Control and Cybernetics

vol. **39** (2010) No. 4

Matrioshka's soft approaches to personalized web exploration*

by

Gloria Bordogna[1] and Giuseppe Psaila[2]

[1] CNR-IDPA
Via Pasubio 5, I-24044 Dalmine (BG), Italy
[2]Università di Bergamo, Facoltà di Ingegneria,
Viale Marconi 5, I-24044 Dalmine (BG), Italy
e-mail: gloria.bordogna@idpa.cnr.it, psaila@unibg.it

Abstract: In this paper, we present the soft approaches and techniques developed within *Matrioshka*, a prototypal meta-search system aimed at providing personalized exploratory facilities, addressing the well known *Ranked-List Problem*. *Matrioshka* implements a novel *interaction framework* that provides tools for clustering documents retrieved by search engines, tools for exploring the content of clusters through the analysis of some cluster properties, tools for generating disambiguated queries from the clusters, and tools for combining the clusters to highlight their shared contents. All these tools are defined based on soft operations, in order to deal with intrinsic semantic ambiguity, imprecision and uncertainty of complex web searches. In this way, the user is supported in the deployment of complex web search exploratory activities.

1. Introduction

Users of a web search engine expect to find their favorite results in the first positions of the ranked list provided by the search engine. Unfortunately, this happens only for a limited number of situations. Very often, the query submitted by the user does not yield the desired results in the first positions, for several possible reasons: the query is too generic; there is a problem of semantic ambiguity of terms in the query; the pages of interest are not sufficiently linked by other web pages (for example, *Google* exploits, among the others, the *page rank* method, Brin and Page, 1998).

Consequently, the desired pages might be very far away from the top positions of the result lists, due to the large amount of pages indexed and retrieved

*Submitted: September 2010; Accepted: November 2010

by a search engine. Thus, scattered among hundreds and hundreds of useless web page references, they become practically unreachable, since users usually disregard documents appearing after the second page of the result list. This phenomenon is known as the *ranked list problem*.

Typically, users try to overcome the problem by following two distinct behaviors: some of them choose to submit the same query to several search engines, trying to somehow fuse the results; others prefer to search the Web by iterative trial and error cycles of query reformulation (Notess, 2006). In both cases, the retrieval of distinct and numerous lists of documents produces, as a major effect, information overload: it may happen that almost the same documents appear in the first positions of all the result lists. Furthermore, it is hard to perform an effective comparison of the different lists. Automatic techniques, such as document diversification and disambiguation (Gollapudi and Sharma, 2009; Carbonell and Goldstein, 1998) have been proposed to cope with the ranked list problem, but they are opaque techniques that the user cannot control nor is aware of the fact that they are applied for ranking documents.

Observing in detail the aforementioned behaviors, we noticed that they are two sides of the same coin: they are steps of a complex exploratory process aimed at discovering interesting and hidden documents, i.e., those relevant for the actual needs of users in a specific context at a specific time. Suitable software tools should be made available, able to effectively support the process. Soft techniques can be adopted; in fact, they are able to deal with semantic ambiguity and intrinsic imprecision and uncertainty of complex web searches.

To this aim, we developed several soft techniques for assisting and enhancing the web search exploratory process. These techniques are collected in the *Matrioshka*[1] meta search system (Bordogna et al., 2008a,b, 2009b). *Matrioshka* implements a novel *interaction framework* that relies on clustering of ranked lists (a typical approach for meta search tools, Carpineto et al., 2009; Chen and Dumais, 2000). It provides tools for clustering documents retrieved by search engines, tools for exploring clusters through a suitable user interface, tools for generating disambiguated queries from the clusters, and tools for manipulating clusters of similar documents.

In this paper, we present the soft approaches and techniques for meta web search developed within *Matrioshka*. After a discussion about related work (Section 2), we introduce the basic concepts that characterize the interaction framework (Section 3). At this point, Section 4 presents the methods for exploring clustered results: in particular, the diverse ranking methods for clusters and the preference-based technique for flexibly combining ranking methods. Section 5 explains our technique for generating disambiguated queries. Section 6 defines the operators provided by the interaction framework (and implemented within *Matrioshka*) to manipulate clusters, discussing an example. Section 7 reports some real tests performed with *Matrioshka*, in order to show the effectiveness

[1] `matrioshka.unibg.it`

while performing complex exploratory searches. Finally, Section 8 draws the conclusions.

2. Related work

Sanderson reports that from 7% up to 23% of Web searches in query logs consists of less than three terms only (Sanderson, 2008). Not surprisingly, with so little information a common experience of users when analyzing the ranked list of results is the inability of finding the information that suits their purposes (Jansen et al., 2009; Jansen and Spink, 2006).

