 (find a restaurant): S_4RQ: {*Tourism + Toulouse + “restaurant” + Italian OR Vegetarian*}.
- S_5 (photos of Toulouse): S_5RQ: {*Tourism + Toulouse + “Photos”*}.
- S_6 (watch the news): S_6RQ: {*Tourism + Toulouse + “News” OR Weather*}.

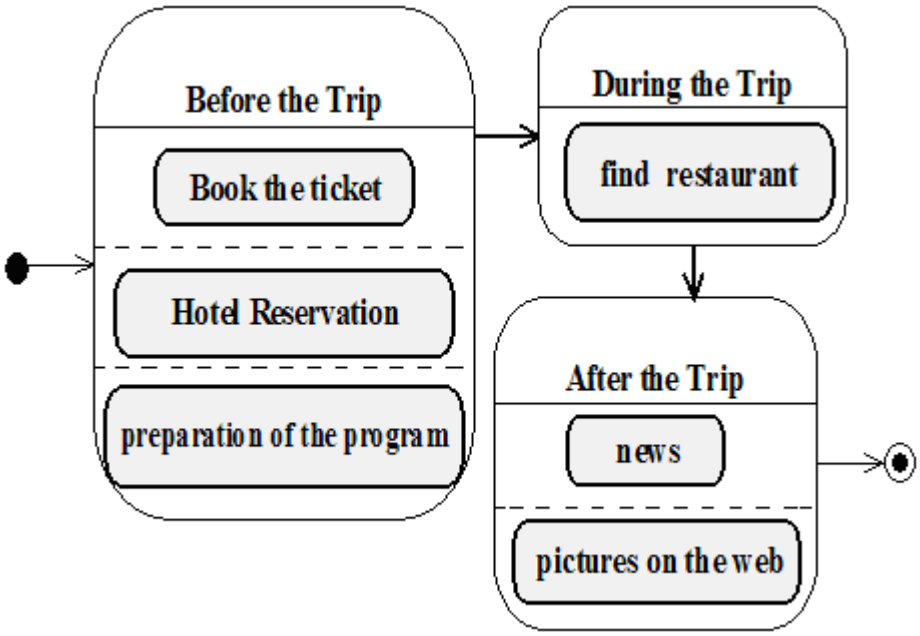


Figure 3.13. Example of a "*Travel*" task represented by using UML activity diagram.

Finally, the new reformulated query S_iRQ will be submitted to the preferred search engine in order to retrieve the relevant results that will aide the user to perform his/her current task. Hence, we will prove by an experimental study in Chapter 4 that the results D_i at the task state S_i are more relevant than the results that are returned by using the initial query q at the same state S_i.

3.7 Scenarios

In order to proof the effectiveness of our system and understand its different phases and the functions of the models during time changes, we can imagine many scenarios for the different task types. We can use two main types: outside scenario and inside scenario.

- *Outside scenario*: for example the shopping assistant scenario or a tourist guidance scenario which was discussed in Section 3.6.4 as an example of this type. This is the case of a mobile user.
- *Inside scenario*: for example the user tries to organize a trip.

3.7.1 Shopping Assistant Scenario

This is an *outside scenario*, let suppose the user U_1 has a PDA as a main device for both locating user and presenting information.

The user is at the shopping center trying to figure out what he needs to finish his shopping, a shopping assistant, which is applied in PDA, can:

- Tell the user what parts he needs.
- Where to find them relative to his location in the store.
- What is on sale?
- Do comparative pricing.
- Use his previous profile information to customize shopping and delivery.

For this scenario the user submits his query, such as: “shopping”, “buying an item“…etc, to the system. This query is short and it will not provide relevant results at each state of the current user’s task. Table 3.9 presents the relevant results at the different task states. Thus the shopping assistant must reformulate this query to provide the desired results at each task state as shown in Table 3.9.

Table 3.9. Shopping Assistant Scenario.

Task States	Desired Results
S_1 Tell you what parts you need	Items, product,…etc.
S_2 Where to find them relative to your location in the store.	Addresses of the shops that contain the desired items.
S_3 What is on sale?	Items exist in the shops.
S_4 Do comparative pricing.	Prices of the items.
S_5 Use your previous profile information to customize shopping and delivery.	The shops by the delivery.

Let us consider the initial query q= {Laptop}, for the user *U*, the system represents the different states of the current task using UML activity diagram which is shown in Figure 3.14.

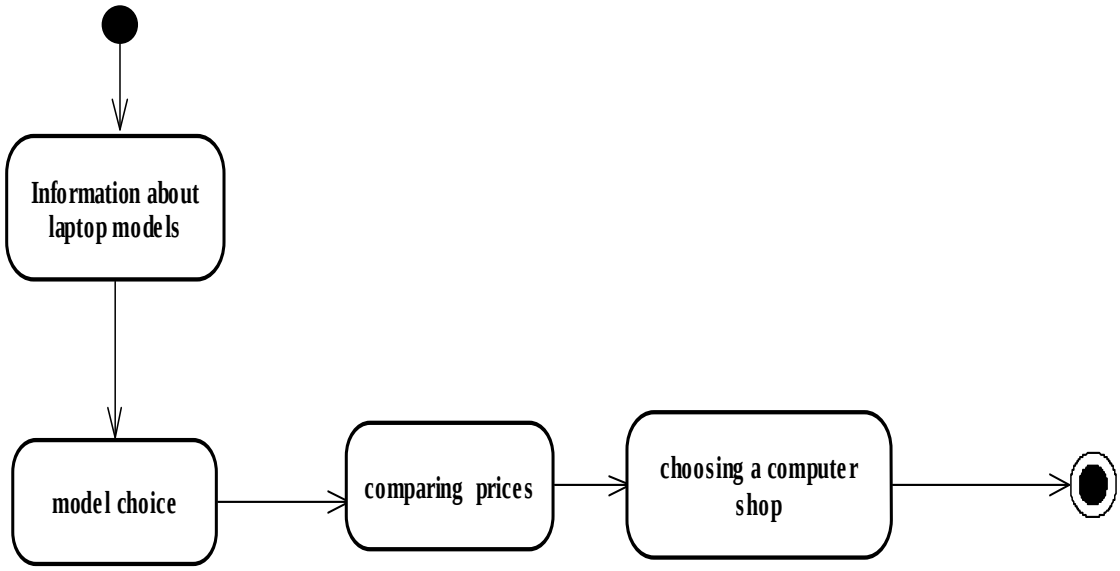


Figure 3.14. Example of a task "*Shopping*" represented by UML activity diagram.

Next, our query reformulation system can propose new queries for each task state in order to guide the search to the desired results at the actual state. Thus, for the initial query q, the proposed state reformulated queries SRQ will be:

- S_1 (Information about laptop models): S_1RQ: {*laptop+ "information"*}.
- S_2 (model choice): S_2RQ :{ *laptop+ "model" + HP OR Asus*}.
- S_3 (comparing prices): S_3RQ :{ *laptop+ "price" OR Inexpensive*}.
- S_4 (choosing a computer shop): S_4RQ: {*laptop+ "address" OR Paris*}.

Where:

"HP", "Asus", "Inexpensive" and "Paris" are the relevant terms from the user operational profile.

"Information", "model", "price" and *"address"* are the terms that represent task state attributes.

3.7.2 Composing a travel plan Scenario

We can consider the task of composing a travel plan. This is an *inside scenario,* it is one of limited complexity. A travel plan in this environment consists of a destination, an accommodation, a travel means and travel route, and the costs. An aid to the user in his/her search during this task to retrieve the desired results at the actual state of this task will be useful.

The task "*compose a travel plan*" includes the following states or sub tasks: (1) Choose a destination, (2) Choose an accommodation, (3) Find a travel route, (4) Calculate total costs, as shown in Figure 3.15 below.

Apart from subtasks (2) and (3), which are order-independent, the tasks must be performed in the given order. If an impasse occurs at some stage due to a mismatch between options available and the user's requirements, or because constraints of the overall task, e.g. cost limits are violated, the user may need to backtrack and redo preceding subtasks.

As scenarios may be depicted using sequence diagrams, we can represent this task "*compose a travel plan*" by a UML activity diagram that includes the user activities required to accomplish this task and related temporal relationships between them as shown in Figure 3.15. Each activity in this UML diagram represents a task state. The task state (task context) would indicate what information currently required for answering the user's query at this state and then moving to the next state in the UML diagram. Knowing that, a state is a stage of the task processing, or an efficient way of specifying a particular behavior.

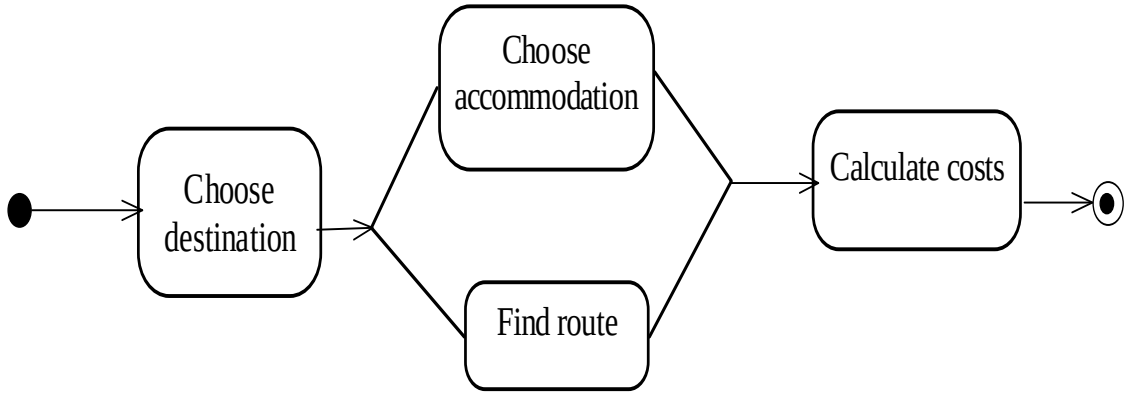


Figure 3.15. UML activity diagram for the task "*compose a travel plan*".

Queries are formulated and modified incrementally, by adding or removing terms at each task state, before the information retrieval system (Google for example) is required for choosing a destination and an accommodation, respectively. Both tasks typically involve a cycle of specifying relevant search attributes, and evaluating and modifying queries, until a (preliminary) decision is made concerning the desired destination or accommodation.

Although the user's precise goals are unknown and cannot be directly mapped to a specific query, it can be safely assumed that at some states he will have the goals of searching for destinations or accommodations. This eliminates some of the uncertainty about users' intentions that renders user behavior in IR so elusive. In addition, these goals can be related to particular types of information being searched for or being specified as search criteria by the user.

Considering the initial query q= {*Trip*}, for the user *U*, and the different states of the current task which is represented using UML activity diagram (Figure 3.15), our system will generate the following SRQ:

- S_1 (Choose a destination), S_1RQ: {*Trip* + "*destination*" + *Spain OR beach*}.
- S_2 (Choose an accommodation), S_2RQ: {*Trip* + "*accommodation*" + *hotel OR 2 star}.*
- S_3 (Find a travel route), S_3RQ: {*Trip* + "*Flight*" + *Ticket OR Inexpensive*}.
- S_4 (Calculate total costs), $\mathbf{S}_4$RQ: {*Trip + Spain + "budget" +Price}.*

Where:

"*Spain*", "*beach*", "*hotel*", "*2 star*" ,"*Ticket*" *and* "*Inexpensive*" are the relevant terms from the user operational profile.

"*Destination*", "*Accommodation*", "*Flight*", "*Price*" and "*budget*" are the terms that represent task state attributes.

Chapter 4

Implementation and Evaluation

This chapter includes an implementation description and an evaluation component of the proposed approach. The evaluation of the personalized information retrieval in context systems is known to be a difficult and expensive (Yang and Padmanabhan, 2005) 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 thesis have been implemented in an experimental prototype, and tested by real users. We will discuss the implementation of our system and its evaluation.

4.1 Implementation

The proposed methodology has been implemented in a prototype using *J2EE* technologies. The models interact with *WordNet* through its Java API, which is used to get the query synonyms, and they interact with ODP (Open Directory Project) taxonomy through its RDF data and Java API to identify the correct senses of the query terms in the ontology, that means the models interact with the ODP taxonomy in order to identify relevant concepts to the query for making inferences about the concepts related to user's query terms.

We use *XPath* and *Dom4j* to parse the user profile tree and retrieve relevant concepts from it. *Dom4j* is an open source Java library for working with XML, XPath and XSLT. It is compatible with DOM, SAX and JAXP standards.

The documents of user profile are indexed by using the database *MySQL* version 1.2.12. Also we use *NetBeans* IDE 5.5.1 to construct the main interface of our system which is shown in Figure 4.1.

