Journal of Uncertain Systems
Vol.1, No.2, pp.109-123, 2007
Online at: www.jus.org.uk

A Smart Query Formulation for an Efficient Web Search

Giuseppe Fenza, Vincenzo Loia*, Sabrina Senatore
Dipartimento di Matematica e Informatica, Universitá degli Studi di Salerno
via Ponte Don Melillo 84084 Fisciano (SA) - Italy
{*gfenza, loia, ssenatore*}*@unisa.it*

Received 3 January 2007; Accepted 30 January 2007

Abstract

Traditional search engines rely on keyword-based matching, recovering the documents which present some occurrences of the input keywords, but ignore at all the data meaning of the retrieved documents. Thus, long lists of pages links are returned but actually only a handful of pages contain reference to relevant web resources and meet the needs of users. The exigency of major awareness in the interpretation of web data yields new approaches and methodologies for improving the web search and retrieval, by taking into account the context of information, related to the user query. This work presents an approach for supporting the user in the Web search activity: it achieves the interpretation of the input query and, on the basis of the the local knowledge, replies by providing (links of) web pages which are more relevant to the content meaning of the input query. The approach combines intrinsic potential of the agent-based paradigm with the modeling of knowledge through techniques of soft computing. The agents encode the semantics of data, by exploiting ontologies, in order to grasp the actual query meaning. The information elicited by the query interpretation represents an add-on, aimed at augmenting the system knowledge, exploited in the discovery of web pages which match the user request.

Keywords: Web search, clustering, ontologies, multi-agent system

1 Introduction

In recent years, the Web has been rapidly increased, by providing a large amount of information, often unstructured (or semi-structured) spread over different hosts in a distributed environment. Also, the growth of the Internet usage and contents makes difficult the information access, making the task of Web search highly critical. Current Web search engines are built to provide an answer to all the requests, independent of the special needs of any individual user. The same query, submitted to a typical search engine returns identical results, regardless of the expectations and needs of different submitting users. Thus, only a handful of the Web sources are related to user query, because no relationship between the requested information and the found one exists.

The search engine quality is mainly measured as retrieval capabilities; some search techniques are based on the classification of the discovered documents and then referenced in a search database by human experts or by artificial agents (softbots) (i.e. Yahoo!, Google) [24]. Interesting research initiatives of famous search engines, try to improve the query results, through the interpretation of the user request and the removal of noisy data as well as to cluster information according to the most likely meaning, through some rife ranking methods. Some approaches converge to the definition and building of domain specific Web search engines which collect and index the relevant Web pages, by exploiting crawlers which gather only domain-specific pages [4],(even though the crawler-based search

*Corresponding author: V. Loia (loia@unisa.it).

engines are not efficient in terms of time and bandwidth). On the other hand, reusing the existing large indexes of general purpose search engines is a solution to retrieve [26], after a filtering activity, documents from a specific domain (though the the response time to the user query are slow too). Similarly, other approaches achieve clustering of results for automatic organization (into categories) of documents (i.e. WiseNut [35] and Vivissimo [34]). Metasearch environments, instead implement strategies that apply user queries to several search engines simultaneously. However many of these approaches do not consider the semantic relationships existing among terms: the query ambiguity and the vocabulary gap represent extant impediments that confirm the search engine technology is far from the ideal response to a certain query.

Analysis on the search engine performances emphasizes that up to half of search sessions fails to deliver good results [29]; in addition, the log files analysis shows a poor correspondence between the query and the results content [6]. The last decade saw a number of research endeavors to deal with these web search problems, producing numerous approach and methodologies for customizing web search activities. Personalization provides an operative method for achieving efficient information retrieval [28]. Many approaches have been proposed for personalized web search systems; firstly the PageRank algorithm [15] ranks the search results on the basis of global importance scores, by using the linkage structure of the Web.

Some approaches exploit the agent technology to help users in the web searching activity [21, 8]; studies on the information gathering emphasize the role of agents to drive the personalization of information, in order to satisfy the user "oneshot" request [36] and find a compromise among cost/quality tradeoffs, information overloading, mining linkage structure and users browsing behaviors in order to improve the web search.

On the other hand, the demand for efficient and effective approaches to retrieve the required information finds in the focused crawlers a flexible tool for personalized web search [16]. In fact, during the traversing of the Web, the links to follow are selected on a basis of a measure of how likely a document is linked to a target page.

In addition, remarkable issues are reached in web usage mining through Soft Computing frameworks [2].

The last trend sees the diffusion of tagging services for allowing users to add terms of their own interest and relevance to Web pages [7, 27]. For instance, in [14] a particular schema for semantic annotation with respect to real world entities is defined.

Nevertheless, the structure of information is still based on HTML-format for data representation and thus makes the conversion of the syntactic Web content into the semantic one, an expensive activity in terms of human resources and time.

In fact, as known, the Semantic Web "support" depends on the availability of a critical mass of metadata for the web content, associated with the respective formal knowledge about the world. Thus, the practical semantic annotations need of flexible knowledge modeling techniques. The diffusion of the Semantic Web supplies new models for providing a major contextual support: some methods use the available ontologies to disambiguate the concepts used to query [1].

The role of ontologies is prominent to discover the semantics of terms and identify the reference context of the query; studies presented in [19, 13] emphasize effectiveness of algorithms for automatic semantic reconciliation and the similarity among ontology-based schema; in [9] a machine learning technique is adopted to measure probabilistically the similarity between concepts of two different reference ontologies through the relaxation labeling technique [25]. In [33] a mapping of ontologies in common domains is accomplished, through the analysis of similarity among concepts which are pre-existing or created collecting information during web searching activity. Similarly, the work in [11] describes a consensus support system, which on the basis of a multi-granular linguistic methodology, provides an expertise degree by combining knowledge defined by different linguistic term sets.