Several papers have been published on the topic of coping with the ranked list problem. One approach is query disambiguation based on the extraction of terms from searched documents: the idea is to replace user-provided query terms with more specific ones, to narrow the search to the context users have in mind (Lawrence, 2000; Teevan et al., 2008). In order to obtain these terms, some approaches exploit external knowledge, such as *Wordnet* (Voorhees, 1993; Luca and Nurnberger, 2005) and existing taxonomies (Ma et al., 2007), or users' models (Shen et al., 2005; Chirita et al., 2007; Zhang, 2008; Liu et al., 2004) to normalize the meaning of the search terms in distinct contexts (Fellbaum, 1998; Patwardhan et al., 2005). Other approaches are based on query log analysis over long periods of time on the server side (Sugiyama et al., 2004; Sun et al., 2005). In an attempt to skip the privacy concerns that these last proposal rise, a recent approach determines user intention based on the analysis of short query sessions (Mihalkova and Mooney, 2009). Other methods apply collaborative analysis evaluating the similarity between query logs of distinct users and corresponding appreciated results (Freyne et al., 2004; Balfe and Smyth, 2005). Nevertheless, these techniques suffer from the start up problem, when no recommendations are available.

The approaches which generate groups of terms based on either terms co-occurrence analysis (Liu et al., 2006), or dynamic clustering of query results (Roussinov and Chen, 2001; Zamir and Etzioni, 1999; Cutting et al., 1992) to identify the distinct contexts of terms, are more similar to our proposal, in which dynamic clustering of search results is applied as well. However, differently from the previous approaches, the goal is to optimize the whole iterative process of query reformulation, by aiding the user to submit more specific disambiguated queries, and favoring the discovery of novel documents focused on the retrieved topics of interest.

The issue of the ranked list problem has also been treated through by *document diversification*, which consists in providing novel contents in the top ranked results (Agrawal et al., 2009; Radlinski et al., 2009; Gollapudi and Sharma, 2009). The diversification of documents is based on the comparison of documents one against another, assuming that similar documents deal with similar topics. Then, to reduce the redundancy of top-ranked results, documents are ordered based on a combination of their diversity and query relevance, which in

Carbonell and Goldstein (1998) and Zhai et al. (2003) is the *Maximal Marginal Relevance* (MMR) criterion, in Zhang et al. (2005) is the *affinity ranking*, that consider richness as well. Novelty detection has been investigated also in relation with *automatic text summarization* (Sweeney et al., 2008) and in *query reformulation*. In this latter case, by submitting similar reformulations of the original query, a user aims at obtaining some novel Web pages in the first positions, focusing on the interesting topics (Lad and Yang, 2007; Xu and Chen, 2006).

In our proposal, we define the *novelty* metric; it is related to clusters of documents, and measures the amount of new documents contained in a cluster w.r.t. documents previously seen and appreciated by the user. Document diversification can be achieved by allowing the user to choose a *personalized ranking* of clusters, based on a balanced combination of two properties, for example novelty and query similarity (Dubois and Prade, 1985). The joint use of suggested disambiguated queries, and a personalized ranking of clusters aids users in discovering new relevant documents, thus reducing the need for long iterative cycles of query reformulation.

Within *Matrioshka*, several operators were developed to explore the shared contents of distinct clusters. We extensively studied their features and presented them in several previous works (Bordogna et al., 2009b, 2009a); their main features are reported in Section 6.2.

A motivation of the utility of operators to explore clusters can be found in Leouski and Croft (1996): users need tools that give them more immediate control over the clusters of retrieved web documents; such tools should serve as means for exploring the similarity among documents and clusters. Leouski and Croft (1996) also suggests to give users some means to correct, or change, the classification structure.

To support the manipulation of clusters, Leouski and Croft (1996) suggests the development of graphic user interfaces. Indeed, the literature on visual paradigms for the presentation of textual search results is too extensive to review; for a survey, the reader can see Card et al. (1999) and Staley and Twidale (2000). One goal of these approaches is to perform some kind of text mining based on conceptual map visualization (Chung et al., 2003; Kampanya et al., 2004). Nevertheless, our proposal is different: we do not exploit a graphical representation of relationships between documents at this level, but we provide a *language* for *flexibly exploring* the hidden relationships.

In the *NIRVE* prototype (Sebrechts et al., 1999), the authors evaluate and compare several graphical interfaces for showing the retrieved results of *NIST*'s *PRISE* search engine. In their conclusions, they state that *"a good visualization of search results depends on a good structure and what often happens is that developers perform a deeper analysis of results in order to generate that structure"*. In this respect, we envisage that our proposed language could be employed for exploring and finding a good structure of results, that can then be presented by taking advantages of the proposed graphic visualization techniques.

An approach that shares some similarity of intent with our proposal, is the *Scatter/Gather* algorithm (Hearst and Pederson, 1996). Its distinctive feature is the way clusters can be selected, recombined (gathered) and re-clustered (scattered). However, the user has to decide which clusters have a relevant theme based solely on keywords and titles. No functionality is available to detect the degree of overlapping between clusters, and to explore the cluster contents by evaluating some cluster property, as in our proposal. Furthermore, since new clusters are generated based on re-clustering, the generation criteria remain implicit and unknown to the user. On the contrary, in our approach, the user can actively control the cluster contents in several ways by the application of operators such as intersection and union.

