Applying the Broadway Recommendation Computation Approach for Implementing a Query Refinement Service in the CBKB Meta Search Engine

Rushed Kanawati†, Michel Jaczynski†, Brigitte Trousse†, Jean-Marc Andreoli‡

† INRIA, Action AID, 2004 Route des lucioles, 06902 Sophia Antipolis
‡ XRCE - 6, chemin de Maupertuis, 38240 Meylan France
{Prenom.Nom}@sophia.inria.fr et http://www.inria.fr/aid/
Jean-Marc.Andreoli@xrce.xerox.com et http://www.xrce.xerox.com/research/ct/

Résumé

This work reports on the applicability of the Broadway recommendation computation approach for implementing a query refinement (QR) service in the context of a distributed information retrieval and gathering system. The Broadway recommendation approach is based on the following hypothesis: recommend to one user items or actions that have *satisfied* users that have *behaved similarly* to her/him. The case-based reasoning (CBR) paradigm is proposed to implement Broadway-based recommenders. We show how to integrate a query refinement recommender in the context CBKB, a meta-search engine developped at XRCE.

1 Introduction

This work adresses the problem of query refinement (QR) in the context of a meta-search engine. Metasearch engines are tools that gather data from heterogeneous on-line repositories while presenting users with a unified interface for information sources they relay on. Examples are MetaCrawler [19], SavvySearch [8] and CBKB [1]. Different QR approaches are proposed in the Information Retrieval (IR) field. These approaches include the use of thesaurus [13], user profiles [14], the current query results [23], past query results [10], relevance feedback [20], or concept recall [22]. In this work we propose a new QR approach, called *Broadway-QR* where query refinement recommendations are made by reusing past query refinement sessions. The approach aims at enabling the reuse of successful past query refinements. To learn form past experiences Broadway-QR relies on using Case-Based Reasoning (CBR) [18], a technology for problem solving based on reusing past experiences. Broadway-QR is a specialization of a general recommendation computation approach, called *Broadway*, developed at INRIA [11, 21]. The Broadway recommendation approach is based on the following hypothesis: if two users *behave* in the same or in similar ways, then they might be seeking the same goal. We report here on the design and the development of a prototype of a searching engine, called BeCBKB that consists on adding a Broadway-based QR service to the metasearch engine CBKB. [1].

The reminder of this paper is organized as follows. Next in section 2. we give a general description of the Broadway-QR approach. We start by describing the general Broadway recommendation approach, then we show how to adapt this approach to address the problem of query refinement. An example of applying Broadway-QR is also discussed. Comparisons with related work is carried out in section 3.

2 The Broadway recommendation approach

2.1 General description

Recommender systems are traditionally partitioned into two main families: *content-based* recommenders and *collaborative filtering* ones [17]. Recommenders of the first type recommend items or actions to the user depending on an evaluation of the user own past actions, while those belonging to the second family recommend to a user items positively evaluated by other *similar* users. The Broadway approach implements a relatively new recommendation approach where the system recommends to a user what other users that have *behaved similarly* to that user, had positively evaluated. Other systems in the IR field rely on using user behavior similarity as a basis for recommendation computation [7, 24]. However, the Broadway approach has the particularity to model user behavior by variable observations rather than by matching user actions to a pre-defined behavior model [11]. CBR techniques are used to implement the Broadway approach. In this approach the problem part of a case is composed of the user behavior while the solution gives the result of that behavior.

The use of CBR technology to implement the Broadway approach needs to deal with cases with temporal indices or what is called in [11] cases with *time extended situations.* CBR*TOOLS, an object oriented framework for CBR applications development allows the manipulation of such cases by implementing a time-extended situation indexing model. Following this indexing scheme, user behavior is modeled by a set of *time series* that hold observations of the evolution with time of variables that are said to be relevant to describe the user behavior. The choice of these variables depends obviously on the application domain and the type of recommendation to compute. Time series are grouped into records that have an application-dependent semantic (i.e. a record could contains the evolution of time series within a searching session). The use of the time-extended situation indexing model involves hence the following phases: 1) identification of variables that model the user behavior, 2) definition of records and associated semantics, 3) definition of case structure to use: the problem and the solution part, and 4) definition of reasoning steps: how to retrieve cases, how to reuse solutions and how to learn new experiences.

Next we discuss the application of the Broadway approach to specifying a QR-recommender to be integrated in the BeCBKB system. BeCBKB implements a new user interface that allows users to interact with the CBKB server and it integrates a QR service.