The solution presented in this work aims at assisting the user during the query formulation: the system provides a support for interpreting the semantic of the query terms and, at the same time, aims at producing a more accurate list of relevant results through fuzzy techniques. The key idea consists of finding alternative formulations of the input query that better details the input query (through synonyms, related terms, etc. of query words), besides to reflect the inchoate user intention. This "augmentation" of the initial query becomes the knowledge the system acquires during the user sessions, useful to tailor the user search context. The agents play a crucial role in the discovering of both semantic similarity or specialization/generalization relationships among terms, in order to understand the meaning of query. On the other hand, the agents contribute to the retrieval of relevant web resources, providing an spidering activity, nearby the more traditional web search.

Numerous approaches in the literature emphasize the role the agents play in this domain [22, 5], for supporting activities of discovery, parsing of web resources; previous our works [17, 18] take advantages of using these techniques in the research domain.

The paper is organized as follows. Section 2 introduces the system description by sketching the whole working flow, and the role played by the agents in each activity; then implementation details in Section 3 deepen the main activities by presenting the relative formal model. Section 4 describes the system performance through the experimental results and finally, the conclusions close the paper.

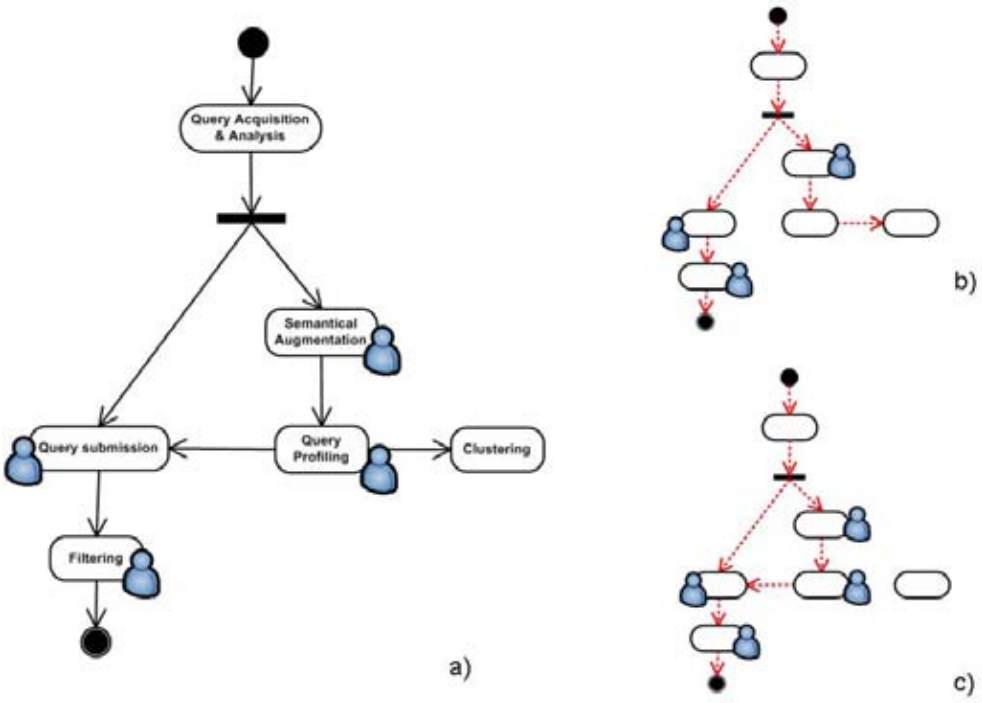

Figure 1: Activities diagram of the system a) and the main action flow b)-c)

2 Framework overview

The whole framework is designed to provide a straightforward interaction between the functional and agent-based components of the system. Figure 1 shows the data flow of the system architecture,

described by the activities diagram. In details, Figure 1-a shows the system architecture: the overall activity diagram is described by tracing the data flow; at the same time, the agents role is emphasized in the activities where they intervene. In this section, the activities and the main agent-based interactions are briefly described; further investigations are given in the next section.

- *Query Acquisition & Analysis*: the input of the query triggers this activity which starts up the system flow. The query is acquired through an aid-user interface, then it is analyzed syntactically by collecting only the relevant words and discarding all the non-informative words. Thanks to the user interface, based on an RDF-based tagging, the main concepts of the query can be discriminated and isolated. The result of this activity is a sequence of "atomic" stemmed words that represents the user query.

- *Semantical Augmentation*: once the query is reduced to the basic form, this activity aims at enriching the interpretation of the query, through the discovering of feasible ontological relationships, in order to get a possible augmentation of the input query. In this activity, the query terms are enriched with by consulting ontologies and dictionaries in order to elicit terms that are related (through specialization/generalization) to the input query. The intervention of agent is indeed, to support the identification of terms semantically related to the input words in order to get a sequence of terms composed of initial query terms and (eventual) discovered additional terms.

- *Query Profiling*: each query is translated into a corresponding vector-based representation; then these vectors are locally stored. Really, this activity manages two flows of information (see Figure 1-a), as described in the following.

  1. All the collected vectors constitute the knowledge of the system, which will be exploited in the execution of the *Clustering* activity. In fact, this flow of activity guarantees the arrangement of the gathered vectors in a matrix form, suitable to be the input of the clustering algorithm.
  2. This flow branch occurs when the system is at regime. In this phase the agents intervene and exploit the clustering (once it has been executed) in order to compare the vector-based representation of the current user query with the clustered ones. The closest clustered vectors represent the queries more similar (as terms and topics) to the input one and will be exploited in the *Query Submission* to improve the expressiveness to the input query.

- *Clustering*: the queries are clustered. Each cluster is interpreted as a category of topics, according to the collected queries. Let us note the clustering is periodically re-executed, triggered by the *Query Profiling* activity.