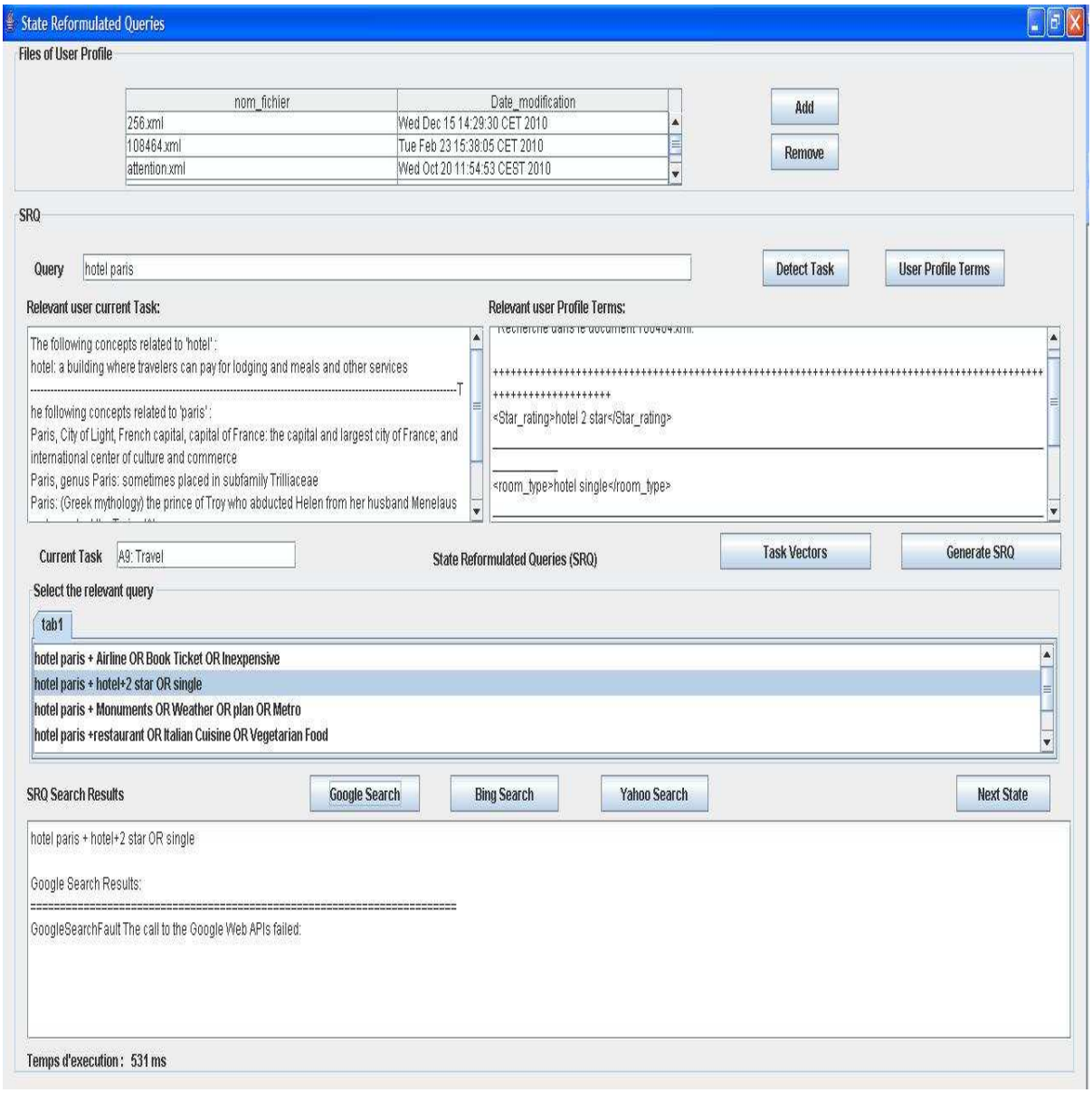


Figure 4.1. Interface of generating expansion terms $E^{(q)}$.

Figure 4.1 illustrates the main interface of our system; and following parts are illustrated in this interface:

- The part of user profile index and the ability of adding and deleting files.
- The part of relevant user profile terms.
- The part of parsing the user query through *WordNet* and Ontology and detecting the current task.
- The part of presenting state reformulated queries SRQ and submitting them to the preferred search engine.

In order to facilitate the evaluation processes, we construct a JSP page that contains the submitting of the initial query q and SRQ to Google, here we use the Apache Tomcat 7.0.4 Server. This JSP page is shown in Figure 4.2.

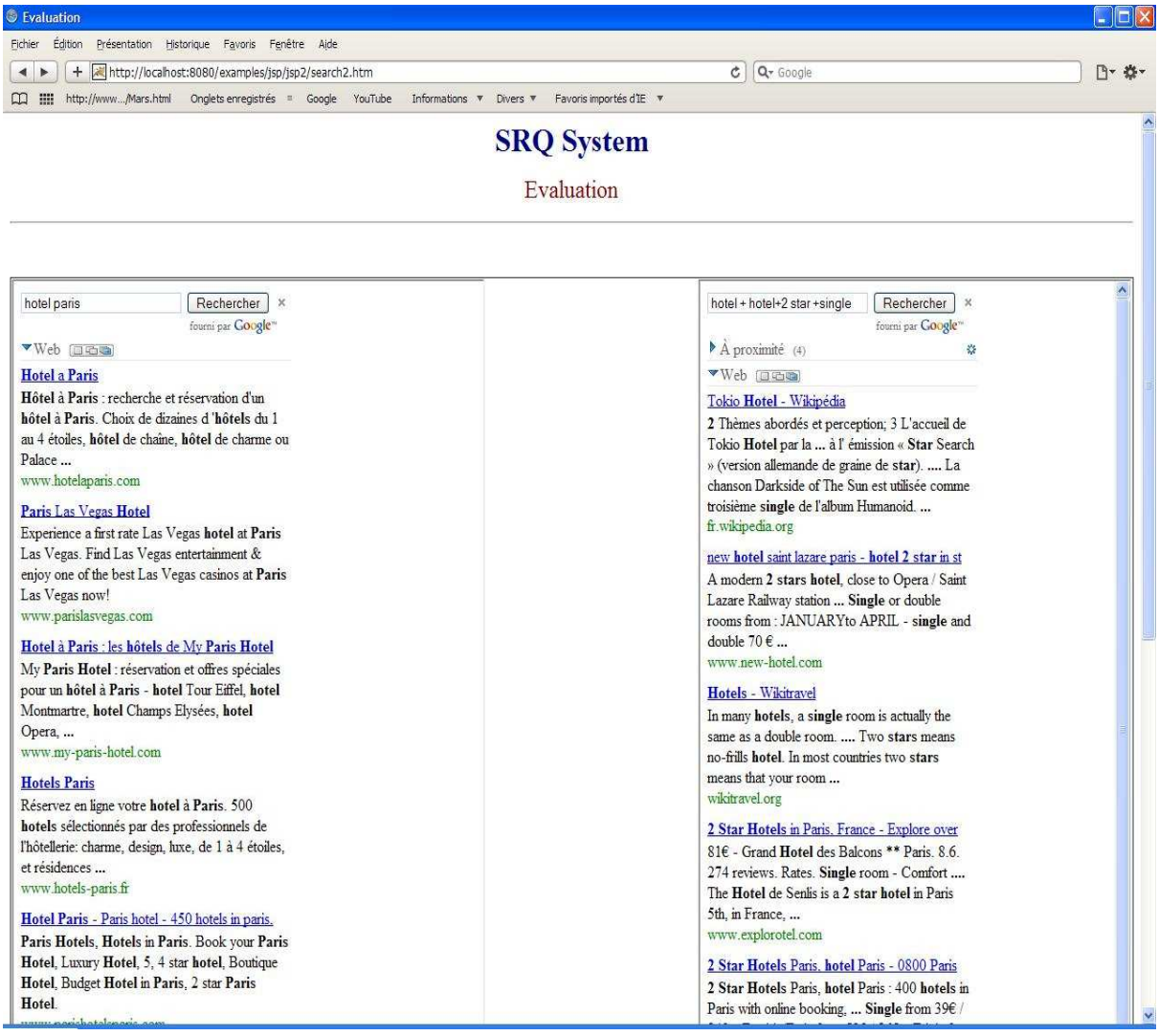


Figure 4.2. Interface which presents to the user to select the relevant documents from the returned results.

4.2 Evaluation

Evaluation in the context of an evolving real-world system is always a challenge, but in fact, the *evaluation* of such systems is complicated due to the dynamic aspect of the system environment. 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 into account the context. A third agent «personalization with context» is also used.

4.2.1 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) (Baeza-Yates et al., 1999), 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) (Kraft et al., 2006; Shen et al., 2005 a),

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 in a IIR (Interactive Information Retrieval) (Shen et al. 2005 a), 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.

4.2.1.1 Quality

The best evidence to verify the quality of the expanded terms or retrieval effectiveness of a system is to cross check with the documents actually visited by the user for the subjected query. Let q be an initial query and $D_c^{(q)}$ be the set of documents actually visited by the user for q. Now, given an IR system and 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:

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

Where:

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

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

4.2.1.2 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 the 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 means:

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

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

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

4.2.1.3 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:

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

If there are *n* instances of the query q then we estimate the average dynamics as follows:

$$E(\delta^{(q)}(i, j)) = \frac{n(n-1)}{2} \sum_{i \neq j} \delta^{(q)}(i, j)$$

We will illustrate how the three previous metrics can be computed after the experimental study presentation.

4.2.2 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. For all experiments, the prototype focus on the first 20 results of Google search engine and presenting them to users.

To conduct the experiments and calculate the previous three 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 the scenarios presented in the previous sections, ranging from travel, to shopping, to restaurant searching, and we added other scenarios that will be illustrated in the next section. Consequently a total of 30 queries are selected as experimental queries.

The users are also asked to look through all the results returned by Google before clicking on any result. The prototype records results on which they 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.

In the following section, we will present three different scenarios. These three scenarios with the two previous scenarios that are presented in chapter 3 (*Tourism in Toulouse* presented in Section 3.6.4, and scenario of *shopping assistant* presented in Section 3.7.1) and the queries submitted during these scenarios are used to calculate the evaluation metrics.

4.2.2.1 Scenario (1)

Assume that we have a mobile user, he/she can surf the Internet by a PDA, he/she is in the city center of Paris, and he/she has one task at hand, which is looking for a restaurant for the dinner. The user is novice in computer science, to perform this task he/she submits the query: "*restaurant dinner*". For this initial query, the user must choose and visit the relevant documents from the Google presented results at this actual context. The user can evaluate them by exploring at the snippets or sites.

After parsing the user's input using linguistic knowledge (WordNet) and semantic knowledge (Ontology), the system searches in the user profile concepts to retrieve preferences for restaurants and food habits. The restaurant preference frame shows that the user likes *Italian* food, likes *Vegetarian* food and doesn't like to drive too far, the user lives in a Paris suburb in France.

All preference information, for restaurants and food habits, are added to the user query, but in some specific context (in actual state of the current task) the user might not want to use some of it. (For example, the user may relax the driving distance restriction because he/she is in vacation). The additional contextual information is also added to the initial query and the SRQ model will generate the new state reformulated query.

Thus the system can propose to the user new state reformulated query:

SRQ: (*Restaurant* + *dinner* + *Italian* + *vegetarian* + *Paris*)

The generated SRQ contains related terms from the operational user profile and his/her context, this SRQ query is submitted to the search engine Google, we present to the user the first 20 returned results using SRQ, which is generated automatically

by our system, and then he/she is asked to evaluate them. Figure 4.3 shows the interface of generating SRQ query and the returned results for this query.

As we mentioned previously, each scenario is performed nearly by 10 users who are asked to evaluate the returned results, we suppose them to have the same profile and context for this scenario, but they could have different profiles for the other scenarios. Finally we can calculate the three selected evaluation metrics for this scenario; we will illustrate that in Section 4.2.3.

After calculation the average number of relevant documents at the first 20 results (P@20) for the initial query q and the new query *SRQ*, we notice that the precision of the relevant results using the initial query q is 0.13 and 0.54, respectively, by using *SRQ* query which is generated automatically by our system depending on the actual state of the current user's task and his/her profile.