3. The interaction framework

In order to understand how soft approaches are applied in *Matrioshka*, the *interaction framework* must be introduced. First of all, the basic concepts are defined, then the exploratory process is described.

3.1. Basic concepts

Here we describe the basic concepts on which the proposed interaction framework implemented by *Matrioshka* is based. We start by considering a *query q* submitted to a search engine; its result is a *ranked list of documents*; in the following, we call *item* a document in a ranked list.

DEFINITION 1: **Item** *An* item *i represents (an instance of) a document retrieved by a web search. It is described by the following tuple of attributes of* $i : \langle \text{Uri}_i, \text{Title}_i, \text{Snippet}_i, \text{Irank}_i \rangle$. Uri_i *is the* Uniform Resource Identifier *(Coates et al., 2001) of the ranked web document;* Title_i *and* Snippet_i *are, respectively, the document title and snippet*[2]*; finally,* Irank_i *is a score (in the range* $[0, 1]$*) that expresses the estimated relevance of the retrieved document w.r.t. the query, and it is computed as defined in Section 4.1.*

The same document (web page) may be represented by distinct items in distinct result lists. In fact, we assume that a document is uniquely identified by its Uri_i, while it may have distinct $Title_i$, $Snippets_i$ and $Irank_i$, when retrieved by different search engines (or by different queries). We assume that $Irank_i$ is a function of the position of the item in the query result list.

In *Matrioshka*'s interaction framework, the results of a user request (or exploration) are not simply a ranked list of documents, but they are gathered in *clusters*.

DEFINITION 2: **Cluster** *A* cluster *c is a set of items, having a rank. It is defined by the tuple* $c : \langle \text{Label}_c, \text{Content}_c, \text{Rankings}_c, \text{Crank}_c \rangle$. Label_c *is a set of*

[2]The snippet is an excerpt of the document, made by a set of sentences that may contain the keywords of the query

terms that semantically synthesize the main content of the cluster; Content_c *is the set of items associated with the cluster;* Rankings_c *is a pool of values that measure (in the range* $[0,1]$*) different properties of clusters, while* Crank_c *is a user-defined combination of measures in* Rankings_c *(measures in* Rankings_c *and the way* Crank_c *is computed will be presented in Section 4).*

At this point, we define the main element of the data model.

DEFINITION 3: **Group** *A* group *g is a non empty, ordered set of clusters. It is described by the pair of attributes* $g : \langle \text{Label}_g, \text{Clusters}_g \rangle$. Label_g *is a set of terms that semantically synthesize the main content of the group;* Clusters_g *is the set of clusters belonging to the group.*

For a group obtained by clustering the result list provided by a search engine, $Label_g$ is the text of the query submitted to the search engine.

3.2. The exploratory process

Matrioshka supports the exploratory process. Since this process can last for long periods, the interaction framework provides a *Process Repository*, in order to store the intermediate results of the exploration.

The atomic elements stored in the process repository are groups, both obtained by the original queries and derived by applying group manipulation operators.

Fig. 1 shows a toy example of exploratory process to illustrate the idea. Consider a sample user named *Jane*. Suppose that Jane submits a query to *Google*: the result list is processed and the group of clusters g_1 is obtained. Consulting the clusters and the suggested disambiguated queries (one for each cluster), Jane decides to submit two queries among the suggested ones to *Google*; groups g_2 and g_3 are obtained.

Again, by consulting clusters in group g_3, Jane finds that one suggested query properly matches her feelings about the needs of the exploration, and submits this query to *Google* obtaining group g_4.

At this point, Jane is quite satisfied with the results, but thinks that it might be possible to further refine the precision of the results, by considering the contribution provided by other search engines, i.e., *Yahoo!* and *Bing*. Thus, Jane submits the same query which g_4 originated from (suggested by *Matrioshka*) to *Yahoo!* and *Bing*, obtaining groups g_a and g_b, respectively.

By applying the *Group Join Operator* (see Section 6.2), clusters in these two groups are fused together if they have common documents; group g_c is obtained.

Finally, by applying the *Group Refinement Operator* (see Section 6.2), Jane refines clusters in group g_4 (whose documents come from *Google*) on the basis of documents in clusters in group g_c, that come from *Yahoo!* and *Bing*. This way, only documents provided by the three considered search engines are trusted by Jane.