- *Query Submission*: the "atomic" input query (coming from *Query Acquisition & Analysis*) as well as some queries (in the clustered collection) that are similar to the input topic (coming from *Query Profiling*) are individually submitted to a search engine. A spidering activity is then triggered.

- *Filtering*: The lists of pages links, returned by the search engine, for each one of submitted queries are then filtered and merged to get a unique ranked list of page links. The agents contribute to produce a final list represents a suitable balancing between the input query (often ambiguous and not enough explicative to be processed by a search engine) and the stored stereotype of queries (which can reflect the input query partially).

These activities are executed according to the traced flow, outlined in Figure 1, supported by the agents. Substantially, the system consists of two main action flows (see sub-parts *b* and *c* of Figure 1):

- *start-up* : at the beginning, the system contemplates a training phase to collect the input queries, stores them and then applies the clustering to getting grouping of discovered semantic similarity. Figure 1 (part *b*) shows all the activities involved (the dotted line shows the relative data flow). In the activity diagram, a fork point is evidenced, which produces a split of execution path. Thus, on one hand, the system replies to the input query of the user, by providing the list of page links as in the typical web search approach (*Query Submission* → *Filtering*), on the other hand, collects the information (query terms, semantic relationships, etc.) to achieve a local arrangement of the acquired knowledge, through the clustering procedure (*Semantical Augmentation* → *Query Profiling* → *Clustering*).

- *regime*: in this phase (see Figure 1 part *c*), the system exploits the acquired knowledge (i.e. the queries previously arranged in clusters) for the *Query Submission* activity. In particular, the agents contribute to produce the resulting list of links to web pages, to return to the user. More specifically, by exploiting the local knowledge, they evaluate which clustered queries are similar to the input one: the selected queries together to the input one are given to the *Query Submission* activity. The returned lists (one for each query) of links are parsed and filtered by the agents (*Filtering* activity); then, these lists are merged into a unique, final one, suitable to be presented to the user (*Semantical Augmentation* → *Query Profiling* → *Query Submission* → *Filtering*).

3 Implementation details

This section presents a more complete description of the introduced activities. The theoretical aspects and the implementation choices will be analyzed, in order to give a more clear characterization of the system and the inherent interactions.

3.1 Query Acquisition & Processing

This activity captures the meaningful terms which compound the input query. Initially, these terms are analyzed to reduce their number to the minimum, by applying typical approaches to text parsing and analysis: after the stop-word removal, (a small stop list of words *for, that, the, these, to, which, a, any, and, etc, an, another, in, of, or* is considered), a stemming algorithm [23] reduces the variant forms of a word to a common root.

Furthermore, the "interpretation" of the query meaning is supported by a user interface, as shown in Figure 2. The snapshot shows a panel, split in two main parts: on the top, a typical text field for a query formulation; the other text fields enable the insertion of additional information (or semantic annotations) which details the concepts expressed in the query.

Just to give an example, let us suppose a user looks for all the publication of a laboratory called "Lasa Lab" on the "Semantic Web" domain. In traditional searching tools, the relative sequence of words should be *Publications of the "Lasa Lab" on "Semantic Web"*. The search engine returns links to web pages which contain the same sequence of words specified in the query, even though the context and the meaning is completely different. Thus, to better address polysemic problems, a habit of most search engine is to provide specific names between apexes, in order to clearly identify the portion of query to evaluate as a whole. Our system accepts the same query sequence (see Figure 2) and then elaborates it to discriminate the main concepts. Thus, after the query input (*Query*

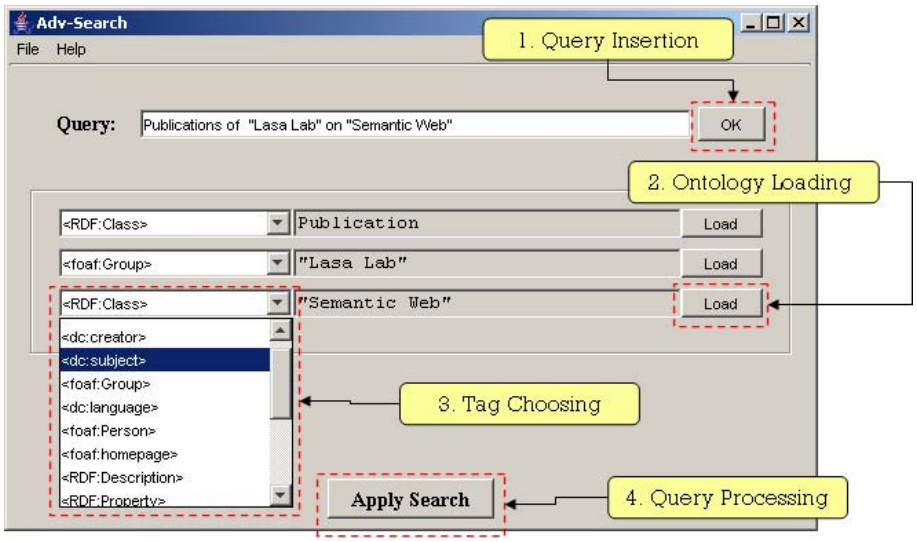

Figure 2: Screenshot of the Query Acquisition interface

Insertion, Figure 2), the system produces a basic sequence of main concepts in a default RDF-based format, listed as follows:
<rdf:Class> Publication </rdf:Class>
<rdf:Class> Lasa Lab </rdf:Class>
<rdf:Class> Semantic Web </rdf:Class>

The user can modify the system setting, by loading the ontologies (previously download and locally saved) which she/he believes more appropriate to describe the elicited concepts (*Ontology Loading*, Figure 2). For instance, looking at the panel in Figure 2, the user has substituted the tag <rdf:Class>, for the term "Lasa Lab" with the tag </foaf:group> of the foaf ontology [12] which better explains the meaning of the term (i.e. a name of a working group), whereas the user is changing the tag for "Semantic Web" with a tag <dc:subject> of the ontology Dublin Core [10]. The prefixes foaf and dc refer to the ontology where these meta-terms are defined. Let us note the user could choose to not specify (because he does not have experience in the semantic web domain); thus only the basic RDF tags *rdf:Class* can be associated to the stemmed words. In this case the system works too, although no further interpretative support is given to the system, in addition to its local acquired knowledge.