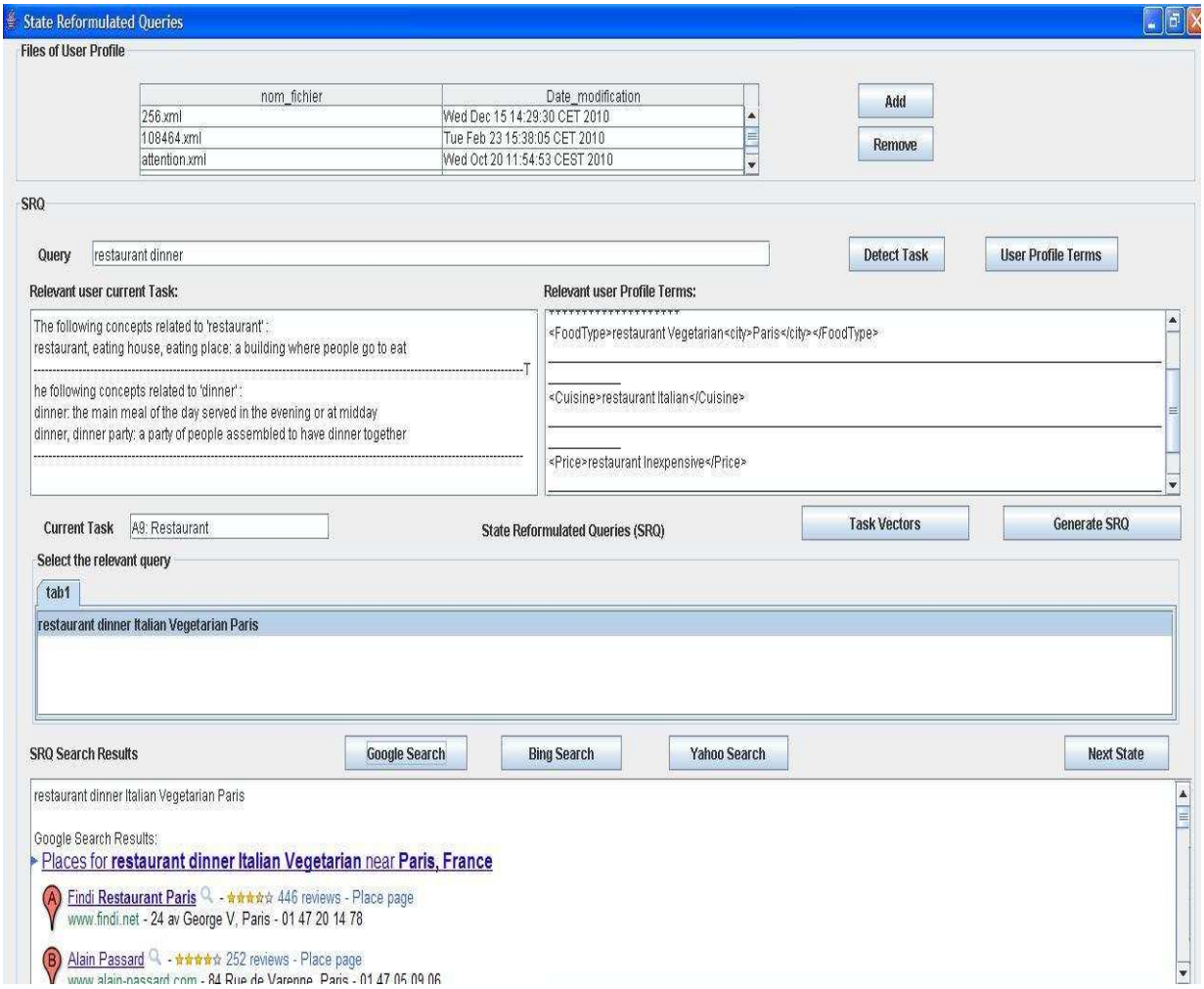


Figure 4.3. Interface of generating SRQ for scenario (1) and the returned results.

4.2.2.2 Scenario (2)

Suppose a mobile user is in the context of walking in the city center of Paris and he has one task on hand that is looking for places to drink mate, which is a kind of tea frequently drunk in South America. In fact the user has a medium level in computer science. To perform his/her current task, he/she submits the query: "*drinking mate*". In the first step, the query is parsed and the output is the set of initial query terms, namely q= {*drinking*, *mate*}. Now we apply the steps of our methodology to generate the state reformulated query *SRQ* for this user at this current task. These steps were shown previously in Table 3.7 for the task "*tourism in Toulouse*".

After q parsing in WordNet and ODP taxonomy, we found three senses for the term *drinking*, namely: Alcoholic beverage, Drink as a noun and Drink as a verb (the act of drinking), and the following senses for the term *mate* (Figure 4.4):

- Paraguay tea, Ilex paraguariensis,
- South American tea,
- Spouse, partner, married person,
- Copulate, pair, couple: make love
- Team-mate, mate: a fellow member of a team,
- Checkmate, mate: place an opponent's king under an attack from which it cannot escape and thus ending the game.

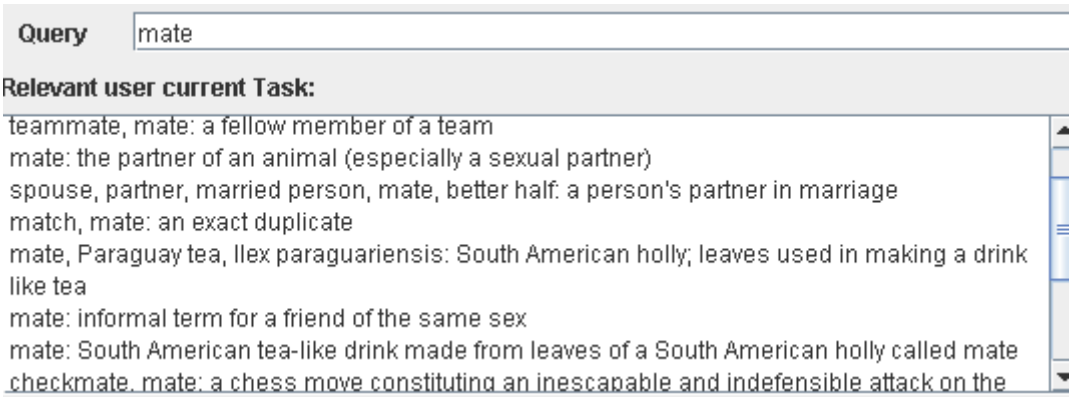


Figure 4.4. Interface of parsing the term *mate*.

Thus, the set of query-relevant concepts, that form query-context vector $C_{q=} <C_1, C_2, \ldots, C_m>$ (with m≥2), are: {{Alcoholic beverage, Drink, Drink Event}, {partner, Tea-Beverage, couple, love, Checkmate}}.

Now, we activate for each query-context concept $C_q[i]$ its semantically related concepts from the ontological user profile $Prof_u$. The relevant nodes from the ontological user profile for the query-context concepts are:

```
<info xmlns="urn:mime:xml/user-profile">
<drinking>

<drinking> drinking tea </drinking >
<drinking> drinking coffee </drinking >
<drinking> drinking mate
<mate> tea beverage </mate></drinking>

<Address> drinking
<street> st germain</street>
<zipcode> 75014 Paris</zipcode>
<country> France </country>
</Address>

</drinking>
</info>
```

Figure 4.5 shows the interface of our system for the relevant nodes which are extracted from the ontological user profile for the query *q=" drinking mate"*.

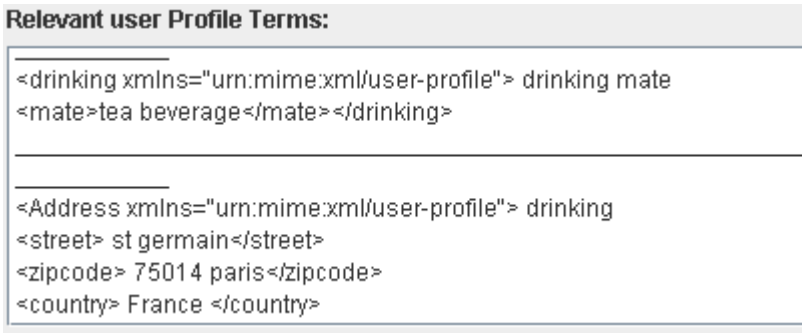


Figure 4.5. Interface of extracting relevant nodes from the ontological user profile.

This current task has two states or search sessions: information about the mate beverage, and place's address to drink mate. Consequently the two generated SRQ queries will be:

- At S_1: Information S_1RQ: *Drinking mate + tea beverage + "information"*.
- At S_2: Address S_2RQ: *Drinking mate + tea beverage + "Address" + Paris 75014*.

Now the user must visit and choose the relevant documents at the two states S_1, S_2 from the Google results which are returned by using the initial query q. The user can evaluate them by exploring at the snippets or sites. And then, he/she must evaluate the first 20 Google returned results in the two cases by using S_1RQ and S_2RQ, respectively. Then we will be able to compute the evaluation metric *Precision@20*. We will see that the *Precision@20* of the results which are obtained by using the reformulated queries, is elevated compared with those obtained by using the initial query at the same task state.

4.2.2.3 Scenario (3)

Assume a user has one task to do, which is looking to buy forks. The user has a medium level in computer science; he/she may submit the query ‘‘*buying forks*”. If we execute this query in Google, only one of the first 10 results is relevant to the user.

The initial query terms are q= {*buying, forks*}, the query is parsed to identify the query-context concepts: $C_{q=}$ <C_1, C_2, …,C_m>. For that our system identifies the synonyms and concepts that are linguistically and semantically related to the query using WordNet and Ontology. Our system interface shows the different meanings of the query terms and mapping these concepts on the ontological user profile to activate the related user profile nodes (operational profile). The query-context of the term “*forks*” contains the following concepts: {*Kitchenware, eating, branching, ramification, agricultural tool, pitchfork, crotch*}. Figure 4.6 shows the interface of our system for the query-context of the term “*forks*”.

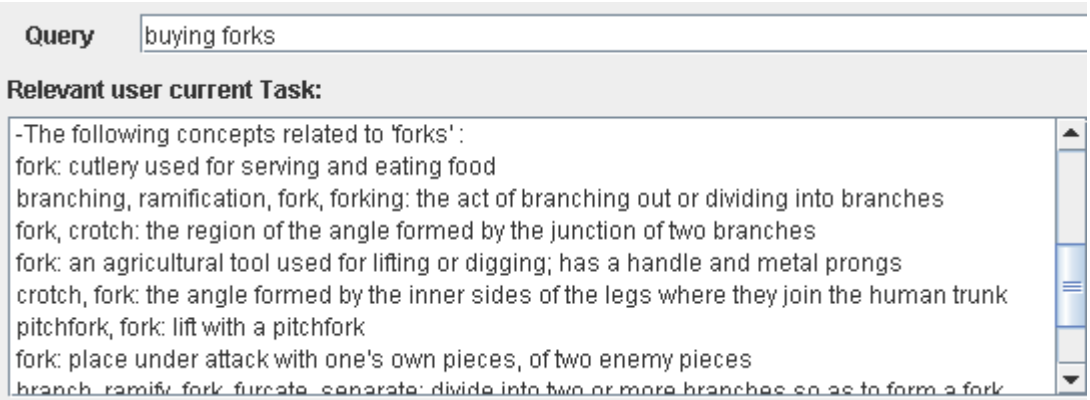


Figure 4.6. Interface of parsing the term “*forks*”.

The query-context concepts which are related to the term "*buying*" are: {*Purchasing, the act of buying, acquires, bribe, corrupt, grease one's palms, acquire by trade*}. Figure 4.7 shows the interface of our system for the query-context of the term "*buying*".

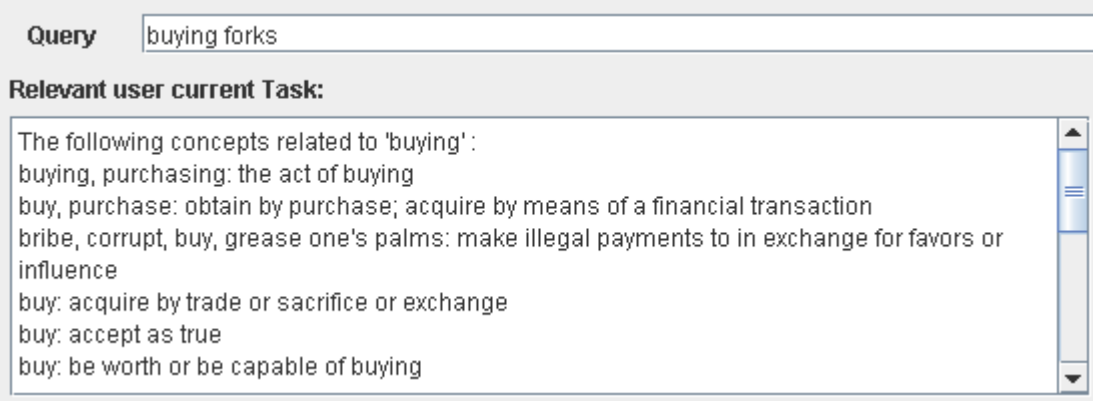


Figure 4.7. Interface of parsing the term "*buying*".

In the same steps that previously had been shown in Table 3.7, the system can detect the current task that is "*Shopping and selling*", and after comparing the query context with the user profile and activating its relevant nodes, the system can generate the state reformulated queries SRQ at each task state, for that the user only clicks on the ''Generate SRQ'' button in our system interface. Knowing that, the relevant nodes from the ontological user profile for the query-context C_q are:

```
<info xmlns="urn:mime:xml/user-profile">
<shopping>
< shopping> buying forks <forks>kitchenware eating </forks>
</shopping>
<Address> buying
<street> st germain</street>
<zipcode> 75014 paris</zipcode> <country> France</country>
</Address>
</shopping>
</info>
```

Figure 4.8 shows the interface of our system for the relevant nodes extracting from the ontological user profile for the query *q="buying forks"*.

The different states of this task are represented in UML Activity diagram, which is shown in Figure 4.9.

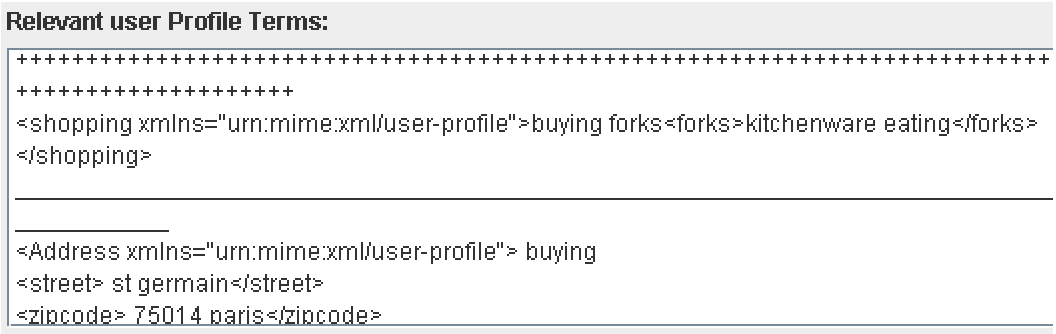


Figure 4.8. Interface of extracting relevant nodes from the ontological user profile.