2.2 Specification of a Broadway-based QR-Recommender

Since user behavior modeling is a central issue in the implementation of the Broadway approach, we need first to describe how users interact with the searching engine before describing how to model their behavior. The BeCBKB system being based on using the CBKB search engine we first describe how to use the CBKB system. Formulating a query using the CBKB system consists of two steps:

1. *Selecting the query domain* e.g. "operas" or "book/articles". The system forwards the query to sources relevant to that domain. The relevancey of a source to a domain is determined by consulting a *static* pre-configured hierarchy where nodes that represent domains are connected to leaves that are the corresponding relevant information sources.
2. *Defining the search pattern.* The search pattern is defined by constraints expressed via attribute/value pairs. For example one might search for "book/article" written by a specific `author` such as "Lamport". Searchable attributes depend on the query domain. For example `author, Title, Keywords` are searchable attributes in the "book/article" domain while `composer` is a searchable attribute in the "operas" domain. A constraint could be composed of a boolean expression, the output of a tool (e.g. stemming tool, automatic translation tool,..) or the result of a sub-query.

Once submitted, the query is sent to the CBKB server and processed further. Results are returned in an asynchronous way realizing hence a *push* information delivery method. In BeCBKB we have considered a sub set of CBKB functions. Essentially, the user is enabled to formulate only simple queries and one query can be active at a given instant. Concerning the query domain definition we propose to separate the choice of the domain (or topics) from the choice of information sources. Our goal is to be able to learn associations between domains and sources rather than relying on a static association scheme as it is currently the case in CBKB. Finally, we've added a button allowing the user to ask for advice when formulating a query. A set of buttons that allow the user to evaluate results returned by the CBKB server is also added. A 4-level evaluation scale is proposed. These evaluations permit us to have a feedback from the user about the relevance of returned documents. The user may ask for advice once s/he achieved at least one query refinement process in the searching session (i.e. at least 2 queries are submitted). The advice computation is made in a synchronous way. Upon receiving an advice request, the recommender searches in its memory for similar cases that can be used for proposing the most adequate query refinement action. If the user selects an advice, the system will automatically submit the advised query to the CBKB server.

2.2.1 Modeling the user behavior

A set of 8 behavioral variables (noted VB_i) have been chosen to model the behavior of users involved in query refinement process. As shown on figure 1 we group these variables into three sets:

Query configuration variables: Three variables are used to specify a query:

- `VB1, QueryTopics` holds the list of *domains* (or topics) in which the user is searching for information (ex. “Cpmputer science”).
- `VB2, QueryInfoSource` holds the set of information sources to which the current query is addressed (ex. “ACM publications”).

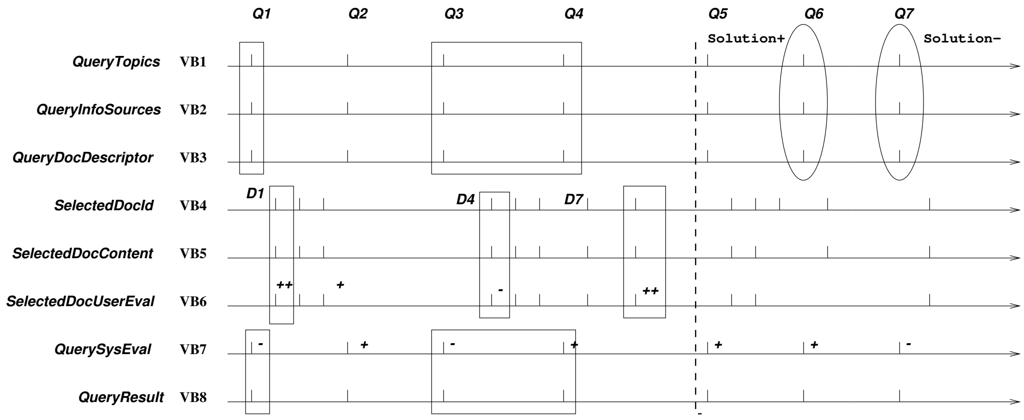

FIG. 1 – *Record and case structure.*

- `VB3, QueryDocDescriptor` is the set of search patterns specified for the current query. We consider here only simple search patterns that consist of specifying that a document attribute *contains* a user-defined list of keywords (ex . "CSCW, Concurrency Control").

Raw query results: Three variables are used to capture the user behavior in exploiting returned results:

- `VB4, SelectedDocId` is the identifier of a document selected by the user from the set of provided answers.
- `VB5, SelectedDocContent` is a key word list describing the content of a selected document.
- `VB6, SelectedDocUserEval` is a user-provided evaluation of the relevance of a selected document. A 4-level scale is provided for this purpose : highly relevant, relevant, unrated and irrelevant.