However the user can arbitrary select appropriate meta-terms, choosing the ad-hoc ontology. In fact, once loaded the suitable ontology, the tagged ontology terms appear in the sliding window of the combo-box, corresponding to the focused query terms (*Tag Choosing*, Figure 2). Thus, the input query is ready to be further processed (*Query Processing*, Figure 2).

3.2 Semantical Augmentation

The user interface, shown in Figure 2 allows the specification of the input query, by adding further details, about the meaning of query. The RDF-based tagging of the terms in the interface aims at detecting and/or disambiguating the actual meaning of the words. In particular, the system maintains a prefixed set of ontologies (ad-hoc defined or already existing [30, 31]), which represents the local *natural knowledge*. In this phase, a specialized agent supervises the discovery of relationships between the "atomic" terms of the initial query and the concepts in the ontologies, loaded in the system. A sketched algorithm lists the main tasks the agent achieves as shown in the following.

Let us suppose Q an input query composed of the sequence of terms $t_1, t_2, \ldots, t_h$ and the set $\mathcal{O} = \{O_1, O_2 \ldots O_n\}$ of reference ontologies.

INPUT: list of terms of input query Q associated to the reference ontologies:
$O_i : t_i, \ldots, O_j : t_j$;
OUTPUT: augmented Q'
$O_i : t_1, \ldots, O_j : t_h; O_k : v_l, \ldots, O_y : v_z$;
where $v_1, \ldots, v_r$ are additional terms of reference ontologies, related semantically to the query terms with i, j, k and y $\in$ [1, n] and l, z $\in$ [1, r].

for each term t_i in the query Q :
if O_j is the default RDF-tag
$O_y \leftarrow$ Agent:locate(t_i, $\mathcal{O}$)
else if O_j is $\in \mathcal{O}$
$v_l \leftarrow$ Agent:findRelation(t_i, O_j)

The functions labeled by the word "Agent" represents the two tasks *locate(t_i, $\mathcal{O}$)* and *findRelation(t_i, O_j)* achieved by the agents. The idea presented in the algorithm emphasizes how the agents affect the execution: if a term is not annotated by the user (the default RDF-tagging is automatically given by the system), the agent tries to find a match of the keywords with a concepts described in one of the ontologies of the reference set $\mathcal{O}$ (task *locate* of the agent). In the example presented in the previous section, for instance, the agent looks for the term "publication" and probably discovers it in the FOAF ontology where the homonym concept appears. This match adds further meaning to the terms which before was "unknown" to the system. Each term without annotation, is looked up in the ontologies in order to individuate the reference domain: if a term appears in more than one ontology, a further analysis on the remaining query terms is applied. The reference domain is determinated by the ontology which describes more query terms. If a term is not defined in any ontology, it will be not considered in the query evaluation and then, it will be put in a list of unknown terms.

Otherwise, when the term is provided by annotation (by selecting the concept of a loaded ontology), the agent works to find some existing relationships with other concepts in the specified ontology, in order to improve the meaning of the initial query (task *findRelation* of the agent). For instance, in previous example, the term "Lasa lab", annotated with the FOAF tag *group*, can be further described by the related term *member* of the same ontology.

This way, the system produces additional local knowledge that represents the *acquired knowledge*, obtained by the semantic augmentation of the query.

3.3 Query Profiling

The system translates the input queries into vector-based representations, before of storing them locally. These vectors compose a data matrix and will be the input of the *Clustering* activity, in the *start-up* phase. Specifically, each query is a sequence of terms, composed of its own terms as well as those elicited by the enrichment in the *Semantical Augmentation* activity. In order to build the data matrix, a set of common terms (among the terms of all the queries) should be defined; this set is called *reference set*. The criterion is to select the most recurring terms among all the obtained queries terms. Thus, once defined the *reference set*, the vectors, associated to the input queries are built: each element is a weight value, in corrispondence to a query term. More formally:

Definition 1: *The query can be defined as* $Q_i = \{(t_1, w_{i,1}), (t_2, w_{i,2}), \ldots, (t_n, w_{i,h})\}$, *where* t_i *is a query term (original or added term) and* $w_{i,j}$ *is the weight associated to the term* t_i *in* Q_i. $W_i = \{w_{i;1}, w_{i;2}, \ldots, w_{i;h}\}$ *represents the feature vector of the query* Q_i.

These weights are computed by measuring the relationships existing between the terms of the current query and the terms belonging to the reference set. The following algorithm of selection computes the weights of both the terms typed by the user (come from the initial query) and the "derivative" terms, obtained by ontological relationships.

Definition 2: More specifically, let us suppose t is a term of the input query and t' is a term in the *reference set* T;

- the term t' comes from the original query $\Rightarrow weight(t') = w_{i,t'} = 1$
- the term t' is derivative and f(t, t') $\neq 0$ $\Rightarrow weight(t') = w_{i,t'} = f(t, t')$

where the function *f(t, t')* $\in [0, 1]$ describes a taxonomic comparison between the terms t and t' and can assume the following values:

$$f(t,t') = \begin{cases} 1 & t = t' \\ \frac{1}{2} + \frac{1}{2*s+1} & \text{if } \exists \text{ a relationship between t and t'} \\ 0 & t \neq t' \end{cases}$$

where s represents the distance in terms of number of edges included between the terms t and t' in the relationships of the reference ontology O.