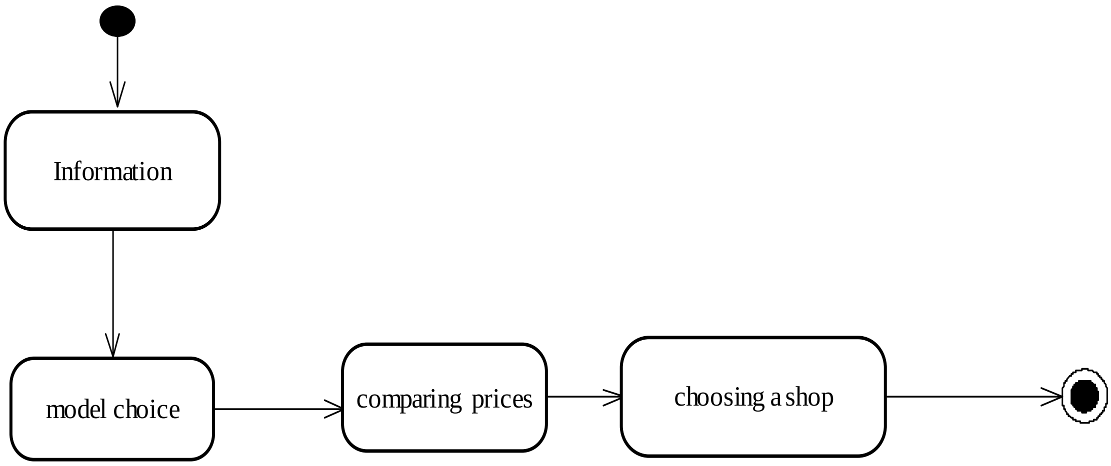


Figure 4.9. UML activity diagram for the "*Shopping*" task.

Thus the state reformulated queries SRQ at each task state will be:

- S_1 (Information): S_1RQ: {*buying forks* + *Kitchenware eating* +*"information"*}.
- S_2 (Model choice): S_2RQ: {*buying forks* + *Kitchenware eating* + *"models"*}.
- S_3 (comparing prices): S_3RQ: {*buying forks* + *Kitchenware eating* + "price"}.
- S_4 (choosing a shop): S_4RQ: {*buying forks* + *Kitchenware eating* +*"shop"* + *Paris 75014*}.

For each query S_iRQ, at each state *i*, the first 20 results produced by using Google are presented to the user, and he/she must evaluate them at each state *i*.

Using the user feedback, we will be able to compute the three selected evaluation metrics for this scenario. This will be done in the following section.

4.2.3 Computing the evaluation metrics based on the experimental scenarios

As we mentioned previously, in order to evaluate our proposed framework we will compute the three evaluation metrics: quality, precision@k, and dynamics, which are defined in Section 4.2.1, based on the previous experimental scenarios.

4.2.3.1 Quality

We mentioned in Section 4.1, and Figure 4.1, that we built an interface to receive the initial user queries q and propose expansion terms $E^{(q)}$ for these queries based on the user context and his/her profile. We use also a retrieval system consists of a meta-search which submits the user queries q, before adding the expansion terms, to Google search engine and presents the results to the user. Next from the returned results, the user visits the relevant documents to his/her actual state, and then the system saves these documents. We had shown in Figure 4.2 the interface that presents to the user in order to visit the relevant documents from the returned results for his needs at the actual context using q, these relevant documents are denoted: $D_c^{(q)}$. The user can also select the $D_c^{(q)}$ by exploring at the snippets. 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 the initial query 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.

Depending on what we mention above, a query has different search goals at different task states interval with time changing. We manually verify and mark these task state instances for our experimental queries, which were presented in the previous section. While verifying we broadly differentiate the goals.

Now after generating the expansion terms $E^{(q)}$, we can calculate the average quality of the expansion terms over 30 queries from the formula:

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

Where $D_c^{(q)}$: Set of documents actually visited by the user for the initial query q, or marked as relevant documents by exploring at the snippets. If a query has no visited documents, we simply ignore it.

$\rho\ (E^{(q)}, D_c^{(q)})$ is the matching terms between $E^{(q)}$ and $D_c^{(q)}$.

Consequently, as long as the expansion terms $E^{(q)}$ exist in the relevant results $D_c^{(q)}$ (visited documents), the quality of these expansion terms $E^{(q)}$ increases.

For example, if we take the scenario presented in Section 4.2.2.3 and the user query q=‘‘*buying forks*’’, during this scenario, we take the fourth state S_4 which is searching for the nearest shop, 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_4. If he/she visits 5 documents then $\left| D_c^{(q)} \right| = 5$.

At this actual state S_4, our system proposes set of expansion terms $E^{(q)}$, this set contains 7 terms which are: *buying, forks, Kitchenware, eating, shop, Paris, 75014*. Thus: $E^{(q)} = 7$. From these 7 terms, if there are 4 terms existing in the 5 visited documents $D_c^{(q)}$ at S_4, then:

$$\left| \rho\ (E^{(q)}, D_c^{(q)}) \right| = 4$$

Where: $\rho\ (E^{(q)}, D_c^{(q)})$ is the matching terms between $E^{(q)}$ and $D_c^{(q)}$.

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

$$Quality = \frac{\left| \rho\ (E^{(q)}, D_c^{(q)}) \right|}{\left| E^{(q)} \right|} = 0.57$$

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 is 0.83. This value is shown in Table 4.1.

Now we can compute the average qualities of other tasks and over other queries, the results are shown in Table 4.1. Table 4.1 shows the average quality of the expansion terms over all experimental queries (30 queries).

Table 4.1. The average qualities of the expansion terms over the 30 experimental queries.

Context (10 queries by 10 different users for each scenario)	Expansion terms $E^{(q)}$ for each state.	Average Quality
Scenario1: Buy forks by user living in Paris... For example, the query: *q=‘‘buying forks’’*. (Section 4.2.2.3)	*Buying, fork, Kitchenware, eating, information*	0.83
	Buying, forks, Kitchenware, eating, models	
	Buying, forks, Kitchenware, eating, price	
	Buying, forks, Kitchenware, eating, shop, Paris, 75014	
Scenario2: Trip to the Toulouse city… For example, the query: *q=“Tourism in Toulouse”*. (Section 3.6.4)	*Tourism, Toulouse, book, Flight, Ticket, Inexpensive,*	0.75
	Tourism, Toulouse , hotel, 2 star, single,	
	Tourism, Toulouse, Monuments, Weather, plan, Metro,	
	Tourism, Toulouse, Restaurant, Food, Italian, Vegetarian,	

	Tourism, Toulouse, Photos,	
	Tourism, Toulouse, News, Weather,	
Scenario3: Search place to drink mate in the city center of Paris… For example, the query: *q=“drinking mate”*. (Section 4.2.2.2)	*Drinking, mate, tea, beverage information,*	0.69
	Drinking, mate, tea, beverage, Address, Paris, 75014,	

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 same conditions, thus the $E^{(q)}$ will be different from the first case based on the user context and his/her profile. In this case and in the same steps we can calculate the average qualities of expansion terms $E^{(q)}$ which are extracted from the user profile and don't taking into account the user context at the same user's queries. These average qualities are shown in Table.4.2.

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. Figure 4.10 shows a comparison between the average qualities in the two cases at the three selected scenarios.

Table 4.2. The average qualities of the expansion terms which are generated depending only on the user profile over the 30 experimental queries.

Context (10 queries by 10 different users for each scenario)	Expansion terms $E^{(q)}$ depending on user profile only.	Average Quality
Scenario1: Buy forks by user living in Paris... For example, the query: ‘‘*buying forks*’’	*Buying, forks, Kitchenware, eating, Paris, 75014*	0.43
Scenario2: Trip to Toulouse city … For example, the query: “*Tourism in Toulouse”*.	*Tourism, Toulouse, book, Flight, Ticket, Inexpensive, book, hotel, 2 star, single restaurant, Italian, Vegetarian,*	0.34
Scenario3: Searches places to drink mate in the city centre of Paris… For example, the query: “*drinking mate*”.	*Drinking, mate, tea, beverage information, Paris, 75014,*	0.49

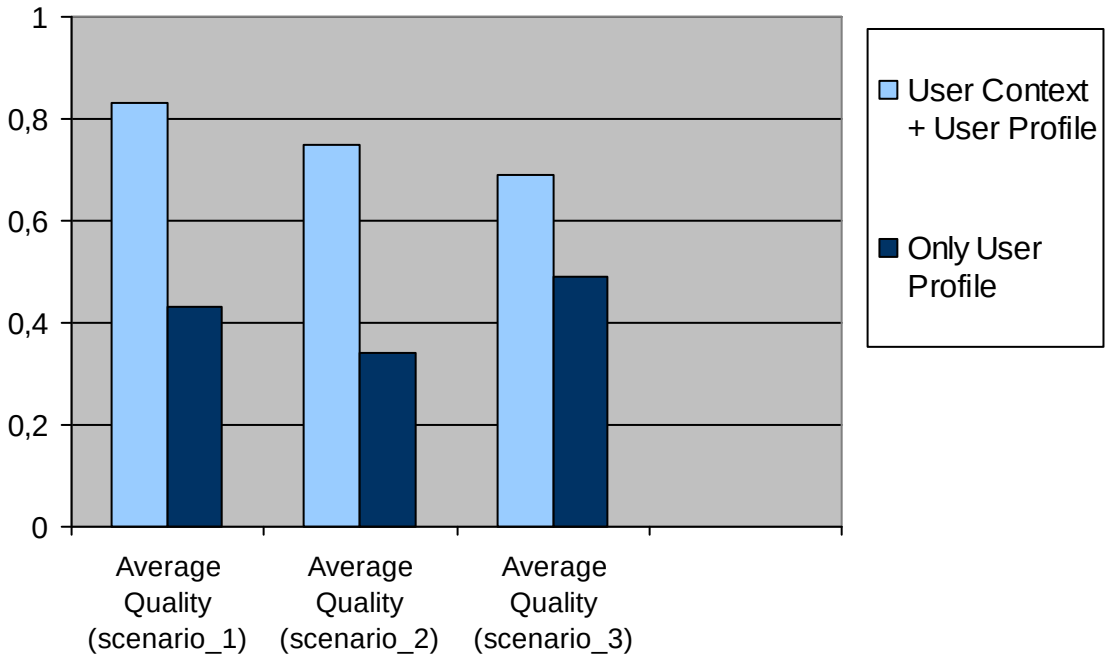


Figure 4.10. Comparing between the average qualities of the expansion terms which are generated in the two cases (depending only on user profile or depending on user profile and user context) over the experimental queries.

4.2.3.2 Retrieval Effectiveness (Precision@k)

We use the precision at *k* measure, which is previously defined in Section 4.2.1.2, in order to estimate the retrieval effectiveness. Let $D_n^{(SRQ)}$ be the set of top *n* documents retrieved by IR system using the state reformulated query SRQ which contains the expansion terms $E^{(q)}$. To facilitate the experiments, let's consider only the first 20 retrieved documents (*n*=20), thus $D_{20}^{(SRQ)}$ represents the first 20 documents from the retrieved results by the IR system (Google for example) by using the state reformulated query SRQ.

In the previous section, we explained 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 other words, $D_c^{(q)}$ represents the set of

visited documents by the user during the actual task state or judged by the user as relevant documents during the actual search session.

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. The cosine similarity is previously explained in Section 3.2.1. 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:

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

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

If a query has no visited documents, we simply ignore it. 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 (Figure 4.2).

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 same three previous 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 example in the previous section that was the scenario presented in Section 4.2.2.3 and the user query q=‘‘*buying forks*’’, in this scenario, we consider the fourth state S_4 which is searching for the nearest shop, at this actual

task state S_4, the $\left| D_c^{(q)} \right| = 5$ and the expansion terms $E^{(q)}$: {*buying, forks, Kitchenware, eating, shop, Paris, 75014*}, like that the S_4RQ will be (Section 4.2.2.3):

S_4RQ = buying forks + Kitchenware eating+ "shop" + Paris 75014

We execute this S_4RQ by using Google and then we compute $D_r^{(s_4rq)}$ in the three cases (k=5, k=10, k=20) by calculating the cosine similarity between $D_c^{(q)}$ and $D_5^{(S_4RQ)}$ for k=5, $D_{10}^{(S_4RQ)}$ for k=10 and $D_{20}^{(S_4RQ)}$ for k=20.