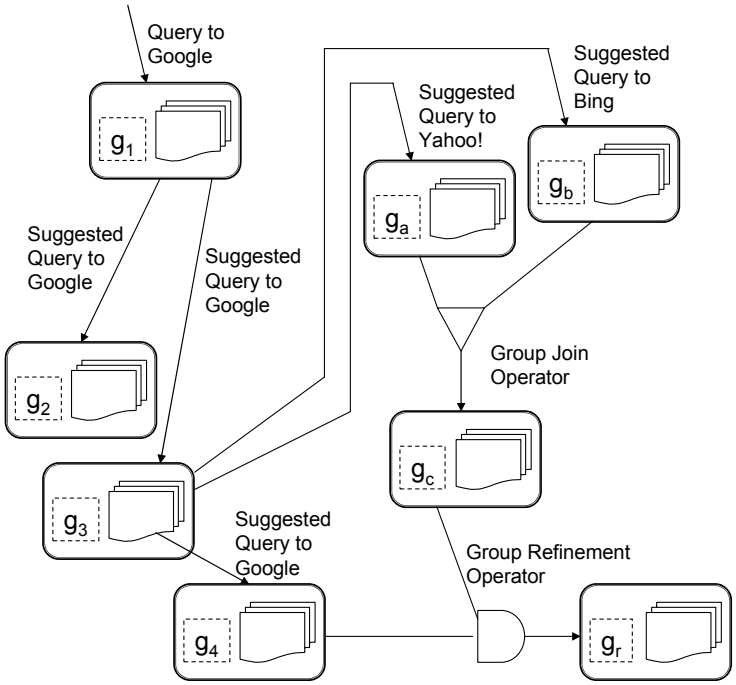

Figure 1. Sample exploratory process

The whole process can be performed through the Graphic User Interface of *Matrioshka*, shown in Fig. 2. As the reader can see, the user can choose the search engine, can specify the maximum number of retrieved documents, can consult the groups of clusters and explore the content of clusters; finally, each cluster is associated with the disambiguated query and with a set of ranking measures, that will be discussed in Section 4.

In parallel with the *Process Repository*, *Matrioshka* provides the *History Repository*. This is automatically updated by the system when the user decides to submit a suggested query: documents contained in the cluster associated with the suggested query are stored in the *History Repository*, in order to assist the further computation of disambiguated queries (see Section 5).

The *History Repository* is defined as follows.

DEFINITION 4: **History Repository** *The* History Repository H *is defined as the set* $H = \{(d_1, \mu_H(d_1)), \ldots, (d_h, \mu_H(d_h))\}$ *of pairs* $(d_i, \mu_H(d_i))$*, where* d *is a document URI, and* $\mu_H(d)$ *is the membership degree of* d *in the History* H*.*
At the beginning of a search session, it is empty; it is updated whenever a user submits a suggested query to a search engine with the ranked items belonging to the associated cluster.

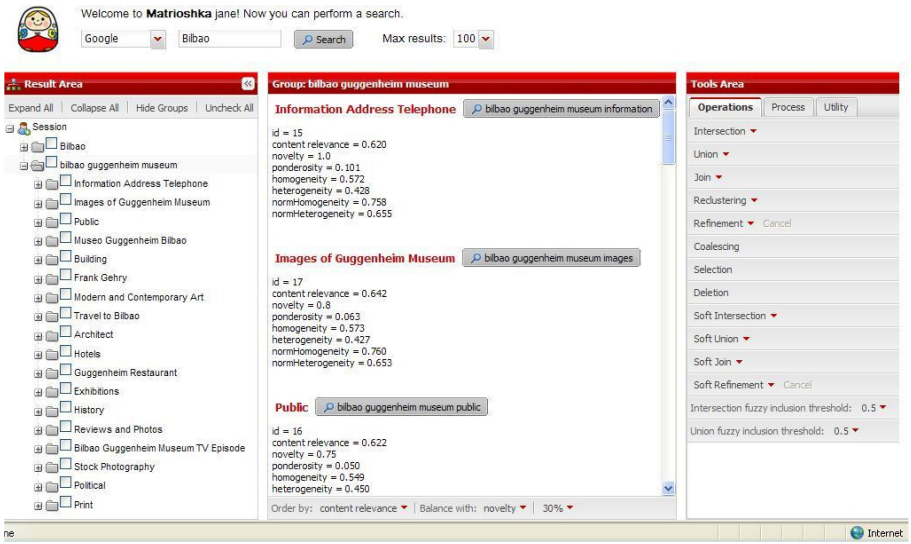

Figure 2. The *Matrioshka* User Interface

4. Exploring clustered results

The concepts defined in the previous section are the basis on which our soft approaches rely. In this Section, we extensively present how clusters are generated, their soft properties, and the flexible combination of cluster properties that the user can exploit to re-rank clusters in a dynamic way.

4.1. Generation of query-based groups

The exploration process starts with the generation of a *query-based group of clusters*, i.e., a group whose clusters are obtained by clustering the ranked list provided by a search engine (e.g., *Google*, *Yahoo*!, *Bing*, etc.).

A web search engine provides a ranked list of results: each item in the list describes a document by means of its URI (Uniform Resource Identifier), title and snippet (an excerpt of the document content).

The steps performed to cluster the items in the result ranked list are the following.