Hence, given a collection of queries $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_n\}$, each query Q_i is represented by an h-dimensional vector of weight $W_i = \{w_{i,1}, \ldots, w_{i,j}, \ldots, w_{i,h}\}$, where each element $w_{i,j}$ is the weight associated to a term $t_j \in T$ with $j = 1, \ldots, h$. These vectors form a data matrix $n \times h$ which is the input of the clustering algorithm, as described in the following. Let us observe the data matrix is built in the *start-up* phase, when an initial arrangement of submitted queries (clustering) is required. At the *regime* phase, each incoming input query is translated in vector too, for guarantying homogeneous comparisons with the previous clustered queries. This task is entrusted to a task-oriented agent which carries out this comparison: it inspects the clustered queries, choose those queries (one or more) which are more similar to the current one (as described in the following) and then exploits them in the web search activity (*Query Submission* and *Filtering*).

3.4 Clustering

The clustering algorithm achieves a partitioning of given data into clusters, so to group queries which present similarity. In general a partition holds two properties: homogeneity within the clusters (data in a cluster must be similar) and homogeneity between clusters (data of different clusters have to be as different as possible).

Our clustering approach is based on the well-known Fuzzy C-Means algorithm (shortly, FCM) [3]. The FCM algorithm takes as input a collection of patterns of a universe U (in our case, the collection of queries $\mathcal{Q}$) in form of matrix and produces fuzzy partitions. The output is a partition of the given patterns (queries) into (prefixed) c clusters. The FCM algorithm recognizes spherical clouds of points (clusters) in a multi-dimensional data space and each cluster is represented by its center point (prototype). This process is completely unsupervised, aimed at identifying some inherent structures in a set of data.

More formally, the FCM algorithm aims at minimizing the objective function constituted by the weighted sum of the distances $dist_{i,j}$ between data points $\underline{x_i} = (x_{i,1}, x_{i,2}, \ldots, x_{i,h})$ and the prototypes $\underline{v_j} = (v_{j,1}, v_{j,2}, \ldots, v_{j,h})$, according to this formula:

$$J(U,c) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{i,j}^{m} (dist_{i,j})^2 \quad (1)$$

where c ≥ 2 is the number of clusters, $u_{i,j} \in$ [0,1] is the membership degree of $\underline{x}_j$ (j=1, ...,n) in the i-th cluster A_i (i=1, ..., c), and $m > 1$ is the fuzzifier, which controls the quantity of fuzziness in the classification process (common choice of fuzzifier is $m = 2$) and finally $dist_{i,j}$

$$dist_{i,j} = dist(\underline{x}_i,\ \underline{v}_j) = \sqrt{(||\underline{x}_i, -\underline{v}_j||^2)} \tag{2}$$

just represents the euclidean distance between the data $\underline{x}_i$ and the center $\underline{v}_j$ of the j-th cluster.

In details, $U = (u_{i,j})$ is a $c \times n$ matrix of cluster memberships satisfying some constraints. In particular, let us denote M_{fc} a family of fuzzy partition matrices:

$$M_{fc} = \left\{ U \mid u_{i,j} \in [0,1]; \sum_{i=1}^{c} u_{i,j} = 1; 0 < \sum_{j=1}^{n} u_{i,j} < n, \forall\, i, j \right\}, \tag{3}$$

and $V = (\underline{v}_1, \ldots, \underline{v}_c)$ is the ordered collections of cluster centers.

The data matrix is composed of n queries, each one with h values $w_{i,k}$ ($i = 1, \ldots, n$ and $k = 1, \ldots h$), associated to the corresponding term. In our approach the generic vector $\underline{x}$ is a query $\underline{p} \in \mathcal{P}$ as defined above, whereas the center $\underline{v}$ of the cluster represents the sample topics of that cluster. The FCM algorithm produces clusters which represent the main topics or subjects, elicited by the collected queries. As said, the output of clustering is a fuzzy partitions, where each cluster describes topics or subjects (relative to the queries terms). Consequently, a straightforward mapping can be defined, which associates these clusters to classes (or categories) of topics. Furthermore, each query of the input set assumes a membership value for each cluster. When the memberships of a query is equally distributed among more clusters, it means that query does not reflect the topics of no specific class; thus it is discarded and stored it in a "dump" category.

In our system, the clustering is an activity which is periodically re-evaluate and eventually re-executed, when, for instance, the number of queries not well represented by no defined categories increases. Let us observe the distance $dist_{i,k}$, defined in (2) is exploited by the agents to measure the distance between the current query and those clustered ones. In fact, the queries closer to the input one (in terms of minimal distance) are more similar to it and are candidate to be exploited in the submission.

3.5 Query Submission and Filtering

The data flow coming from *Query Profiling* and *Query Acquisition & Analysis* converges in the Query Submission activity: substantially the original query as well as the closest clustered queries (one or more) are separately "submitted" to a search engine. The results of the traditional search are further processed by the agents: they achieve a spidering activity, for each submitted query: inspect each returned Web page, and cross inner links, (given a fixed depth level of navigation) and select the pages that match completely/partially the input topic. Before of presenting to the user, the *Filtering* activity combines the relevant links associated to the page crossed by the spiders, according to this algorithm:

INPUT: input query Q
list of page links $L_1, L_2, \ldots L_n$
OUTPUT: ranked list of $L_1, L_2, \ldots L_k$, with $k \leq n$

for each link in the $\bigcup(L_1, L_2, \ldots L_n)$
  for each term t_i in the query Q
    $occ_{i,j}(t_i) \leftarrow$ Agent:explores($L_j, depth$)
      (computes the occurrences of t_i, $occ_{i,j}(t_i)$ in L_j
      and in all the joint pages, crossing up at fixed *depth* level of navigation)
    compute $w_{i,j}(t_i)$ according to Definition 2 in Section 3.3

$rel_{i,j}(t_i) = occ_{i,j}(t_i) * w_{i,j}(t_i)$
$W(L_j) = \sum_i rel_{i,j}(t_i)$
sort decreasingly $W(L_j)$
return the first L_j, $j = 1, \ldots$,k, according to the computed rank.