Knowing that $D_5^{(S_4RQ)}$ is the set of top 5 documents retrieved by IR system using S_4RQ, and:

$$D_r^{(s_4rq)} = \left\{ d_i \,\middle|\, d_i \in D_5^{(S_4RQ)}, \exists d_j \in D_c^{(q)} \text{ s.t. } Sim(d_i, d_j) \geq 0.5 \right\}.$$

Thus: $precision@5 = \frac{\left| D_r^{(s_4rq)} \right|}{5} = \frac{3}{5} = 0.6$

For K=10: $precision@10 = \frac{\left| D_r^{(s_4rq)} \right|}{10} = \frac{4}{10} = 0.4$

Where: $D_r^{(s_4rq)} = \left\{ d_i \,\middle|\, d_i \in D_{10}^{(S_4RQ)}, \exists d_j \in D_c^{(q)} \text{ s.t. } Sim(d_i, d_j) \geq 0.5 \right\}$

For K=20: $precision@20 = \frac{\left| D_r^{(s_4rq)} \right|}{20} = \frac{7}{20} = 0.35$

Where: $D_r^{(s_4rq)} = \left\{ d_i \,\middle|\, d_i \in D_{20}^{(S_4RQ)}, \exists d_j \in D_c^{(q)} \text{ s.t. } Sim(d_i, d_j) \geq 0.5 \right\}$

Otherwise we can calculate $D_r^{(s_4rq)}$ 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 previous three scenarios. Table 4.3 illustrates the precision of our system at k=5, k=10 and K=20 for the three experimental scenarios and over the 30 queries submitted by 10 different users during these 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 have to calculate the retrieval performance of the 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 three experimental scenarios and the same users.

To do that, we consider again the same state S_4 which was searching for the nearest shop. The number of relevant results at this task state was $\left| D_c^{(q)} \right| = 5$. The expansion terms $E^{(q)}$, for the initial query q, depending only on the user profile for all task states are: {*buying, forks, Kitchenware, eating, Paris, 75014*}. We execute this new query (*rq*) in Google and then we compute $D_r^{(rq)}$ in the three cases (k=5, k=10, k=20) in the same previous method.

Thus: For K=5: $$precision@5 = \frac{\left| D_r^{(rq)} \right|}{5} = \frac{1}{5} = 0.2$$

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

For K=10: $$precision@10 = \frac{\left| D_r^{(rq)} \right|}{10} = \frac{3}{10} = 0.3$$

For K=20: $$precision@20 = \frac{\left| D_r^{(rq)} \right|}{20} = \frac{4}{20} = 0.2$$

In the actual task state S_4, we also compare the two previous cases with the standard Google search by using the initial query q without any query reformulation.

Thus: For K=5: $$precision@5 = \frac{\left| D_r^{(q)} \right|}{5} = \frac{1}{5} = 0.2$$

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

For K=10: $precision@\ 10 = \frac{\left|D_r^{(q)}\right|}{10} = \frac{1}{10} = 0.1$

For K=20: $precision@\ 20 = \frac{\left|D_r^{(q)}\right|}{20} = \frac{2}{20} = 0.1$

In the same method, we calculate the precision of the user profile-based query reformulation system for the other task states and for the other experimental scenarios. The average of this precision at k= {5, 10, 20} are shown in Table 4.3. Table 4.3 also shows the average precision@k={5, 10, 20} of the standard Google search by using the initial query q for the three experimental scenarios and over the 30 queries submitted by the 10 users during these scenarios. Finally we calculate the average of the precision at k (where K=5, 10, 20) for all experimental queries at the task states which was presented in Table 4.3.

Figure 4.11 shows a comparison between the Precision@5 averages over the experimental queries at the three experimental scenarios in the three cases, same thing in the Figure 4.12 and Figure 4.13 for the Precision@10 averages and the Precision@20 averages respectively.

The precision@K averages of the three system types (reformulation based on user context and his/her profile, reformulation based only on the user profile, standard search without any reformulation) are shown in Table 4.4.

Figure 4.14 shows a comparison between the Precision@5, Precision@10, Precision@20 averages of our proposed system with those of the standard search and personalized search.

We notice from Table 4.4 and Figure 4.14 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 in a specific context than that of the classic information retrieval systems and the personalized retrieval systems in the same context.

Table 4.3. The Precision@k of the different systems.

Context (10 queries by 10 different users for each scenario)		Precision@*k* Google (Using q)			Precision@*k* Personalization (Using q + user profile *rq*)			Precision@*k* Personalization + Context (using SRQ)		
		K=5	*K=10*	*K=20*	*K=5*	*K=10*	*K=20*	*K=5*	*K=10*	*K=20*
Scenario1: Buy forks by user living in Paris... ex: q=‘‘*buying forks’’*	S_1	0,42	0,36	0,30	0,46	0,40	0,32	0,79	0,74	0,7
	S_2	0,22	0,29	0,20	0,32	0,30	0,25	0,62	0,51	0,30
	S_3	0,36	0,32	0,24	0,40	0,41	0,35	0,73	0,68	0,64
	S_4	0,15	0,1	0,05	0,2	0,27	0,15	0,56	0,34	0,32
Averages		0,287	0,267	0,197	0,345	0,345	0,267	0,675	0,567	0,49
Scenario2: Trip to the Toulouse city… ex: q=*“Tourism in Toulouse”*.	S_1	0,12	0,17	0,09	0,1	0,22	0,16	0,74	0,65	0,57
	S_2	0,16	0,26	0,21	0,13	0,20	0,26	0,62	0,45	0,43
	S_3	0,2	0,28	0,09	0,16	0,32	0,14	0,58	0,40	0,31
	S_4	0,08	0,05	0,06	0,06	0,12	0,08	0,56	0,44	0,26
	S_5	0,06	0,13	0,05	0,04	0,17	0,06	0,86	0,55	0,35
	S_6	0	0,02	0,04	0,02	0,02	0,04	0,42	0,39	0,29
Averages		0,103	0,152	0,09	0,085	0,175	0,123	0,63	0,48	0,368
Scenario3:	S_1	0,40	0,31	0,30	0,42	0,34	0,33	0,60	0,58	0,44

Search places to drink mate in Paris… ex: q="*drinking mate*".	S_2	0,14	0,09	0,08	0,16	0,12	0,09	0,42	0,32	0,26
Averages		0,27	0,2	0,19	0,29	0,23	0,21	0,51	0,45	0,35

Table 4.4. The Precision@k averages of the different systems.

Top K	Precision@*k* Google (Using q without any reformulation)	Precision@*k* Personalization (Using q + user profile *rq*)	Precision@*k* *Our System* Personalization+ Context (using SRQ)
K=5	0,22	0,24	0,61
K=10	0,21	0,25	0,50
K=20	0,16	0,20	0,41

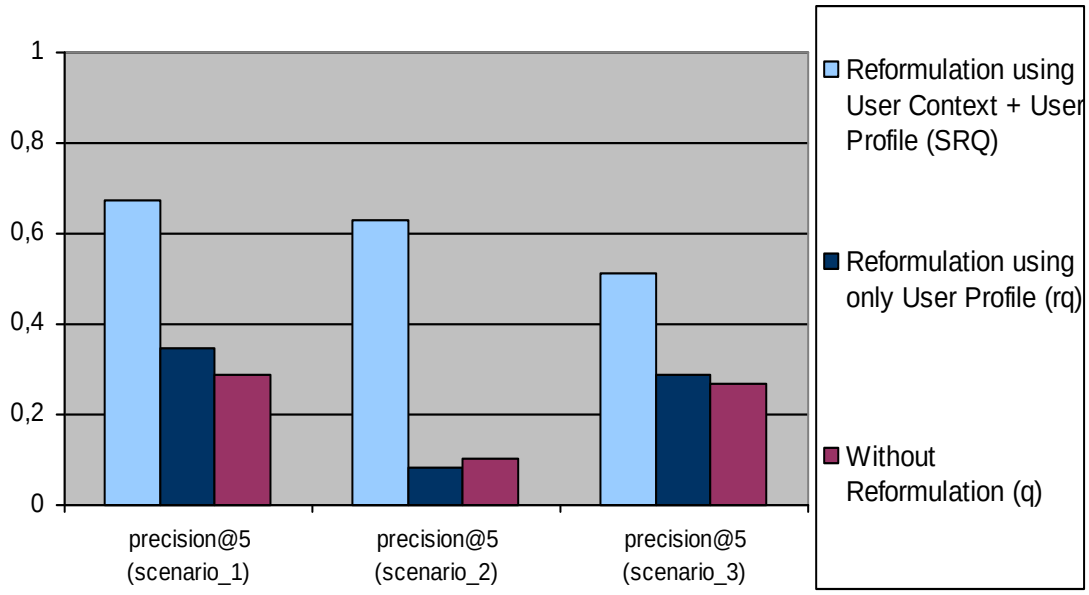


Figure 4.11. Comparison between the averages of Precision@5 over the experimental queries in the three cases.

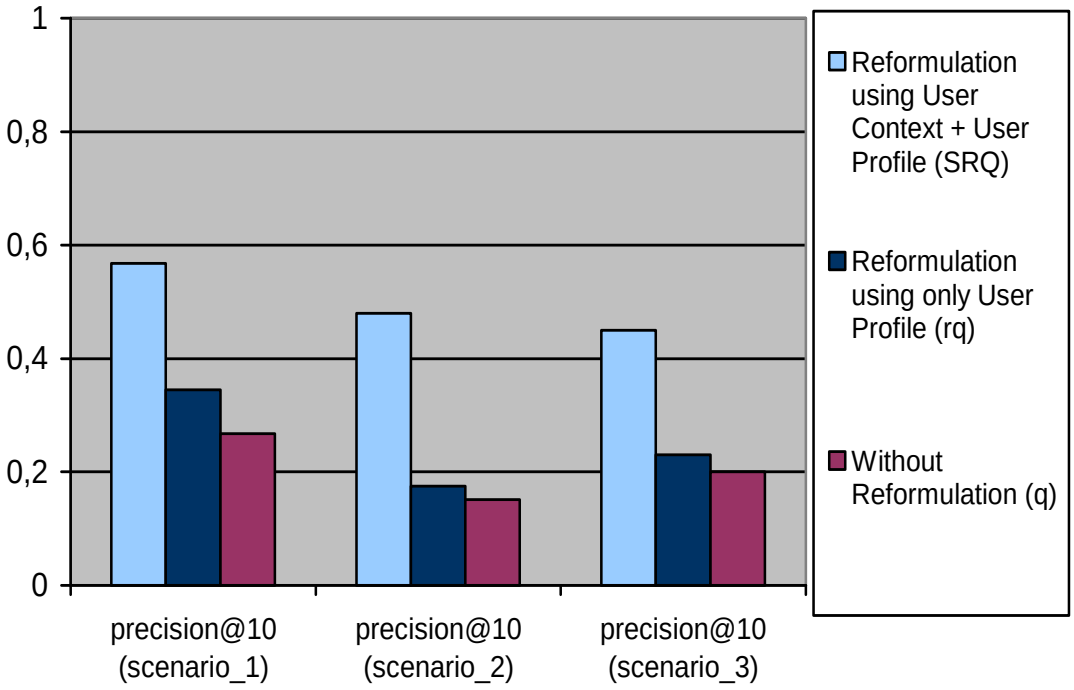


Figure 4.12. Comparison between the averages of Precision@10 over the experimental queries in the three cases.

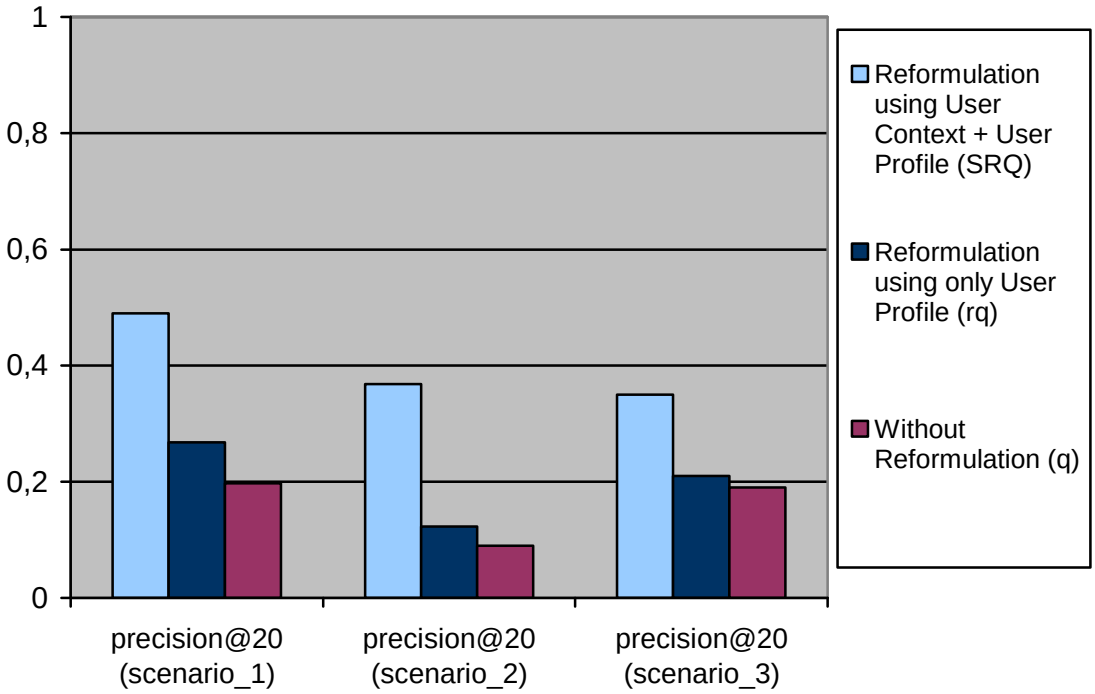


Figure 4.13. Comparison between the averages of Precision@20 over the experimental queries in the three cases.