Synthetic query results: This is a set of variables that summarizes the result of a query. They are computed by analyzing the real values of the set of raw query result variables. Three variables are used:

- `VB7, QuerySysEval` measures the system estimation of the user satisfaction from a query. Two main query effectivness measurements are frequently used in IR filed : *precision* and *recall* [13]. Precision is the ratio of relevant documents retrieved to the total number of retrieved documents, while recall is defined as the ratio of the number of relevant documents retrieved to the total number of relevant documents in the whole document collection. The precision Q_p of a query is computed according to the following formula:

$$Q_p = \frac{\sum_{i=-1}^{2} w_i \times N_i}{N \times \sum_{i=-1}^{2} w_i}$$

  where N is the total number of documents returned in response to the query, i denotes the number of possible ratings and N_i denotes the number of documents that have the rating i. For example N_2 denotes the number of highly relevant documents (retained documents) and w_i denotes the attributed weight for the rating i.
  Computing the exact query recall Q_r is practically impossible since we can not estimate the number of relevant documents that are not retrieved by the query. Thus, we propose a simple estimation of the query recall as follows: if the number of returned documents is bigger than a given threshold N_{min} then the query recall is set to 1 otherwise the query recall is set to a predefined value LR where $0 < LR < 1$. We define the user satisfaction of a query to be the product of the query recall and the query precision: $VB7 = Q_p \times Q_r$
- `VB8, QueryResult` is composed of two lists, the first (respectively the second) one contains the most relevant terms that occur in the description of retained (respectively excluded) documents.

2.2.2 Records and context

The evolution of the user behavior variables described in the previous section is represented by a set of *event-driven time series* [11], one for each variable. Defining an event-driven time series needs to determine the set of events which defines the instant of measuring the variable values. Three types of events are used as the time step: submitting a query, selecting a document and evaluating a document. The observation of a set of time series is segmented into *records*. A record groups time series that are observed during a defined period of time. In our application we create a new record for each search session. Hence a record starts with session initialization and ends when the user terminates the session. Both events starting and ending a search session are made explicitly by the user. For each record (i.e. a search session) we define a *context*. The context represents a synthesis of the session: type of submitted queries, a summery of obtained results and a measure of the overall satisfaction of the user from the whole session. A set of seven variables (noted VC_i) have been identified to capture a search session context:

- `VC1, SessionTopics` is a set containing the *broad topics* addressed in the session. According to the used query model, the user has the ability to define a list of topics for each submitted query (see variable `VB1`). These topics are chosen from a predefined static hierarchy of topics H. We define the `SessionTopic` to be the set of first-level topics that are either in one of the QueryTopic list in the session or are a generalization of a topic that is used for at least in one query in the session. More formally, let $H(i)$ denotes the set of topics of level i in the defined topic hierarchy. $H(0)$ is the hierarchy root and $H(1)$ is the set of first level topics. Based on the hierarchical structure we say that topic T_a is generalization of T_b (denoted $T_b \Rightarrow T_a$) if T_b is a sub topic of T_a. Let Qt_i be the value of the variable `VB1: QueryTopics` of query i in the session. We define Qt to be the union of all query topics on the whole session: $Qt = \cup_{i=1}^{N} Qt_i$ where N is the number of queries in the session. The session topics list is then defined as follows: $VC1 = \{t_i : t_i \in H(1) \text{ or } t_i \in QT \text{ and } \exists t_j \in QT : t_j \Rightarrow t_i\}$
- `VC2, SessionInfoSource` contains the most frequent information sources that contribute in providing positively rated documents in the session.
- `VC3, SessionDocContent` is a double list of keywords ; the first contains the most relevant terms extracted form retained documents, the second is composed of the most relevant terms extracted from excluded documents.
- `VC4, SessionResult` is the most relevant terms that occur in the representations of retained document at the end of the session.
- `VC5, StartQuery` is the configuration of the first query in the session.
- `VC6, SessionSysEval` is a binary variable that is set to 1 if the searching session is successful and to 0 otherwise. A session is said to be successful if the user is satisfied by the last query in the session.
- `VC7, SessionLength` the total number of queries submitted in the session.

2.2.3 Case representation

A case references a precise experience in a search session. In general a case is composed of three main parts:

1. *Problem*: it is composed of a set of indexes that describe the state of the world when the case has occurred. These indexes help to retrieve the most useful source cases that could provide a solution for the current target case. Our query refinement recommender is based on the idea of comparing user's behavior and recommend to one user what other that have a similar behavior have made when formulating similar queries. Thus the problem part of a case is composed of a time extended situation. Such a situation is formed of two parts: *an instantaneous part* and *behavioral part* [11]. The instantaneous part is composed of the session context from which the case is extracted, as well as an integer that gives the rank of the last formulated query just before the reference instant. The behavioral part is composed of:
   (a) The p last query configurations observed on the behavioral variables VB_1,VB_2,VB_3,VB_7,VB_8.
   (b) The set of retained and excluded documents observed on the variables VB_4,VB_5,VB_6.
   (c) The last query configurations that has lead to a failure; that's to say configurations that have a negative $QuerySysEval$ value. These queries are also observed on the same set of variables as the last p queries. Note that the intersection of such identified query configurations and the set of the last p query configurations could not be empty.