- *Computation of the Item Search Engine Relevance.* The search engine relevance rank $Irank_i$ (similarity rank w.r.t. the query) for each item is computed, based on the position of the document in the result list. The rationale is to achieve independence of the actual ranking score computed by the particular search engine. Hereafter the definition is:

$$Irank_i = \frac{N - Pos(i) + 1}{N}$$

where $Pos(i)$ is the position of item (document) i in the query result list as provided by the search engine, and N is the size of the list. In this way, the first ranked document gets the value $Irank_i = 1$, while the last one takes the value $Irank_i = 1/N$.
Notice that the user can set the value for N, i.e., the maximum number of documents to ask the search engine for.

- *Word Expansion, Stop-words Cleaning and Indexing.* For each item i, terms in its titles $Title_i$ and snippets $Snippet_i$ are expanded with synonyms, while Stop-words are removed.
  Finally, the resulting texts are indexed by means of an inverted index data structure. In particular, *Matrioshka* exploits the indexing functions provided by the *Lucene* library.
  The expansion operation, implemented based on the *WordNet* thesaurus, increments the number of common terms among the retrieved items, and is done to favour the successive clustering phase. Thus it is a critical issue and permits to realize a kind of semantic web search results clustering, like in the recent paper by Sameh and Kadray (2010).
- *Cluster Generation.* The items are clustered, based on the expanded and cleaned titles and snippets, indexed in the *Lucene* inverted index.
  Matrioshka exploits the *Lingo* algorithm (Osinski, 2003). *Lingo* is a multilingual clustering algorithm, provided by the *Carrot2* libraries; it is tightly integrated with the *Lucene* library. This algorithm generates a flat partition of the whole documents retrieved by a search, on the basis of the Singular Value Decomposition technique.
  The label of a cluster must help the user to understand the main topic of the cluster. Since clusters are built in such a way, clustered documents are similar as far as the content of title and snippet is concerned, the most relevant document w.r.t. the query is the one that best represents the main topic. For this reason, we decided that $Label_c = Title_I$, i.e., the title of item $I \in Content_c$ which is the most relevant item in the cluster.
- *Group Generation.* The clusters so far generated are considered as belonging to a single group: such a group is labeled with the query q.
  Since several queries can be submitted, at a given time several groups can be present in the Process Repository.

Fig. 2 presents the main GUI window of *Matrioshka*. In the top pane, it is possible to see the text field where to write the query, the list of available search engines (*Google* is the default one), the field to specify the maximum number of retrieved documents.

In the left pane, the groups of clustered retrieved documents are presented in a tree fashion: each group contains clusters and each cluster contains documents. The central pane shows either clusters in the selected group (as in the figure) or the documents in the selected cluster.

Finally, the right pane presents the operators for comparing groups of clusters, such as intersection, join, refinement and many others (see Section 6.2).

4.2. Cluster ranking methods

An important issue for which soft approaches are really effective, is the evaluation of properties/qualities of clusters by means of ranking measures.

For each cluster, four different properties are evaluatedů (ranked). Each of them gives the user information about some quality of the cluster of documents. Through the user interface, the user can decide to sort clusters on the basis of a different property and with different sorting criteria, to get in the first positions of the list the stronger/weaker clusters w.r.t. the desired property.

- *Search Engine Relevance*: Rel_c is defined as the average of the relevance scores of items belonging to the cluster; the relevance scores of items are the $Irank_i$ values computed as previously defined from the document positions in the ranked list returned by the search engine.
  Ordering clusters by decreasing values of their search engine relevance means being primarily interested in the search engine assessed relevance of documents in the clusters w.r.t. the query submitted to the search engine.
- *Ponderosity*: Pon_c is defined as the cluster relative cardinality, and it measures the percentage of documents that belong to the cluster w.r.t. all the documents in the group which the cluster belongs to.
  Sorting clusters in decreasing order of their ponderosity can be useful for users interested in high recall.
- *Homogeneity*: Hom_c is defined as the complement of the average distance of the documents vectors $\overline{d}_i$, represented in the space of index terms extracted from their titles and snippets, and weighted by their relative frequency, w.r.t. the cluster centroid vector A (defined as the average vector of all the documents vectors belonging to the cluster):

  $$Hom_c = \frac{\sum_{i=1\ldots|c|} sim(d_i, A)}{|c|}$$

  where *sim* is the cosine similarity measure. It is obvious that $Hom_c \in [0, 1]$.
  The smaller the homogeneity (i.e., the greater the variance), the more heterogeneous is the cluster: choosing to sort clusters in the descending order of their homogeneity means being interested in contents focused on the specific meaning expressed by the label of the cluster, since the cluster label is generated from its centroid vector. This can be useful in target searches.
  Conversely, choosing to sort clusters in the ascending order of their homogeneity (i.e., decreasing order of heterogeneity) means being more tolerant

with respect to the meaning expressed by the cluster label; this can be useful when one is unsure whether the query expresses the actual information needs and wants to soften the selection conditions.