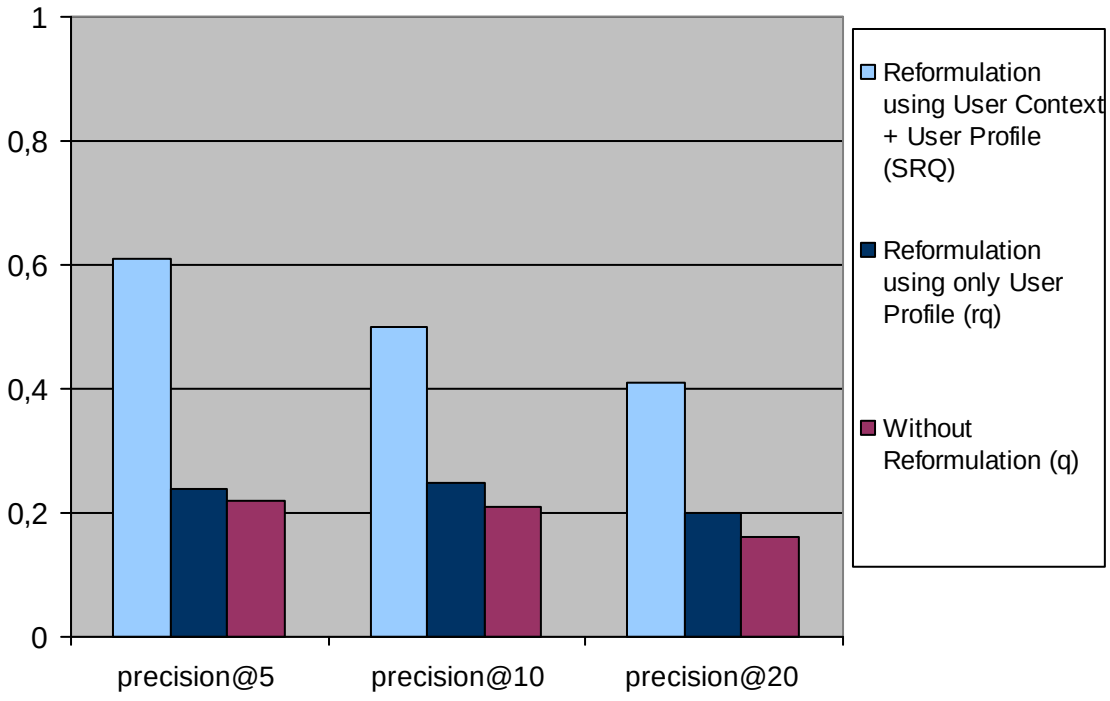


Figure 4.14. Comparison between the Precision@k averages of the different systems.

4.2.3.3 Dynamics

The third selected evaluation metric is the dynamics in query expansion; it measures the system capability of adapting to the changing needs of the user in relation to his/her current task and its states. Let q be an initial user query; our proposed system of query reformulation, based on a user task and his/her profile, returns different expansion terms to each task state and thus for the different search goals. Let $E_i^{(q)}$ and $E_j^{(q)}$ be the set of expansion terms which are proposed by our system for a query q at the two different task states *i* and *j*. Then we define the dynamics between the two states as follows:

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

For example, to calculate the dynamics in query expansion terms of the two states S_1, S_2 in the first proposed scenario (*shopping*), 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 6 terms, there are 4 common terms, and thus the similarity between these two states is 4/6, and the dynamics will be:

$$\delta^{(srq)}(s_1, s_2) = 1 - Sim\,(E_{s_1}^{(srq)}, E_{s_2}^{(srq)}) = 1 - (\frac{4}{6}) = 0.33$$

Another example to calculate the dynamics in query expansion terms of the two states S_1, S_2 in the second proposed scenario (*travel*):

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

In the same method we can calculate the dynamics in query expansion terms of the other states and for the three experimental scenarios. Table 4.5 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 Table 4.5 that our proposed system has a small dynamics in the expansion terms among the states of the simple tasks, such as scenario 1 and scenario 3, and it has a high dynamics in expansion terms among the states of the complex tasks, such as task in scenario 2. Figure 4.15 shows a comparison between the averages of the dynamics in query expansion terms over the experimental queries in the three previous proposed scenarios.

Thus our proposed framework is able to adapt to the changing needs of the users and generate expansion terms dynamically.

Table 4.5. Average dynamics in query expansion terms at the experimental scenarios.

ueries by 10 different users for each scenario)	Expansion terms $E^{(q)}$ for each state.	Dynamics between two successive states	Average Dynamics
Scenario1: Buy forks by user living in Paris... ex: q = ‘‘*buying forks”*	*Buying, forks, Kitchenware, eating, information*	$\delta^{(q)}(1,2)=0.33$	0.42
	Buying, forks, Kitchenware, eating, models	$\delta^{(q)}(2,3)=0.33$	
	Buying, forks, Kitchenware, eating, price	$\delta^{(q)}(3,4)=0.5$	
	Buying, forks, Kitchenware, eating, shop, Paris, 75014	$\delta^{(q)}(4,1)=0.5$	
Scenario2: Trip to the Toulouse city… ex: q= *“Tourism in Toulouse”.*	*Tourism, Toulouse, book, Flight, Ticket, Inexpensive*	$\delta^{(q)}(1,2)=0.78$	0.74
	Tourism, Toulouse, hotel, 2 star, single	$\delta^{(q)}(2,3)=0.78$	
	Tourism, Toulouse, Monuments, Weather, plan, Metro	$\delta^{(q)}(3,4)=0.8$	
	Tourism, Toulouse, Restaurant, Food, Italian, Vegetarian	$\delta^{(q)}(4,5)=0.71$	
	Tourism, Toulouse, Photos	$\delta^{(q)}(5,6)=0.6$	
	Tourism, Toulouse, News, Weather	$\delta^{(q)}(6,1)=0.75$	

Scenario3: Searches places to drink mate in the city centre of Paris… Ex: q= "*drinking mate*".	*Drinking, mate, tea, beverage information*	$\delta^{(q)}(1,2)=0.5$	0.5
	Drinking, mate, tea, beverage, Address, Paris, 75014	$\delta^{(q)}(2,1)=0.5$	

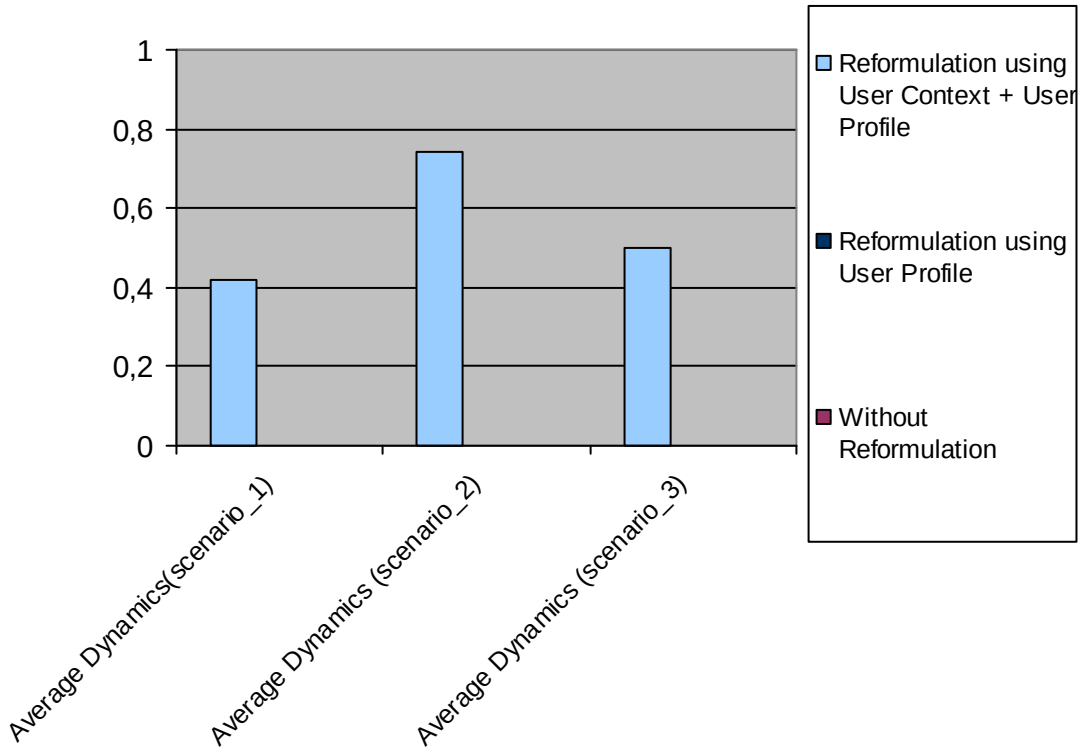


Figure 4.15. The average dynamics in query expansion terms for our system in the three experimental scenarios.

4.2.4 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 can not determine a specific ideal number of expansion terms.

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

Chapter 5

General Conclusion and Perspectives

5.1 General Conclusion

The system, presented in this thesis, offers new approach to help a user in an information search. 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 thesis 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.

We have constructed a general architecture that combines several models: task model, user profile model and SRQ model. And we have constructed theoretical 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 does not capture the semantic relationships between concepts and the semantic knowledge does not 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 model 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. From these 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.

5.2 Thesis's Contributions

- Combine knowledge about query (two types of knowledge, linguistic knowledge by using WordNet and semantic knowledge by using ODP taxonomy) and knowledge about user (user profile and user's task context) into a single framework in order to reformulate the user query, and thus generate SRQ (state reformulated query) that orientate the search to the relevant results, for that we construct a general architecture that combines the several models for query expansion: Task model, User profile model and SRQ model.
- The user context, in this thesis, is defined as the user's current task, its changes over time and its states. To detect and describe the task performed by the user when he/she submits his/her query to the information retrieval system, we propose a task model which depends on study questionnaires, which were used to elicit tasks that were expected to be of interest to subjects during the study, and defining concept vectors for the main tasks including their states. UML activity diagram is used to represent the user's current task.
- Our proposed approach involves new methodology to retrieve query-related elements from the ontological user profile which is constructed from existing concepts in domain ontology (for example ODP taxonomy). This methodology has been applied successfully to retrieve information from the semi-structured data.
- 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.

5.3 Thesis's Limitations

Detect implicitly the transition between the task states:

Here, the limits or the disadvantages of our system can be summarized by two points: in one hand, the automatic detection of the actual task state, and on the other hand, the transition to the next state after completing the actual state. The ideal system must perform these two points implicitly without any feedback from the user. But in our system we need a feedback from the user to detect the actual state of his/her task and to know when he/she will transfer to the next state of the current task.

In our system, according to our assumption, we have defined, to each user profile, UML Activity diagrams for the main pre-defined tasks, one UML graph for each main task including the pre-defined task states. But the problem, here, is how can we detect implicitly the actual state from the UML Activity diagram? Of course we are in need of a user feedback. In our platform, after the user's query is submitted, the related task is assigned automatically to the user query and a set of SRQ (State Reformulated Queries), related to each state, are presented to the user. We suppose that the user chooses his/her relevant reformulated query at his/her actual state from the list of SRQ, this is the first needed user feedback. At this point, the user can browse the returned search results which are appropriate to the state that has chosen. Next, he/she clicks on a button "Next", in our interface, to be transferred directly to the next state in the UML diagram; this is the second needed user feedback.

To overcome the two previous limits without the user feedback, we need a contextual model that can follow the UML activity diagram to present the appropriate query SRQ to the user, this contextual model can use contextual information may come automatically from various sources such as the user's schedule, sensors, entities that interact with the user, in order to detect the task state changes and the transition between the states. Also we can use an observer of user activities that is executed in the user machine.

Limits of our experiments:

Here, the limits or the disadvantages of our experiments are 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.

5.4 Perspectives

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 (Brezillon, 2005) instead of UML activity diagram to represent the user's current task. In our future work we plan to use this contextual graph.

Also we can use GOLOG, which is a logic programming language like SWRL and Prolog, to find the action compatible to the current task state during the implementation of the application (In the runtime).

For example: `If` S_1 `then action A else if` S_2 `then action B`

`Else action C`

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.

Our Publications

Asfari O., Doan B-L., Bourda Y., Sansonnet J-P., *Personalized Access to Contextual Information by using an Assistant for Query Reformulation.* IN: IARIA Journal, IntSys11v4n34, 2011, (Submitted).

Asfari O., Doan B-L., Bourda Y., Sansonnet J-P., *Context-based Hybrid Method for User Query Expansion*. IN: Proceedings of the fourth international conference on Advances in Semantic Processing, SMAPRO 2010. Italy, Florence, 2010. **Best Paper Award IN SEMAPRO 2010.** (Invitation to submit an extended version in IARIA journal: www.iariajournals.org).

Asfari O., Doan B-L., Bourda Y., Sansonnet J-P., *A Context-Based Model for Web Query Reformulation.* IN: Proceedings of the international conference on Knowledge Discovery and Information Retrieval, KDIR 2010. Spain, Valencia, 2010.

Asfari O., Doan B-L., Bourda Y., Sansonnet J-P., *Improving User Query Processing Based on User Profile and Task Context.* IN: Proceedings of the 2010 international Conference on Semantic Web & Web Services. Las Vegas, Nevada, USA, 2010.

Asfari O., Doan B-L., Bourda Y., Sansonnet J-P., *Personalized access to information by query reformulation based on the state of the current task and user profile.* IN: 2009 IEEE The third international conference on Advances in Semantic Processing, SEMAPRO 2009. Sliema, Malta, 2009.