   The first item of a behavioral part exists by definition in any case. According to the used indexing model [11] this item is called the *case restriction*. The other two items might be empty in case. These are called *elementary behaviors*.
2. *Solution*: Each case may contain two types of recommendation: a positive recommendation and a negative one. The former recommends actions that, according to the *case experience* would improve user satisfaction while a negative recommendation shows a query configuration that has lead to a failure. In both cases, a recommendation is composed of a query configuration, that's to say a set of values for the first three variables that define a query VB_1,VB_2 and VB_3. A negative recommendation is taken to be the first negatively evaluated query configuration after the reference instant. While the positive recommendation is computed by choosing the last positively evaluated query configuration which all predecessors are also positively evaluated.
3. *Evaluation*: This part contains an index that measures the confidence in the positive solution proposed by the case. A confidence measure is a real

number between 0 and 1. The value 0 means that the proposed solution is not dependable while the value 1 means that the solution is largely confirmed as a good one. At the case extraction, the confidence index is set to a median value, let's say: 0.5. The more the solution proposed by the case is approved by the users, the more the confidence in the solution is increased. A solution is approved by a user if the user satisfaction of the resulting query after applying the recommendation is positive.

2.2.4 Reasoning steps

The reasoning follows the classical *CBR* cycle: search, reuse and learn phases. Next we describe the first two phases. Learning is ensured by saving into the record base records that have at least two queries (one QR step).

Searching: The purpose of the search phase is to retrieve from the system memory past cases which solutions fit the best the current target case. To do so, we measure the similarity between the problem part of the target case and the problems parts of cases saved in the system memory. The similarity between two users behaviors is taken to be an aggregation function of the similarities between variables used to model these behaviors. Next we describe the used similarity functions and then we describe how these functions are used for case retrieval.

Similarity measures: The following elementary similarity functions are used:

$Sim_{Topics}(T_1,T_2)$: Let T_1, T_2 be two lists of query topics. T_1 (respectively T_2) contains n (respectively m) elements. Without loosing generality, we suppose that $n \leq m$. The similarity between two simple elements (x,y) (i.e single topics) is computed by the function `ElemTopicSim(x, y)` defined as follows:

$$1 - \frac{h(x,MSCA(x,y)) + h(y,MSCA(x,y))}{h(x,root) + h(y,root)}$$

where $MSCA$ is the most specific topic common to x and y, and h returns the number of links between two nodes in the hierarchy of topics. The similarity function between two list of topics is then defined as follows:

$$S_{Topics}(T_1,T_2) = \frac{\sum_{i=1}^{n}(\frac{\sum_{j=1}^{m}(ElemTopicSim(t_i^1,t_j^2)}{m})}{n}$$

This similarity function is applied to the following variables: VB_1, VC_1.

$Sim_{Eval}(E_1,E_2)$: The similarity between two evaluations E_1,$E_2$is given by the matrix shown in the Table 1.

Evaluation	-1	0	1	2
-1	1	0.5	0	0
0	0.5	1	0.5	0.5
1	0	0.5	1	0.75
2	0	0.5	0.75	1

TAB. 1 - Query and document evaluation similarity matrix.

This is applied to both variables VB6, and VB7

$Sim_{Set}(x,y)$: we define a similarity function between two sets x and y as follows:

$$Sim_{Set}(x,y) = \frac{Card(x \cap y)}{Card(x \cup y)}$$

We use this simple similarity to compare the variables VB2, VB5, VB8, VB9, VC2, VC3 and VC4.

Aggregation functions should be defined to measure the similarity of values of composed variables : such as VB3, VB8, VC5, VC7, as well as for measuring a similarity between a set of variables. An aggregation function takes a set of n similarity values S_i (thus n is a real number between 0 and 1), and a set of n weights W_i,$i : 1 \rightarrow n$, where W_i represents the weight the elementary similarity number i should play in the aggregated value. The result of an aggregation function is a value between 0 and 1 that is computed as follows:

$$A = \frac{\sum_{i=1}^{n} W_i \times s_i}{\sum_{i=1}^{n} W_i}$$

The retrieval algorithm: The system memory is decomposed into two sub-memories: 1) *The record base* that contains all saved records that represent past searching sessions, and *the case base* that contains the set of already extracted cases. The case retrieval algorithm functions as follows: first the algorithm tries to retrieve cases from the case base. If no cases with *enough* similarity are found the algorithm tries to retrieve new cases by searching for matching cases in the record base. Matching cases are determined by applying the following steps:

1. *Step1. Filtering on the record context*: The goal is to determine the most useful sessions from which interesting lessons can be learned in order to guide the current query refinement process. A session is potentially useful if the similarity between the *session context* and the current session context is over a given threshold T_{ps}. The most similar sessions are identified at the end of this process. If no session is found the reasoning process is stopped. The similarity between the current session context and memorized sessions is defined as an aggregation similarities between the values of the following variables: VC1-5. A weight vector W_{SC}

is used for computing the aggregated value. We believe that both variables VC1 and VC5 should assigned the most important weights compared to the other variables. The set of retained sessions points to a set of cases that have been extracted from these sessions. These cases are first examined in order to identify useful cases following steps 2 and 3 described below. If no case that has a similarity over a given thresholdi= is found the system tries to extract new matching cases form records saved in the history of identified sessions. If no case could be retrieved form these sessions the reasoning process is stopped.

2. *Step2. Restriction filtering*: The set of cases resulting from the first step are compared with the target case by computing the similarity of the restriction variables: the p last configurations of queries. This requires the definition of a new aggregation function that aggregates similarities of values of the following variables : VB1-3, VB7-8. This function uses a weight vector W_q. We retain from this step the most useful cases.
3. *Step3. Elementary behavior filtering*: The set of cases returned by the previous step are further examined to search from cases that match the best the current case situation. This is done by chec-cking whether the retained cases have a similar elementary behavior as the target case. Elementary behaviors in a case are taken to be the set of query configurations that have lead to a failure as well as the set of both retained and excluded documents. The best matching cases are then fed to the reuse phase in order to determine the solution to present to the current case.

Reuse: The retrieval phase returns a set of cases, each proposing at least one type of recommendations: positive or negative. The goal of the reuse phase is to identify the most *reusable* positive solutions (positive recommendations). Negative solutions are used in order to make sure that a solution proposed by system would not lead to a *predictable* failure. For each proposed positive solution we compute what we call the solution cost. This is a metric that measures the distance that separates the proposed solution from the last query submitted in the current session (the session for which the recommendation is computed). The distance d between two queries Q_1,Q_2 is defined to be the number of actions required to modify the Q_1 to become identical to Q_2. For example, given two simple queries expressed as a set of keywords: Q_1 = (CSCW, Concurrency, Control), Q_2 =(Groupware, Coherency, Control). The distance between these two queries is $d(Q_1,Q_2) = 4$ since we need to remove two keywords form the Q_1 (CSCW, Concurrency) and to add two new other keywords (Groupware, Coherency) in order to transforms Q_1 into Q_2. The strategy we adopt for ranking obtained solutions is to propose first solutions that are close to the last submitted query in the session from which the target case is issued. This has the advantage of not disturbing the user by recommending new query configurations that are very different form her/his last submitted query. A similar strategy is also adopted in [22] where they prove that expanding queries by adding one single at a time is more efficient and more comprehensible by users than adding multiple terms.

2.3 Example

The purpose of this example is to illustrate the application of the Broadway recommendation approach for query refinement tasks. Tracing the exact computation of the QR-recommender as specified in (2.2) is complex. Therefore, we consider here a simplified model of user behavior where the following hypothesis are made:

1. Queries submitted by the user are composed of a list of keywords (as it is the case when using a simple web searching engine).
2. The query results are composed of two lists, the first contains the set of documents identifiers that has been judged relevant by the user while the second contains the set of irrelevant documents.
3. The user rates explicitly his/her satisfaction from the query.

The case structure is also simplified by applying the following hypothesis:

- The behavioral part of a case is reduced to the restriction part (see 2.2.4). This is taken to be the last two submitted queries (with their results and evaluations).
- A case defines only a positive solution. This is taken to be the **next** positively evaluated query configuration just after the query configuration at the considered time reference, if such a configuration exists. Otherwise, if the last query itself is positively evaluated then it is taken to be the solution. If neither of situation are verified then no solution is proposed from this case.

The similarity function Sim_{Set} is applied for computing similarities of the query configuration (since this is reduced to a set of keywords) and the query results. The similarity function Sim_{Eval} is used to compute similarities of query evaluations. A simple arithmetic average function is used for computing the similarity between two problem parts of two cases.

Let's consider the following scenario where two people are searching for scientific publications about: *Concurrency control protocols for groupware systems*. Figure 2 summarizes the searching session of the first user. Data handled in this example are obtained by submitting specified queries to the Yahoo search engine (http://www.yahoo.fr). Queries are submitted in a conjunctive form. Only the first 20 documents returned by Yahoo for each query are considered for query evaluation. Relevancy of documents are evaluated by the first author of this paper since he has a good knowledge about the fixed search theme. Evaluated documents are identified by their URLs.

Now the second user, that share the same information need as the first one, starts a searching session as illustrated on the figure 3.