- *Novelty*: Nov_c, the novelty of a cluster c is the proportion of new documents in cluster c w.r.t. those present in the *History Repository* H. A document is added to the *History* repository when the user selects the cluster c which it belongs to, and submits the disambiguated query generated from c. Nov_c is defined as follows.

$$Nov_c = \frac{|c - H|}{|c|} = \frac{\sum_{d \in Content_c}(\mu_c(d) - \mu_H(d))}{\sum_{d \in Content_c} \mu_c(d)}$$

H is defined to represent the (weighted) set of documents retrieved by past queries and constituting the content of the *History* repository. The membership degree of document d in H is $\mu_H(d) \in \{0, 1\}$.[3]
Consider now $\mu_c(d)$, denoting the membership degree of document d in cluster c. $\mu_c(d) = 1$ if d belongs to cluster c (i.e., $\mu_c(d) = 1$ if $\exists d_{c_i} \in Content_c \mid d \equiv d_{c_i}$); otherwise, $\mu_c(d) = 0$. At the beginning of the exploratory process, H is empty; thus, $Nov_c = 1$ for the first generated cluster (its documents are all completely new). In successive phases, Nov_c for a newly generated cluster possibly decreases, and might be very small when many documents in the new cluster are already present in H.
Computing the cluster novelty can be very useful in letting new documents emerge in the exploration process. Sorting clusters in the decreasing order of novelty ranking is useful when the user is interested in new documents on the topics of a search; thus, Novelty can be useful in the context of bibliographic surveys.

We are now ready to complete the definition of clusters, by specifying the $Rankings_c$ attribute as a tuple of the above defined ranking measures.

$$c = \langle Label_c, Content_c, Rankings_c : \langle Rel_c, Pon_c, Hom_c, Nov_c \rangle, Crank_c \rangle$$

Attribute $Crank_c$ is the membership degree of the cluster, that is computed based on a combination of ranking measures. When the cluster is generated, its default value is the *Search Engine Relevance Score* Rel_c.

Notice that in defining the data model for clusters we have drawn experience from fuzzy databases, in which each tuple has a special attribute which is its membership degree, taking values in $[0, 1]$, that can be used to rank the tuples; in our interaction framework, $Crank_c$ plays a similar role.

In *Matrioshka*, the user can dynamically change the way $Crank_c$ is computed: consequently, its value is recomputed on the fly. In this way, the user can

[3] Note that here we consider a document (identified by its *Uri*) and not an item: in fact, a document appearing in several clusters is described by different items. In contrast, we need to deal with novelty of documents w.r.t. past queries/groups.

fully exploit the potentiality provided by soft approaches of in depth analyzing clusters from different perspectives.

4.3. Flexible combination of ranking measures with preference

During the analysis of results, the user might be interested in ranking the clusters based on a combination of two properties.

We adopted a flexible soft approach: the user is provided with a tool by means of which two properties can be combined; a preference of one property w.r.t. the other can be specified. In other words, the User Interface permits to select two distinct properties, by specifying which of them is preferred w.r.t. the other. As a result, for each cluster, a satisfaction degree, that becomes the new value for attribute $Crank_c$, is computed, and clusters are sorted in ascending or descending order of $Crank_c$ (based on the user's choice).

Consider two properties R_c^1 and R_c^2 chosen among Rel_c, Pon_c, Hom_c and Nov_c, with R_c^1 preferred on R_c^2. The $Crank_c$ of each cluster c is computed based on a prioritized fuzzy anding of the two properties (Dubois and Prade, 1985).

$$Crank_c = Tnorm(Tconorm(1-\lambda, R_c^2), R_c^1) \in [0,1]$$

where *Tnorm* and *Tconorm* are a T-norm and a T-conorm operator, respectively, defined as *min* and *max*.

This combination is based on a preference degree $\lambda \in [0,1]$, that defines the user determined preference for R_c^2 w.r.t. R_c^1. By varying the value of λ, the combination can be balanced in different ways.

If $\lambda = 0$, the property expressed by R_c^2 is considered irrelevant; thus, $Crank_c$ is computed by considering solely the property expressed by R_c^1:

$$\min(\max(1-0, R_c^2), R_c^1) = \min(1, R_c^1) = R_c^1.$$

On the contrary, $\lambda = 1$ means that the property expressed by R_c^2 has the same importance as the property expressed by R_c^1, and in this case the $Crank_c$ of a cluster is computed based on the satisfaction of both measures. The simultaneous satisfaction of the two criteria is guaranteed by modeling the aggregation function through a T-norm operator: we chose the *min* function, corresponding to the fuzzy AND operator.

$$\min(\max(1-1, R_c^2), R_c^1) = \min(R_c^2, R_c^1).$$

In the case, in which $0 < \lambda < 1$, the property expressed by R_c^2 is taken into account in computing $Crank_c$ only if it is above the threshold $1-\lambda$, while, for lower values, $Crank_c$ is computed solely based on R_c^1.