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

References

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

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

Ahn Jae-wook and Brusilovsky, P., “Envisioning user models for adaptive visualization”, IN: Visual Analytics Science and Technology, pp. 175-176 VAST'08, 2008.

Baeza-Yates, R. and Ribeiro-Neto, B., eds. “Modern Information Retrieval”. IN: Addison Wesley Longman Publishing Co. Inc., 1999.

Baeza-Yates, R., Hurtado, C., Mendoza, M., “Query recommendation using query logs in search engines”. IN: EDBT ’04, 588-596, 2004.

Balbo, S., Ozkan, N., Paris, C., “Choosing the right task modelling notation: a taxonomy”. IN: Chapter 4, this volume, 2002.

Bauer, T. and Leake, D., “Real time user context modeling for information retrieval agents”. IN: CIKM '01: Proceedings of the tenth international conference on Information and knowledge management. ACM, pages 568-570, Atlante, Georgia, USA, 2001.

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

Billsus, D., Hilbert, D., Maynes-Aminzade, D., “Improving proactive information systems”. IN: IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces. ACM, pages 159-166, San Diego, California, USA, 2005.

Bittner, K., Spence, I., “Use Case Modeling”. IN: book, Addison Wesley Professional, 2-3. ISBN 0-201-70913-9, 2002.

Bouchard, H. and Nie, J., "Modèles de langue appliqués à la recherche d'information contextuelle". IN: Proceedings of CORIA 2006 Conf en Recherche d'Information et Applications. pp. 213-224, Lyon, 2006.

Bottraud, J.C., Bisson, G., Bruandet, M.F, "Apprentissage de profils pour un agent de recherche d'information". IN : Actes de la Conférence Apprentissage (CAP'03), pp. 31-46, 1-2 juillet, Laval, 2003.

Bradley J. R., Starner T., "Remembrance Agent: A continuously running automated information retrieval system". IN: Proceedings of the 1st International Conference on The Practical Application of Intelligent Agents ad Multi Agent Technology (PAAM '96), pp487-495, 1996.

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

Brown, P.J., Bovey, J.D., Chen, X., "Context-Aware Applications: From the Laboratory to the Marketplace". IN: IEEE Personal Communications, 4(5) 58-64, 1997.

Broder, A., "A taxonomy of web search". IN: SIGIR Forum, 36(2): pages 3-10, 2002.

Brusilovsky, P., and Millán, E., "User Models for Adaptive Hypermedia and Adaptive Educational Systems". IN: P. Brusilovsky, A. Kobsa, and W. Nejdl (Eds.): The Adaptive Web, LNCS 4321, pp. 3-53, 2007. © Springer-Verlag Berlin Heidelberg 2007.

Byström, K., and Hansen, P., "Conceptual Framework for Tasks in Information Studies". IN: Journal of the American Society of Information Science and Technology, vol. 56, issue 10, 1050-1061, 2005.

Challam, V., Gauch, S., Chandramouli, A., "Contextual Search Using Ontology-Based User Profiles". IN: Conference RIAO2007, Pittsburgh PA, U.S.A. May 30-June 1, Paris, France, 2007.

Chirita, P. A, Nejdl, W., Paiu, R., Kohlschutter, Ch., "Using ODP Metadata to Personalize Search". IN: Proc. of the 28th Annual Int'l ACM SIGIR Conf., pp 178–185, Salvador, Brazil, 2005.

Chirita, P.-A., Firan, C. and Nejdl, W., "Summarizing local context to personalize global web search". IN: CIKM '06: Proceedings of the 15th ACM international

conference on Information and knowledge management. ACM, Arlington, Virginia, USA 287-296, 2006.

Cingil I., Dogac A., Azgin A.,"A broader approach to personalization" IN: Communications of the ACM, Vol. 43, No. 8, 2000.

Conesa, J., Storey, V.C., Sugumaran, V., "Using Semantic Knowledge to Improve Web Query Processing". IN: NLDB 2006, pp. 106 – 117, Springer-Verlag Berlin, 2006.

Cordon, O., Moya, F., Zarco, C., "Fuzzy logic and multiobjective evolutionary algorithms as soft computing tools for persistent query learning in text retrieval environments". IN: IEEE International Conference on Fuzzy Systems 2004, pages 571–576, Budapest, Hungary, 2004.

Daoud, M., Tamine, L., Duy Dinh, Boughanem, M ., "Contextual Query Classification For Personalizing Informational Search" IN: Web Information Systems Engineering, kerkennah Island, Sfax, Tunisia, 12/06/2009-14/06/2009, ACM, (en ligne), juin 2009.

Dey, AK., and Abowd, GD., "Toward a better understanding of context and context-awareness". IN: Workshop on the What, Who, Where, When, and How of Context-Awareness, 1999.

Dourish, P., "What We Talk About When We Talk About Context". IN: Personal and Ubiquitous Computing, 2007.

Dou, Z., Song, R., Wen, J., "A Large scale Evaluation and Analysis of Personalized Search Strategies". IN: WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.

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

Efthimiadis, E. N., "Query expansion". IN: M.E. Williams (Ed.), Annual Review of Information Systems and Technology, 31, pp. 121-187, Medford, NJ: Information Today, 1996.

Eirinaki, M., Vazirgiannis. M., "Web mining for web personalization". IN: ACM Trans. Internet Techn. 3(1): 1-27, 2003.

Felicissimo, C., Chopinaud, C., Briot, J-P., “Contextualizing normative open Multi-Agent Systems”, 2007.

Ferragina, P., Gulli, A., “A Personalized Search Engine Based on Web Snippet Hierarchical Clustering”. IN: WWW 2005, May 1014, 2005, Chiba, Japan.

Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G. and Ruppin, E., “Placing search in context: the concept revisited”. IN: World Wide Web, Hong Kong, pp 406-414, 2001.

Fowler, M., Scott, K., “UML Distilled Addison-Wesley 2000”. IN: Book, 2000.

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

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

Gauch, S., Speretta, M., Chandramouli, A., and Micarelli, A., “User Profiles for Personalized Information Access”. IN: The Adaptive Web: 54-89, 2007.

Gabrilovich, E., Broder, A., Fontoura, M., Joshi, A., Josifovski, V., Riedel, L., Zhang, T., “Classifying search queries using the Web as a source of knowledge”. IN: Transactions on the Web (TWEB) , Volume 3 Issue 2, April 2009.

Gowan J., “A multiple model approach to personalized information access”. IN: Master thesis in computer science, Faculty of science, Université de College Dublin, February, 2003.

Henzinger, M., Chang, B.-W., Milch, B., Brin, S., “Query-free news search”. IN: WWW'03 Proceedings of the 12th international conference on World Wide Web. ACM, Budapest, Hungary, 2003.

Hlaoua L., Boughanem M., “Towards Contextual and Structural Relevance Feedback in XML Retrieval” IN: workshop on Open Source Web Information Retrieval, compiègne, Michel Beigbeder, Wai Gen Yee (Eds.), ISBN:2-913923-19-4, p. 35-38, 2005.

Jameson, A., “Numerical Uncertainty Management in User and Student Modeling”, An Overview of Systems and Issues. IN: User Modeling and User-Adapted Interaction 5(3-4), 193-251, 1995.

Jansen, B. J., “Seeking and implementing automated assistance during the search process”. IN: Information Processing & Management, 41, pages 909-928, 2005.

Jansen, B.J., Spink, A., Bateman, J., Saracevic, T., “Real life information retrieval: a study of user queries on the Web”. IN: SIGIR Forum 32(1): 5-17, 1998.

Jansen, B. J., “A software agent for performance improvement of existing information retrieval systems”. IN: 1999 International ACM Conference on Intelligent User Interfaces. Los Angeles, CA, 122-123, 1999.

Järvelin, K. and Ingwersen, P., “Information seeking research needs extension towards tasks and technology”. IN: Information Research, 10 (1) paper 212 [Available at http://InformationR.net/ir/10-1/paper212.html], 2004.

Kelly, D. and Teevan, J., “Implicit Feedback for Inferring User Preference: A Bibliography”. IN: SIGIR Forum 32(2): pp.18-28, 2003.

Kellar, M., Watters, K., and Shepherd, M., “A Field Study Characterizing Web-based Information-Seeking Tasks”. IN: Journal of the American Society for Information Science and Technology, vol. 58, issue 7, 999-1018, 2007.

Kim, W., ”Personalization: Definition, Status, and Challenges Ahead”. IN: ETH Zurich, Chair of Software Engineering JOT, ©2002, Vol. 1, No. 1, 2002.

Kim, H. and Chan, P. K., “Personalized ranking of search results with learned user interest hierarchies from bookmarks”. IN: Proc. of the 7th WEBKDD workshop on Knowledge Discovery from the Web (WEBKDD’05), pages 32–43, Chicago, Illinois, USA, 2005.

Kobsa. A., “Generic User modeling Systems”. IN: P. Brusilovsky, A. Kobsa, and W. Nejdl (Eds.): The Adaptive Web, LNCS 4321, pp. 136 – 154, 2007. © Springer-Verlag Berlin Heidelberg 2007.

Koutrika, G. and Ioannidis, Y. E., “A Unified User Profile Framework for Query Disambiguation and Personalization”. IN: PIA 2005, Proceedings of Workshop on New Technologies for Personalized Information Access, 2005.

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

Kofod-Petersen, A. and Cassens, J., “Using Activity Theory to Model Context Awarenessa”. IN: American Association for Artificial Intelligence, Berlin, 2006.

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

Kruger., A., Baus, J., Heckmann, D., Kruppa, M., “Adaptive Mobile Guides”. IN: The Adaptive Web, P. Brusilovsky, A. Kobsa, and W. Nejdl (Eds.): LNCS 4321, pp. 521–549, 2007. Springer-Verlag Berlin, Heidelberg, 2007.

Kules, W., Wilson, M. L., Schraefel, M. C., and Shneiderman, B., “From Keyword Search to Exploration: How Result Visualization Aids Discovery on the Web”. IN: Technical Report, School of Electronics and Computer Science, University of Southampton, 2008.

Kullback, S., Keegel, J.C., Kullback, J.H., “Topics in statistical information theory”. IN: Springer-Verlag in Berlin, New York , 1987.

Lafferty, J., Zhai, C., “Language models, query models, and risk minimization for information retrieval”. IN: SIGIR’01, Proceedings of 24th ACM International conference on research and development in information retrieval, pp. 111-119, New-York: ACM, 2001.

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

Li, L., Yang, Z., Wang, B., Kitsuregawa, M., “Dynamic Adaptation Strategies for Long-Term and Short-Term User Profile to Personalize Search”. IN: APWeb/WAIM 2007, pp. 228-240, 2007.

Liu, F., Yu, C., Meng, W., “Personalized Web Search For Improving Retrieval Effectiveness”. IN: IEEE Transactions on Knowledge and Data Engineering 16(1), pages 28-40, 2004.

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

Martin, I. and Jose, J. M., Fetch, “A personalized information retrieval tool”. IN: RIAO2004: Proceedings of the 8th Recherche d'Information Assiste par

Ordinateur (computer assisted information retrieval), Avignon, France, pp. 405-419, 2004.

Maes, P., “Agents that reduce work and information overload”. IN: Comm. of the ACM 37 (7), pages 31-40, 1994.

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

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

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

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

Paris, C., Lu, S., Linden, K.V., “Environments for the Construction and Use of Task Models”. IN: The Handbook of Task Analysis for Human-Computer Interaction. Edited by Prof. Dan Diaper & Prof. Neville Stanton. Lawrence Erlbaum Associates. pp 467-482, 2004.

Qiu, F., and Cho, J., “Automatic identification of user interest for personalized search”. IN: Proc. of the 15th Int’l Conf. on World Wide Web (WWW’06), pages 727–736, Edinburgh, Scotland, 2006.

Raskutti, B., Beitz, A. and Ward, B.: “A Feature-based Approach to Recommending Selections based on Past Preferences”. IN: User Modeling and User-Adapted Interaction 1997, 7(3), 179-218. 1997.

Rhodes B. J., Maes P, “Just-in-time Information Retrieval Agents”. IN: IBM Systems Journal, vol. 39 N°3 & 4, 2000.

Rumbaugh, J., Jacobsen, I., Booch, G., “The Unified Modelling Language Reference Manual”. IN: Reading, Mass.: Addison-Wesley, 1999.

Ryan, N., Pascoe, J., Morse, D., “Enhanced Reality Fieldwork: the Context-Aware Archaeological Assistant”. IN: Gaffney,V., van Leusen, M., Exxon, S. (eds.) Computer Applications in Archaeology, 1997.

Sauvagnat K., Boughanem M., Chrisment C., “Answering content and structure-based queries on XML documents using relevance propagation”, IN: Information Systems, HElsevierH, Numéro spécial Special Issue SPIRE 2004, V. 31, p. 621-635, janvier 2006.

Schickel-Zuber, V., and Faltings, B., “Inferring User’s Preferences using Ontologies”. IN: AAAI'06 proceedings of the 21st national conference on American Association for Artificial intelligence - Volume 2, 2006.

Schilit, B., Theimer, M., “Disseminating Active Map Information to Mobile Hosts”. IN: IEEE Network, 8(5) 22-32, 1994.

Shen, X., Tan, B., Zhai, C., “Implicit User Modeling for Personalized Search”. IN: CIKM’05, Bremen, Germany, October 31–November 5, 2005. (a).