After submitting two queries, that do contain keywords relevant to the information need theme, the user does not get relevant information. Now if this user asks for advice about how to refine her/his query to a Broadway-QR recommender. The recommender will function as follows:

First the recommender will compute the session context of the current searching session. A target case (C_{Target}) is built where only the problem part is known. According to the case structure we have defined the target case has a behavioral part that is composed of the sequence composed of the configurations, results and evaluations of both queries $Q_{2.1}$ and $Q_{2.2}$. The instantaneous part is composed of the current session context (see 2.2.3).

Now once the target case is built, the recommender will search in its memory for matching cases. The case memory being empty the reasoner checks the record base to search for sessions from which it can extract new cases. Case retrieval is made through two steps. First the system memory is searched in order to find cases that have a similar context. We will not trace this step but we invite the reader to agree with us that the context of the first session is similar to the current session context thus all cases that can be extracted from that session should be examined.

The first session contains four queries, thus three cases can be extracted (since the behavioral part of a case should contain one query refinement step). As a result we have three cases that should be examined by the recommender. These cases and the solution they propose are summarized in the next table, a line per case. To better understand this table we explain how to read the first line: This line should be read as follows: the case C_1 has a behavioral part that is composed of the sequence of query configurations, results and evaluations of queries $Q_{1.1}$ and $Q_{1.2}$. The solution proposed by this case is the configuration of the query $Q_{1.3}$.

In table 5 we show the similarity measures between the different query configuration. The similarity between two queries is taken to be the arithmetic average of the similarities of each of the field that describe the query (with its results and evaluation). Note that the similarity between $Q_{2.2}$ and $Q_{1.1}$ is not computed since this similarity is not used in any computation of case similarities.

This leads to have the following similarities between the target case and three cases extracted from the first session:

If only the two most similar cases should be returned then we will have to choose between the solutions proposed by C_1 (the configuration of $Q_{1.3}$ and the solution proposed by C_3 (the configuration of $Q_{1.4}$).

Now the proposed solutions should be sorted according to their distance form the current query configuration of the current session ($Q_{2.2}$). We can verify easily the distance between the configurations of $Q_{2.2}$ and $Q_{1.3}$ is equal to 4 and the distance from $Q_{1.4}$ is equal to 5. Hence the solution proposed by the case C_3 will be proposed to the user. This results in proposing to the user the following query terms : `Synchronous, Groupware, Distributed, Concurrency, Control`, while the last query s/he has submitted contains the terms: `CSCW, Distributed, Coherency, Control`.

3 Related work

A great number of works have addressed the problem of query refinement in the information retrieval field. Most of works focus on *adding* new query terms to either improve the query recall [6] and/or the query precision [22]. One first difference between our approach and classical QR approaches that it does propose complete query configuration rather than just term adding (e.g. in addition to suggesting new terms our recommender may suggest to delete some existing terms). In addition, the Broadway-QR approach handles at the same time both problems of term selection and information source selection ; two problems that are usually handled separately in the literature [8]. In order to better compare the Broadway-QR with existing approaches we propose next a classification of QR approaches that is based on the following criteria:

Adaptability: We define *adaptability* as the system ability to *learn* form users interactions. Notice that this definition is slightly different from other adaptability definition, such as those given in [12, 9] where authors define an adaptable system as the one that allows users to explicitly modify its behavior.

Adaptivity: We define *adaptivity* as the recommender ability to *adapt* its recommendations according to the goals, knowledge and other characteristics of an individual user [4]. Adaptivity implies hence the use of some user modeling, or user profile construction, techniques [15]. Different methods for acquiring assumptions about the user are discussed in the literature [5]. In other words a system is said to be adaptive if two different users formulating at the same time, the same query got different recommendations.

Query	Query terms	Relevant document	Irrelevant docuemnt	Eval	Doc No.
$Q_{1.1}$	CSCW Concurrency Control	http://www.dgp.torento.edu http://www.acm.org	http://www.telekopperation.de http://www.diku.dk http://www-mmt.inf.tu-dresden.de	-1	353
$Q_{1.2}$	Groupware Concurrency Control	http://www.infomatik.munchen.de http://www.darstadt.gmd.de http://kiliwa.cicese.mx	http://salut.qucis.queensu.ca http://www.telekoperation.de	0	453
$Q_{1.3}$	Synchronous Groupware Concurrency Control	http://www.cse.ucsc.edu http://www.informatik.munchen.de http://www.unf.edu http://www.dgp.torento.edu	http://aether.cms.dmu.ac.uk	1	158
$Q_{1.4}$	Synchronous Groupware Distributed Concurrency Control	http://www.cse.ucsc.edu http://www.informatik.munchen..de http://www.cpsc.ucalgary.ca http://www.cs.columbia.edu http://www.caip.rutgers.edu	http://www.telekoperation.de http://www.browsbooks.com	1	148