To clarify, consider as R_c^2 the novelty degree Nov_c and as R_c^1 the search engine relevance degree Rel_c. On this basis, notice that the novelty threshold is the minimum value above which the *novelty* of a cluster starts becoming

meaningful to the user. It is inversely proportional to the preference degree of the novelty, so that the greater is the preference of novelty for the user, the smaller is the novelty threshold that user can accept in order to have the relevance computed solely based on the cluster search engine relevance w.r.t. the query. Of course, similar considerations hold for any combination of other properties. In Fig. 2, the central pane shows the retrieved clusters. At its bottom, it is possible to see that the user has chosen to combine the *search engine relevance* with the *novelty* by a preference degree of 0.3 (corresponding to 30 %).

5. Generation of disambiguated queries

For each cluster the system generates the disambiguated query. This technique is based on a mix of soft approaches, since it is necessary to balance the importance of several aspects, in order to extract the most representative and significant words from titles and snippets of clusters.

The generation of disambiguated queries is a two-step process.

Step 1: Extraction of cluster candidates terms. The first step of the process extracts r most representative terms associated to each cluster c, denoted as:

$$Re_c = < t_{1,c}, tf_{1,c} >, ..., < t_{j,c}, tf_{j,c} >, ..., < t_{r,c}, tf_{r,c} > .$$

This step is crucial in the process, since the quality of the disambiguated queries strongly depends on the quality of these terms. We tried several definitions of the term representativeness weights $tf_{j,c}$, and we found that the best results were obtained by defining $tf_{j,c}$ based on the following multiple criteria.

- The first criterion is the *absolute frequency* $occ_{t,c}$ of term t in cluster c, defined as the number of occurrences in the snippets and titles of the documents in c, differently weighted: assuming that titles are a more objective representation of the document contents than snippets, a single occurrence in the title is weighted twice the single occurrence in the snippet. The absolute frequency is normalized w.r.t. the maximum frequency $max_{k \in Content_c}(occ_{k,c})$ of the terms extracted from cluster c. In this way, the relative frequency of a term in a cluster is computed as follows:

$$rel\_freq_{t,c} = \frac{occ_{t,c}}{\max_{k \in Content_c}(occ_{k,c})} .$$

- The second criterion is $|D_{t,c}|$, the *number of documents* of cluster c, *containing the term* t. In order to be a representative of a cluster, a term must be present in most of the documents of the cluster, not just in a few of them. To evaluate the satisfaction of this qualitative criterion, we impose a fuzzy restriction on the cardinality of set $D_{t,c}$, i.e., $|D_{t,c}|$. This

fuzzy restriction is specified by a fuzzy quantifier *most* defined empirically by a monotonic non decreasing function *most*: $[0,1] \rightarrow [0,1]$.

$$most(x) = \begin{cases} 1 & \text{if } x > M \\ (x-m)/(M-m) & \text{if } m < x \leq M \\ 0 & \text{if } x \leq m \end{cases}$$

m is the limiting share of documents below which a term d is considered not at all representative of a cluster c. M is the lower-bound on this share considered sufficient, for a term t, to fully represent the majority of documents in a cluster c.
We then compute $most(\frac{|D_{t,c}|}{|Content_c|})$ as the satisfaction of this majority criterion. Experimentally, we tried out several settings for the values m and M and achieved the best results with $m = 0.5$ and $M = 0.7$. With these settings the fuzzy quantifier *most* states that if 70% of the documents of a cluster contain term t, then the fuzzy restriction it defines is completely satisfied, i.e., t is a candidate representative term of the cluster; when t is contained in less than 50% of the documents of the cluster, the restriction is not satisfied at all; for percentage values between 50% and 70%, the satisfaction increases linearly.

- The third criterion is the *Inverse Cluster Frequency* (ICF_t) of a term t. It is computed to estimate the specificity of the term for clusters, i.e., its ability to distinguish the contents of a cluster w.r.t. the other ones. It is defined as follows:

$$ICF_t = \log \frac{(|Clusters_g|)}{|C_t|} .$$

$|Clusters_g|$ is the total number of clusters in the group and $|C_t|$ is the number of clusters containing t ($C_t \subseteq Clusters_g$). ICF_t is maximum when the documents containing t are only in one cluster. The greater is the number of clusters containing t, the smaller is the ICF_t.

These three criteria are aggregated by a product to compute the *representativeness* degree $tf_{t,c}$ of term t in cluster c:

$$tf_{t,c} = (rel\_freq_{t,c}) \times most\left(\frac{|D_{t,c}|}{Content_c|}\right) \times \left(\frac{ICF_t}{MaxICF_t} + 0.5\right) .$$

$MaxICF_t$ is the greatest among the values ICF_t.