Shen, X., Tan, B., Zhai, C., “Context-sensitive information retrieval using implicit feedback”. IN: SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, pp. 43-50, Salvador, Brazil, 2005. (b).

Sieg, A., Mobasher, B., and Burke, R., “Representing context in web search with ontological user profiles”. IN: Proceedings of the 6th International Conference on Modeling and Using Context, Denmark, August 2007. (a).

Sieg, A., Mobasher, B., Burke, R., “Learning Ontology-Based User Profiles: A Semantic Approach to Personalized Web Search”. IN: IEEE Intelligent Informatics Bulletin, Vol.8 No.1, November 2007. (b).

Sieg, A., Mobasher, B., Burke, R., “Ontological User Profiles for Representing Context in Web Search”. IN: 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. (c).

Snasel, V., Abraham, A., Owais, S., Platos, J., Kromer, P., “User Profiles Modeling in Information Retrieval System”, IN: Emergent Web Intelligence: Advanced

Information Retrieval, Y. Badr et al. (Eds.), Springer Verlag, London, ISBN 978-1-84996-073-1, pp. 169-198, 2010.

Speretta, M., Gauch, S., “Personalized search based on user search histories”. IN: Proc. of the IEEE / WIC / ACM Int’l Conf. on Web Intelligence (WI’05), pages 622–628, Compiegne, France, 2005.

Storey, V.C., Sugumaran, V., Burton-Jones, A., “The Role of User Profiles in Context-Aware Query Processing for the Semantic Web”. IN: Proceedings of the 9th International Conference on Applications of Natural Language to Information Systems, June 23-25, Manchester, UK, 2004.

Suhail, S., Owais, J., Kromer, P., and Snasel, V., “Query Optimization by Genetic Algorithms”. IN: Karel Richta, Vaclav Snasel, and Jaroslav Pokorny, editors, DATESO, volume 129 of CEUR Workshop Proceedings, pages 125–137, 2005.

Subercaze, J., Maret, P., Dang, N.M., Sasaki, K., “Context-aware applications using personal sensors”. IN: BodyNets ‘07: Proceedings of the ICST 2nd international conference on Body area networks, ICST ed. Florence. pp. 1-5. ISBN 978-963-06-2193-9, 2007.

Sugiyama, K., Hatano, K., Yoshikawa, M., “Adaptive Web Search Based on User Profile Constructed without Any Effort from Users”. IN: WWW 2004, New York, USA, May 17-22, 2004.

S. Rath, A., Devaurs, D., and Lindstaedt, S., “Studying the Factors Influencing Automatic User Task Detection on the Computer Desktop”, IN: Lecture Notes in Computer Science, 2010, Volume 6383, Sustaining TEL: From Innovation to Learning and Practice, Pages 292-307.

Tamine-Lechani L., Zemirli N., Bahsoun W., “Approche statistique pour la définition du profil d’un utilisateur de système de recherche d’information”. IN : Information - Interaction - Intelligence, Cépaduès Editions, Vol. 7, N. 1, 2007.

Tamine-Lechani, L., and Boughnem, Mohaned., “Accès personnalisé à l’information approches et techniques”. Equipe SIG/RFI, Rapport interne, 2005.

Tan, B., Shen, X., Zhai, C., “Mining long-term search history to improve search accuracy”. IN: KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, Philadelphia, PA, USA, 718-723, 2006.

Teevan, J., Dumais, S., Horvitz, E., “Personalizing Search via Automated Analysis of Interests and Activities”. IN: SIGIR 2005, pages 449-456, 2005.

Tedesco, R., Dolog, P., Wolfgang, N., Heidrun A., “Distributed Bayesian Networks for User Modeling”. IN: ELearn'2006: World Conference on E-Learning in Corporate, Government, Health Care, and Higher Education. Hawai, USA, October 2006.

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

Trajkova, J., Gauch, S., “Improving Ontology-Based User Profiles”. IN: RIAO 2004, pp. 380-390, 2004.

Tricot, A. and Nanard, J., “Un point sur la modélisation des tâches de recherche d'informations dans le domaine des hypermédias”. IN: Hypetextes et Hypermédia, J.P. Balpe, et al., Editors, Hermes. pp. 35-56, 1998.

Vallet, D., Castells, P., Fernández, M., Mylonas, P. and Avrithis, Y., “Personalized Content Retrieval in Context Using Ontological Knowledge”. IN: IEEE Transactions on Circuits and Systems for Video Technology 17(3), special issue on the convergence of knowledge engineering, semantics and signal processing in audiovisual information retrieval, pp. 336-346, March 2007.

Vidal, M.E., Raschid L., Marquez N., Cardenas M., Wu Y., “Query Rewriting in the Semantic Web”. IN: Proceedings of the 22nd International Conference on Data Engineering Workshops, ICDEW, USA, 2006.

W. White, R., Kelly, D., “A Study on the Effects of Personalization and Task Information on Implicit Feedback Performance”. IN: CIKM’06, USA, 2006.

Wakaki, H., Masada, T., Takasu, A., Adachi, J., “A New Measure for Query Disambiguation Using Term Co-occurrences”. IN: Lecture Notes Computer Science, NUMB 4224, pages 904-911, 2006.

Widyantoro, D. H., Ioerger, T. R., Yen, J., “Learning user interest dynamics with a three-descriptor representation”. IN: JASIST, 52(3):212–225, 2001.

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

Yu, Clement., “Personalized web search by mapping user queries to categories”. IN: CIKM’02 usa copyright 2002.

Zigoris, P. and Zhang, Y., “Bayesian adaptive user profiling with explicit & implicit feedback”. IN: CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management. 397-404, ACM, Arlington, Virginia, USA, 2006.

Zukerman, I., and W. Albrecht, D., “Predictive Statistical Models for User Modeling”. IN: school of Computer Science and Software Engineering, Monash University, Clayton, VICTORIA 3800, 2001.

Appendices

Appendix A: Details for the Scenarios Based Experiments

Name: **Age**:

<20	20-29	30-39	40-49	50-60	>60

Date:

Period: **Gender**:

M	F

Experience in computer science:

Novice	Medium	Expert

I. Experimental Scenarios (1)

Assume that you are a mobile user, you can surf the Internet by a PDA, you are in the city center of Paris, and you have a current task which is looking for a restaurant for the diner.

In your profile, you like *Italian* food, Like *Vegetarian* food, you don't like to drive too far, and you live in Paris France.

User context	User profile
- In the city center of Paris. - The time now is Saturday evening 7pm. - Surf the Internet by a PDA. - Current task: looking for a restaurant for the diner.	- Live in Paris France. - Don't like to drive too far. - Like *Italian* food. - Like *Vegetarian* food.

1. You enter your query in PDA at this context: for example ("*restaurant dinner*"),
2. You choose the relevant pages at your actual context from the Google presented results.
3. The system propose to you new reformulated query SRQ (*Restaurant + eating house+ Italian + vegetarian + Paris*) which contains related terms from your profile and your context, this query is submitted to the search engine, We present the first 20 results returned by our system "SRQoogle" to you, and you have to evaluate them.

II. Experimental Scenarios (2)

User context	User profile
- Organise personal trip in Toulouse. - You have 6 states: o S_1: *book a flight*. o S_2: *book a hotel*. o S_3: *Search for tourist information*.	- You live in Paris France. - You travel by airline and with inexpensive budget. - You prefer the 2 star hotels, and usually you book a single room, and you prefer wireless Internet in the room. - Like *Italian* food or *Vegetarian* food,

o S_4: *find a restaurant*. o S_5: *Tourist photos of Toulouse*. o S_6: *News*.	and your budget is not high (Inexpensive**).** - You like to visit the tourist monuments and. - You like to enjoy in the nightclubs.

1. You enter your query in PDA at this context: for example: q = {*trip Toulouse*}.
2. You choose the relevant pages at each task state from the Google presented results, knowing that there are 6 task states, (S_1, S_2, S_3, S_4, S_5, and S_6), and the relevant pages for each state must be evaluated.
3. The system propose to you new reformulated query SRQ which contains related terms from your profile and your context,

- S_1 (Book a flight): S_1RQ: {*trip Toulouse +" Airline"+ Book+ Ticket*}.
- S_2 (Book a hotel): S_2RQ: {*trip Toulouse + "hotel" +2 star +single*}.
- S_3 (Search for tourist information): S_3RQ: {trip *Toulouse+ "Monuments" + Weather + plan + Metro*}.
- S_4 (Find a restaurant): S_4RQ: {*trip Toulouse+ "restaurant" + Italian Cuisine + Vegetarian Food*}.
- S_5 (Tourist photos of Toulouse): S_5RQ: {*trip Toulouse + "Photos"*}.
- S_6 (News about Toulouse city): S_6RQ: {*trip Toulouse + "News"*}.

4. These queries are submitted to Google, we present to you the first 20 results returned by our system "SRQoogle" at each task state, and you have to evaluate them.

III. Experimental Scenarios (3)

User context	User profile
- Looking to buy a laptop. - You have 4 states: o S_1: Buzz about best laptop. o S_2: Choosing a laptop model and a shop with best price.	- Live in Paris 75014, France. - The favourite laptop model is: HP. - Your budget to buy a laptop is about 500 euros.

1. You enter your query in *PDA* at this context, for example: q = {Laptop}.
2. You choose the relevant pages at each task state (S_1, S_2) from the Google presented results. The relevant pages for each state must be evaluated.
3. The system propose to you new reformulated query SRQ which contains related terms from your profile and your context,
   - S_1 (buzz about best laptop): S_1RQ: {*laptop +" Models"*}.
   - S_2 (Choosing a laptop model and a shop with best price): S_2RQ: {*laptop + "shop" +HP + Price<500* + Paris}.
4. These queries are submitted to Google, we present the first 20 results returned by our system "SRQoogle" to you, and you have to evaluate them at each task state.

IV. Experimental Scenarios (4)

Assume that you are a mobile user, you can surf the Internet by a PDA, you was walking in the city center of Paris and you search place to drink mate, which is a kind of tea frequently drunk in South America.

User context	User profile
- Surf the Internet by a PDA. - Walking in the city center of Paris. - Search place to drink mate.	- You live in Paris 75014, France. - Your origin is from South America.

1. You enter your query at this context: for example ("*drinking mate*").
2. Now you have to evaluate the relevant pages at each contextual task state from the Google presented results. Knowing that there are two states (S_1: Information, S_2: Address) and you have to select the relevant results for each one.
3. The system propose to you new reformulated query SRQ for each task state that contains related terms from your profile and your context, this query is submitted to the search engine Google:
    - S_1: Information S_1RQ: *Drinking mate tea beverage" information"*.
    - S_2: Address S_2RQ: *Drinking mate tea beverage" address"+ Paris*.
4. We present the first 20 results to you, and you have to evaluate them.

V. Experimental Scenarios (5)

Assume that you have one task at hand that is looking to buy forks.

User context	User profile
- Looking to buy forks. - You have 4 states: o S_1: Information, o S_2: Model choice, o S_3: comparing prices, o S_4: choosing a shop	- Live in Paris 75014, France.

1. You submit your query at this context: for example (q=*"buying forks"*) or (q=*"forks").*
2. Now you have to evaluate the relevant pages at your actual task state from the Google presented results. Knowing that there are four states for this context (S_1: Information, S_2: Model choice, S_3: comparing prices, S_4: choosing a shop) and you have to select the relevant results for each one.
3. Our system "SRQoogle" can produce reformulated queries SRQ for the query q at each task state:
   - S_1 (Information), S_1RQ: {*buying forks*+ *Kitchenware eating* +*"information"*}.
   - S_2 (Model choice), S_2RQ: {*buying forks* + *Kitchenware eating* + *"models"*}.
   - S_3 (comparing prices), S_3RQ: {*buying forks*+ *Kitchenware eating* + "price"}.
   - S_4 (choosing a shop), S_4RQ: {*buying forks*+ *Kitchenware eating* +*"shop" Paris* 75014}.
4. The first 20 results of these queries by Google are presented to you, and you have to evaluate them at each task state.

Appendix B: GOLOG

We mentioned in the perspective that we can use GOLOG to find the action compatible to the current task state in the runtime. For that, we can provide, here, some details about GOLOC.

GOLOG is a high-level rule-based logic programming language, like SWRL and Prolog, based on top of the situation calculus, which is a first-order logic language for reasoning about action and change, GOLOG is an extended language of situation calculus for the specification and execution of complex actions in dynamic domains, but that didn't take into account information-gathering actions.

GOLOG can be queried, reasoned at runtime about the state of the world and consider the effects of possible actions before behavior is chosen. For example:

• Desirable (A, S) specifying that action A is desirable in situation S

• Possible (A, S) specifying that action A is possible in situation S

GOLOG allow specifying the kind of paths users may follow in url-space. In the content presented to the user, whole blocks can be derived by logical reasoning about the current user profile and common knowledge about the current domain.

A knowledge-based agent can be written in GOLOG. It communicates with the world using a predefined network protocol and a restricted set of interfaces.

Applications of GOLOG: Robotics, Artificial Intelligence, Mechanical Devices, Modeling and Simulation.

In fact we can not represent the task and its changes by GOLOG (like as the "contextual graph" and the UML Activity diagram) because it is a logic programming language. But it is useful to find the action compatible to the current situation during the implementation of the application.

Ex.: If S_1 (task state1) then action A

Else if S_2 (task state2) then action B

Else action C;