FIG. 2 – *The execution of the first searching session*

Query	Query terms	Relevant document	Irrelevant docuemnt	Eval	Doc No.
$Q_{2.1}$	CSCW Coherency Control	http://www.cse.ucsc.edu http://www.tascntes.com http://www.computer.org http://stc.nepean.uws.edu.au	http://www.diku.dk http://www.ics.uci.edu http://www.acm.org/chi95 http://leibinz.imag.fr	0	19
$Q_{2.2}$	CSCW Distributed Coherency Control	http://www.cse.ucsc.edu http://www.tascntes.com http://www.computer.org http://stc.nepean.uws.edu.au	http://www.ics.uci.edu http://computer.org http://leibniz.imag,fr http://www-sor.inria.fr	0	16

FIG. 3 – *The execution of the first searching session*

Case	Behavioral part	Solution
C_1	$[Q_{1.1},Q_{1.2}]$	$Q_{1.3}$
C_2	$[Q_{1.2},Q_{1.3}]$	$Q_{1.4}$
C_3	$[Q_{1.3},Q_{1.4}]$	$Q_{1.4}$

FIG. 4 – *Cases that can be extracted form the first searching session.*

	$Q_{1.1}$	$Q_{1.2}$	$Q_{1.3}$	$Q_{1.4}$
$Q_{2.1}$	0.29	0.3	0.19	0.19
$Q_{2.2}$	-	0.29	0.19	0.22

FIG. 5 – *Similarities between queries submitted in session2 and those submitted in session1.*

	C_1	C_2	C_3
C_{Target}	0.29	0.19	0.20

FIG. 6 – *Similarities between the target case and the available cases in the system memory.*

Recommendation computation control scheme: Two main control schemes are identified for triggering the refinement process: user centred scheme and system-centred one. In IR literature the term *relevance feedback* method is oftenly employed to designate the first approach while the second scheme is frequently designated by the term *automatic query refinement*.

Recommendation selection scheme: The application of a query refinement recommendation can be either explicitly made by the user, or it might be automatically done by the system. In the first case we talk about user-controlled QR while in the second case we talk about system-controled QR.

Help nature: Different kinds of help can be provided by different QR recommenders. The most adopted help consists of adding new terms to the query [3, 10, 16]. Some approaches, such as Broadway-QR, suggest also modifying the current query terms. A third kind of help aims at helping users selecting adequate information sources [8].

In table 7 we illustrate the characteristics of some of the best known QR approaches comparing them to the Broadway-QR approach.

This table shows that few QR recommenders combine features presented by the Broadway-QR. Among the cited systems only one recommends information source (e.g. SavvySearch), and the systems other than Broadway which are adaptive and adaptable at the same time are rare. However, while most of existing QR approaches could produce recommendations starting form the first query, the Broadway-QR approach is unable to suggest any query refinement action unless the user or the system has already executed some previous query refinement process. For this reason, the Broadway-QR can be used as an additional step in a query refinement process that uses some of the existing QR approaches in a first step.

4 Conclusion

This work reports on the design and implementation study of a query refinement recommender to be used with query-based meta-search engine. The proposed approach is based on learning from past experiences of query refinement process in order to reuse successful refinements and to avoid already encountered failures. One first advantage of this approach is that it addresses both problems of query terms recommending and search engine selection at the same time. The recommender we propose relies on CBR techniques. This study demonstrates the feasibility of our approach. An experimentation in real world settings is planned for near future in order to better *configure* the recommender. Recommender configuration consists of setting the appropriate values for the different thresholds and parameters used in the specification of the recommender (e.g. the number of last queries to consider in the case restriction, the upper bound on case number to return at each case retrieval step, the similarity function parameters, etc).

Références

[1] J. M. Andreoli, U. Borghoff, Chevalier P.Y, B. Chidlovskii, R. Pareschi et J. Willamowski. The constrainte-based knowledge broker system. In *Proceedings of the 13th International conference on data engineering*, Birmingham, April 1997. IEEE Computer Society press.

[2] T.L. Brauen. *Document Vector Modification in the Smart Retrieval System: Experiments in Automatic Document Processsing.* 1971.

[3] C. Buckley, A. Singhal, M. Mitra et G. Salton. Automatic query expansion using SMART. In *TREC 3, Proceedings of the Third Text Retrieval conference*, pages 500–525, 1995.

[4] P. Bursilovsky. Efficient techniques for adaptive hypermedia. In C. Nicholas et J. Mayfield, éditeurs, *Intelligent Hypertext, Advanced Techniques for the World Wide Web*, volume 1326 of *LNCS*, chapitre 2, pages 12–30. Springer-Verlag, 1997.