Thus, the agents compute the occurrences of relevant terms, for each the visited pages, during their spidering activity. The algorithm computes the overall relevance value $rel_{i,j}(t_i)$ associated to each term, then the final value is assigned to the link L_j of the joint web page. The first k pages links (ranked according to their weight W) are presented to the user, arranged in a web page arranged in a web page typically returned by a search engine.

In this version the approach does not apply a further analysis on the possible user feedback.

4 Performance evaluation

The system performance is evaluated, by considering the *start-up* and *regime* phases of activities. In the *start-up* phase, the performance evaluation is based on the clustering execution of the collected queries. Final goal, indeed is to evaluate the effectiveness of the system, once the training phase is completed rightly. The clustering has been executed on different samples of input queries, by trying to extract the main categories of topics. Afterward, in the *regime* phase, the computed categories have been used for detecting the proximity of the current input query with the clustered queries. The testing set simulates the search activity of web communities of users. In fact, our opinion is the described system can be efficiently employed for community-based personalization of Web search results. In this environment, the argumentations often are based on a limited number of topics, bound to the common context of the groupings of people which share expertise and interests on specific subjects. This restriction meets the limited though manifold set of terms used in the initial training (which is naturally limited to represent only a set of concepts); on the other hand, the context of web communities is suitable to assess the performance of our system.

Let us emphasize the system is still in progress. Therefore the testing activity is rather constrained. However, the *start-up* phase, which carries out a classification of input queries, is evaluated through the two well-known measures: the *precision* and the *recall*. In our approach, we evaluate the recall as the ratio of the number of relevant, well-classified queries to the total collection of collected queries. A "well-classified" query is a query that, after the clustering, will appear in the expected class. Thus we have a set of predefined categories of topics. Let us note the produced clustering is strongly related to the human interpretation of the given sample of queries: the clusters represent arguments and topics subjectively evaluated, bearing in mind what the cluster, and hence the corresponding class, represents for the humans. Thus, according to the set of input queries, the clustering algorithm produces some clusters that are in correspondence with categories/classes interpreted by humans.

More formally, given a set of the prior expected categories $\{A_1, A_2, \ldots, A_n\}$ of topics (based on a wide though limited set of query subjects); let us suppose that after the clustering execution, the associated discovered clusters are$\{C_1, C_2, \ldots, C_n\}$. Let us express the individual recall r_i as follows:

$$r_i = \frac{|A_i \cap C_i|}{|A_i|} = \frac{well\ classified\ queries\ in\ C_i}{all\ the\ queries\ in\ A_i}. \quad (4)$$

Thus, the overall recall can be expressed the weighted average of individual recalls (substituting the (4)) [20]:

$$\begin{aligned} R &= \frac{\sum_{i=1}^{n} r_i * |A_i|}{\sum_{i=1}^{n} |A_i|} = \frac{\sum_{i=1}^{n} |A_i \cap C_i|}{\sum_{i=1}^{n} |A_i|} \\ &= \frac{\sum_{i=1}^{n} well\ classified\ queries\ in\ C_i}{all\ the\ queries} = \frac{well\ classified\ queries}{all\ the\ queries}. \end{aligned} \tag{5}$$

The same way, the precision p_i represents the percentage of all the well-classified queries, with respect to the expected clusters:

$$p_i = \frac{|A_i \cap C_i|}{|C_i|} = \frac{well\ classified\ queries\ inf\ C_i}{all\ the\ queries\ in\ C_i}. \tag{6}$$

The overall precision P is the weighted average of individual precision

$$P = \frac{\sum_{i=1}^{n} p_i * |A_i|}{\sum_{i=1}^{n} |A_i|}. \tag{7}$$

Let us note the recall and precision are two measures exploited for the evaluation of crisp clusterings.

Herein we assume that a query belongs to that cluster whose membership is the highest. The queries, whose membership is equally distributed among all the clusters, are not considered. These measures evaluate the clustering performance (in *start-up* phase); in the *regime* phase instead, the performance of the system is based on the query response, given a set of submitted queries. Herein, we exploit the more classical definition of recall and precision in the Information Retrieval domain.

Indeed, according to [32], we consider the micro-average of the individual precision-recall curves, exploiting a number λ of steps up to reach the highest recall value. Thus, let Q be the set of queries and D the set of all the relevant pages (or better, all the links to the web pages that match the queries). The micro-averaging of recall and precision at step λ (with $\lambda \in N$) is defined as follows:

$$Rec_\lambda = \sum_{R \in Q} \frac{|D_R \cap B_{\lambda,R}|}{|D|}, \quad Prec_\lambda = \sum_{R \in Q} \frac{|D_R \cap B_{\lambda,R}|}{|B_\lambda|} \tag{8}$$

where D_R is the answer set of pages for given query request R $\in$ Q, B_λ is the set of retrieved pages at the step λ and $B_{\lambda R}$ is the set of all relevant pages, retrieved at the step λ.

4.1 The experimental results

In the light of the study described in the previous section, firstly the quality of clustering in the *start-up* phase is estimated. Table 1 shows the performance of some clustering executions, expressed in term of recall/precision. For each row, some parameters have been set: the number of submitted queries, the retrieved pages, the reference terms (belonging to the *reference set*), the number of clusters; then the evaluation of recall and precision, expressed in percentage. In general, the clustering performance presents interesting results: the computed values in terms of recall and precision are rather accurate for the most of experiments.

Table 2 shows the main categories elicited by this experimentation: according to the topics of terms in the reference set, a pertinent category is associated. Furthermore, for each category, the more relevant sample of documents associated to the first three page links of returned list is shown. Each test execution can exploit a different selection of terms in the reference set.

Exp. No.	#query	# pages	# reference terms	#clusters	Recall	Precision
1	40	550	13	3	94.8%	96%
2	50	100	18	3	95.1%	97.2%
3	70	100	20	5	93.7%	94.7%
4	100	200	30	5	96.9%	97.1%
5	120	300	32	6	97.7%	92.2%
6	150	370	38	6	97.9%	92.1%

Table 1: Evaluation measures of clustering performance

CatNo.	common feature words*	sample documents	Category Name
1	PDA, palm, bid, battery, UBS, connectivity, operative, system	1. Web Portal of the auction communities 2. uBid.com 3. onSale.com	Auction/ shopping
2	craft, soup, candle, handmade, gift, soy fragrance, tart, scent	1. The crafts fair online 2. Artsefest.com 3. The Woodland Candles	Craft
3	Galaxy, Solar, Stars, Chart, Sun, System, Planet	1. Introduction to Astronomy 2. NASA Watch: Space news 3. Astronomy Magazine	Astronomy
4	Web, interchange, Community, RDF, Ontology, URI, technology	1. W3C Semantic Web Activity 2. Semantic Web roadmap 3. Portal of the Semantic Web Community	Semantic Web
5	music, information, lyric, artist, list, player, musician instrument, chorus	1. Welcome to MusicBrainz! 2.commUNITY music 3. Internetional Journal of Community Music	Music
6	golf, player, club, winner competition, schedule, tournament, tip, draw	1. Welcome to the tee1up golf society web site 2. GOLF- Welcome to Scottish Putting 3. Foxtail Men's Golf Club	Sport:Golf

* composed of terms of original query and the additional ontological terms

Table 2: The computed clusters and the associated categories

The *regime* phase evaluates a further hundred of queries submitted to the system: we measure the improvement of the system, by considering the "acquired knowledge", gathered by the *start-up* phase. Thus, the original query submitted by the user is enriched by some terms belonging to the closest queries, among all the clustered ones. Figure 3 shows the graph of micro-averaging of recall and precision; we have fixed λ= 13 and we have evaluated these measures at each steps λ. Let us note at each step λ, we compute recall and precision by considering a subset of retrieved pages which is incremented at each step λ, until to get (at the step λ=13) the complete set of retrieved web pages (more precisely, the set of the links associated to the retrieved web pages). In this case indeed, the recall assumes value 1 (i.e. all the relevant pages are retrieved).

According to the characterization of these two measures, there is a reverse connection between these two measures: an higher precision usually implies a lower recall and vice versa. In fact, on the average, the recall assumes low values (less than 0.6) when the precision assumes an high and rather constant value (in the range [0.7, 0.8]). Successively, the recall increases, even though the precision never assumes values lower than 0.4.

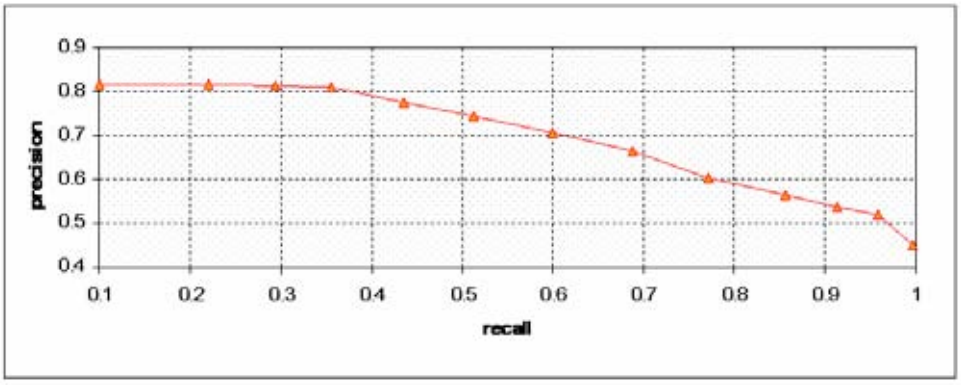

Figure 3: Micro-averaging of recall/precision (with $\lambda = 13$)

5 Conclusions

The gap existing between the desired information and the gathered one reveals limits of semantical rather than syntactic interpretation. Research studies and challenging endeavors are yet in progress to improve the query results of web search engines, through the interpretation of the user request, although the performance in terms of page relevance and retrieval capabilities is yet far from to being acceptable.

The work presents an approach for improving the effectiveness of the web search. The system exploits techniques of crawling often used by the common web search engines. The add-on of this system is to interpret the query submitted by the user and to enrich it with related terms. In addition the system suggests some other queries (among those ones stored locally) that could be similar to the given one. Indeed, the basic idea is "if the user query is X, probably further similar queries like X *and* Y or Z, where Y and Z are semantically related to the information X could be relevant for him". Our approach constitutes a support for the web search activities of the user: it improves the accuracy of reply during the web search activity, by exploiting the "knowledge" acquired by the system in the start-up phase. Anyway, it is important to build a good sample of queries to guarantee the system provides satisfactory results in terms of the performances and the response to the input query. In this first analysis the system has been evaluated in the context of web communities, even though the system maintain, through the *Query Profiling* activity, some grouping of users, better called "virtual" communities of users which present similar inclinations, interests and topics, according to the interpretation of submitted queries. As future extensions, we would like to evaluate the feedback of the users (for instance, the page links clicked in the list returned by the search engine), in order to benefit of the the exploitation of such information for improving the query characterization.