[5] D.N. Chin. Acquering user models. *Artificial Intellligence Review*, pages 185–197, 1993.

[6] J. Cooper et R. Byrd. Lexical navigation: Visually prompted query expansion and refinement. In *Proceedings of international conference on digital libraries (DL'97)*, Philadelphia PA, USA, 1997.

[7] F. Corvaisier, A. Mille et J. M. Pinon. Information retrieval on the Web using CBR system: focusing on the similarity problem. In *Proceedings of the RIAO'97 Symposioum*, Montreal, Canada, June 1997.

[8] D. Dreilinger et A.E. Howe. Experiences with selecting search engines using metasearch. *ACM Transactions on Information Systems*, 15(3):195–222, July 1997.

[9] J. Fink, Ksba A et J. Scherk. Personnalized hypermedia information provision through adaptive

System	Adaptive	Adaptable	Computation	Selection	Help
Broadway-QR	Yes	Yes	User	User	IS Selec. Terms add.
LiveTopics (NetScape)	No	No	System	User	Terms add.
Smeaton's system [20]	No	No	User	User	Terms add.
DM-RMAP [22]	No	No	System	User	Terms add.
Top document feedback [3]	No	No	System	User	Terms add.
Past document feedback [10]	No	Yes	System	System	Terms add
Steepest descent QR [16]	No	Yes	System	User	Terms modify
Brauen's system [2]	No	Yes	System	System	Doc. Rep. Modify
Profile-based QR [13]	Yes	-	-	System	Term add.
SavvySearch [8]	No	Yes	System	System	IS. select

FIG. 7 – *Comparison of principal QR approaches.*

and adaptable system features: user modeling, privacy and security issues. In A. Mullery, M. Besson, M. Campolargo, R. Gobbi et R. Reed, éditeurs, *Intelligence in Services and Networks: Technology for Cooperative Competition, 4th International conference on intelligence in services and Networks (IS&N'97)*, LNCS, Cernobbio, Itlay, May 1997.

[10] L. Fitzpatrick et M. Dent. Automatic feedback using past queries: social searching? In ACM, éditeur, *Proceedings of SIGIR'97*, pages 306–313, Phialdelphia, PA, 1997.

[11] M. Jaczynski. *Modèle et plate-forme à objets pour l'indxation des cas par situation comportementales: application à l'assistance à la navigation sur le Web*. PhD thesis, Université de Nice Sophia-Antipolis, December 1998. (In french).

[12] C. Kaplan, J. Fenwick et J. Chen. Adaptive hypertext navigation based on user goals and context. *User modeling and user adapted interaction*, 3:193–220, 1993.

[13] R. R. Korfhage. *Information Storage and Retrieval*. Wiely Computer Publishing, 1997.

[14] H.S. Myaeng et R.R. Korfhage. Integrartion of user profiles: Models and experimenets in information retrieval. *Information Processing and management*, 26(6):719–738, 1990.

[15] S.C. Newell. User model and filtering agents for improved internet information retrieval. *User modeling and user adapted Interaction*, 7:223–237, 1997.

[16] V.V. Raghavan et H. Sever. On the reuse of past optimal queries. In *Proceedings of the 18th International ACM-SIGIR Conference on Research and Development in Information Retrieval*, Seatel, Washington, July 1995.

[17] P. Resnick et H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[18] C.K. Riesbeck et R.C. Scamck. *Inside Case-based reasoning*. Elbraum, 1989.

[19] E. Selberg et O. Etzioni. Multi-service serach and comparison using the MetaCrawler. In *Proceedings of the 4th International Wolrd Wide Conference*, 1995.

[20] A. F. Smeaton et F. Crimmins. Relevance feedback and query expansion for searching the web: a model for searching a digital library. In C. Peters et C. Thanos, éditeurs, *Research and advanced technology for digital libraries*, number 1324 in LNCS, pages 99–112. Springer, 1997.

[21] B. Trousse, M. Jaczynski et R. Kanawati. Using user behavior similarity for recommandation computation: The broadway approach. In *Proceedings of 8th International conference on human computer interactions (HCI International'99)*, Munich, August 1999.

[22] B. Vélez, R. Weiss, M.A. Sheldon et D.K. Gifford. Fast and effective query refinement. In *Proceedings of SIGIR'97*, Philadelphia, USA, 1997.

[23] J. Xu et W.B. Croft. Query expansion using local and global document analysis. In *Proceedinngs of the 19th Annual International ACM SIGIR Conference on Research and Developement in Information Retrieval*, pages 4–11, Zurich, August 1996.

[24] T.W. Yan, M. Jacobsen, H. Garcia-Molina et U. Dayal. From user access patterns to dynamic hypertext linking. *Computer Network and ISDN systems*, 28:1007–1014, May 1996. (proceedings of the 5th international WWW conference).