References

[1] Somjit Arch-int. Web document clustering using semantic link analysis. In *CIMCA '05: Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-2 (CIMCA-IAWTIC'06)*, pages 13–18, Washington, DC, USA, 2005. IEEE Computer Society.

[2] Dragos Arotaritei and Sushmita Mitra. Web mining: a survey in the fuzzy framework. *Fuzzy Sets and Systems*, 148(1):5–19, 2004.

[3] Bezdek, J.C. *Pattern Recognition and Fuzzy Objective Function Algorithms.* Plenum Press, N. York, 1981.

[4] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: a new approach to topic-specific web resource discovery. *Comput. Networks*, 31(11-16):1623–1640, 1999.

[5] Michael Chau, Daniel Zeng, Hsinchun Chen, Michael Huang, and David Hendriawan. Design and evaluation of a multi-agent collaborative web mining system. *Decis. Support Syst.*, 35(1):167–183, 2003.

[6] Hang Cui, Ji-Rong Wen, Jian-Yun Nie, and Wei-Ying Ma. Probabilistic query expansion using query logs. In *Proceedings of the WWW '02*, pages 325–332, New York, NY, USA, 2002. ACM Press.

[7] Del.icio.us. http://del.icio.us/.

[8] Peng F.C. Ding C., Patra J.C. Personalized web search with self-organizing map. In *The 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service 2005*, pages 144– 147, 29 March-1 April 2005.

[9] Domingos P. Doan A., Madhavan J. and Halevy A. Learning to map between ontologies on the semantic web. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 662–673, New York, NY, USA, 2002. ACM Press.

[10] Dublin Core. http://purl.org/dc/elements/1.1/.

[11] L. Martnez F. Chiclana E. Herrera-Viedma, F. Mata. A consensus support system model for group decision-making problems with multi-granular linguistic preference relations. *IEEE Transactions on Fuzzy Systems*, 13(5):644–658, October 2005. ISSN 1063-6706.

[12] FOAF. The Friend of a Friend (FOAF) project . http://www.foaf-project.org/.

[13] Trombetta A. Montesi D. Gal A., Anaby-Tavor A. A framework for modeling and evaluating automatic semantic reconciliation. *The VLDB Journal*, 14(1):50–67, 2005.

[14] Atanas Kiryakov, Borislav Popov, Ivan Terziev, Dimitar Manov, and Damyan Ognyanoff. Semantic annotation, indexing, and retrieval. *J. Web Sem.*, 2(1):49–79, 2004.

[15] Page Lawrence, Brin Sergey, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.

[16] Janssen J. Liu H. and Milios E. Using hmm to learn user browsing patterns for focused web crawling. *Data & Knowledge Engineering*, 59(2):270–291, 2006.

[17] Loia V., Pedrycz W., Senatore S. and Sessa M.I. Web navigation support by means of cognitive proximity-driven assistant Agents. *Journal of the American Society for Information Science and Technology (JASIST)*, 57(4):515–527, 2006.

[18] Loia V., Senatore S. and Sessa M.I. Similarity-based SLD Resolution its Role for Web Knowledge Discovery. In *Special Issue on "Advances in Possibilistic Logic and Related Issues"*, volume 144, pages 151–171. Fuzzy Sets & Systems, 2004.

[19] Bernstein P. A. Madhavan J. and Rahm E. Generic schema matching with cupid. In *The VLDB Journal*, pages 49–58, 2001.

[20] M. E. S. Mendes and L. Sacks. Evaluating fuzzy clustering for relevance-based information access. In *Proc. of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2003*, pages 648–653, 2003.

[21] Boudriga N. and Obaidat M. S. Intelligent agents on the web: A review. *Computing in Science and Engg.*, 6(4):35–42, 2004.

[22] Pazzani M.J., Billsus D. Adaptive Web Site Agents. *Autonomous Agents and Multi-Agent Systems*, 5:205–218, 2002.

[23] M. F. Porter. An algorithm for suffix stripping. pages 313–316, 1997.

[24] Henzinger M. R. Algorithmic challenges in web search engines. *Internet Mathematics*, 1(1):115–126, 2003.

[25] A. Rosenfeld, R. Hummel, and S. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man and Cybernetics*, 6:420–433, June 1976.

[26] E. Selberg and O. Etzioni. The MetaCrawler architecture for resource aggregation on the Web. *IEEE Expert*, 12(January–February):11–14, 1997.

[27] Shadows. http://www.shadows.com/.

[28] Cyrus Shahabi and Yi-Shin Chen. Web information personalization: Challenges and approaches. In *DNIS*, pages 5–15, 2003.

[29] Barry Smyth, Evelyn Balfe, Oisín Boydell, Keith Bradley, Peter Briggs, Maurice Coyle, and Jill Freyne. A live-user evaluation of collaborative web search. In *IJCAI*, pages 1419–1424, 2005.

[30] SUMO WG. Sumo ontology site, 2003. Avaliable URL: http://ontology.teknowledge.com/#FOIS.

[31] Semantic Web Technology Evaluation Ontology (SWETO). http://lsdis.cs.uga.edu/projects/semdis/sweto/.

[32] C. J. Van Rijsbergen. *Information Retrieval, 2nd edition.* Dept. of Computer Science, University of Glasgow, 1979.

[33] R. Villa, R. Wilson, and F. Crestani. Ontology mapping by concept similarity. In *International Conference on Digital Libraries (ICDL 2004)*, February 2004.

[34] Vivissimo. http://www.vivissimo.com.

[35] WiseNut. http://www.wisenut.com/.

[36] Chen H. Yang C.C., Yen J. Intelligent internet searching agent based on hybrid simulated annealing. *Decision Support Systems*, 28:269277, 2000.