The first r terms with greatest $tf_{t,c}$ values are finally selected to represent the meaningful topics dealt with by the documents of cluster c. In the remote case of ties, where more than r terms have the same representativeness, we select them in alphabetical order. In this aggregation, the contribution of the inverse cluster frequency is lower-bounded to 0.5 to favor the selection of the query terms as representative terms. In fact, in spite of the fact that the query terms

have null inverse cluster frequency, since they appear in all the snippets of all the retrieved documents, most of the times they are selected as representative terms because they have a high relative term frequency in the document cluster. On the contrary, if a term has a null inverse cluster frequency, is rare in the documents of the cluster, and appears in just a few of the cluster documents, it is likely to be discarded.

Step 2. Generation of disambiguated queries. For a cluster c, the disambiguated query qe_c is generated as a subset of Re_c, its set of candidate terms generated in *Step 1*.

We assume that Re_c represents the main contents of the cluster c as accurately as we can. As a consequence, submitting a query (that is a conjunction of all the terms in Re_c) to a search engine would retrieve mostly documents focused on the contents of cluster c.

To focus the search on the same contents of cluster c, trying at the same time to favor the retrieval of new documents, we generalize the representation of the cluster contents by generating a disambiguated query qe_c that contains a number n_c of terms in Re_c inversely proportional to the *homogeneity* (thus, directly proportional to the *heterogeneity*) of cluster c, i.e., Hom_c. In fact, the greater the heterogeneity is, the more of content is represented in the cluster; then, in order to represent all of it, the longer the query must be.

The function that determines the number of terms n_c to select from Re_c to generate qe_c is defined as:

$$n_c = \mathit{length}(q) + \mathit{round}(1/Hom_c)$$

where $\mathit{length}(q)$ returns the number of terms in the original query, $\mathit{round}(x)$ approximates x to the closest integer value.
Once the number of terms n_c is determined, qe_c is generated as the ordered set of terms consisting of the first n_c most representative terms belonging to the candidate set Re_c as follows:

$$qe_c = (t_{1,c}, ..., t_{n_c,c})$$

where $t_{j,c} \in Re_c$ if its representativeness weight $tf_{j,c}$ is the j-th maximum value among all the $tf_{1,c}, \ldots tf_{r,c}$.

The terms in Re_c with greatest representativeness weights are those in common with the query of the previous cycle which are retained in qe_c. In this way the terms of the original query q are retained in the disambiguated queries of the first iteration, while they may be discarded in further iterations. So the procedure generates more specific disambiguated queries of the original ambiguous query at the first iteration cycle, while in subsequent iterations it may generate disambiguated queries expanding the meanings of the original one.

In Fig. 2, the central pane shows clusters with their properties; on the right hand side of the cluster titles, the grey areas contain the disambiguated queries.

In particular, we sent the query *Bilbao* to Google, obtaining the first group of clusters in the left pane. For cluster labeled *Museum Guggenheim*, the system proposed the disambiguated query *Guggenheim Museum Bilbao*, that we clicked on, obtaining the second group whose clusters are shown in the central pane. In practice, starting from a generic query, we were suggested a more focused and unexpected query.

6. Combining groups of clusters

Matrioshka provides users with the possibility to interact with the results of search services organized in groups of clusters, in order to get more satisfactory and refined results to their needs. To this aim, the user can choose to apply different sequences of operators on selected groups, in order to recombine (modify, explore) their structure and content.

The operators that we are going to illustrate are formally defined in Bordogna et al. (2009b); they are inspired by the operators provided by the Relational Algebra (i.e. intersection, join, union etc.), though they are specifically defined for groups of clusters.
They generate, starting from two input groups g_1 and g_2, one group g' that may contain one or more clusters; it can also be empty, in case no common items are detected.

6.1. Basic operations

First of all, we describe two basic operations that combine items belonging to two input clusters to get a new cluster.

We define two basic operations: **cluster intersection** and **cluster union**. They work on the *uri* of the items of two input clusters, assuming that *uri* is the unique identifier of a document. The rationale of this assumption is the fact that the same document, retrieved by two different search services, may have different title and snippet, maintaining, though, the same *uri*. Consider the intersection of two clusters c_1 and c_2, denoted as:

$$c' = \mathit{ClusterIntersection}(c_1, c_2).$$

The *Irank* of an item $i' \in c'$, the cluster resulting from the intersection, is defined as the minimum *Irank* value of $i_1 \in c_1$ and $i_2 \in c_2$ (such that $i_1.uri = i_2.uri$).[4] In the case of cluster union, denoted as

$$c' = \mathit{ClusterUnion}(c_1, c_2),$$

the *Irank* of i' is the maximum *Irank* value of i_1 and i_2 (such that $i_1.uri = i_2.uri$), assuming that $i_1.Irank = 0$ (resp. $i_2.Irank = 0$) when no item with that

[4] This definition is consistent with the definition of the intersection operation between fuzzy sets (Zadeh, 1965).

uri belongs to c_1 (resp. c_2).[5] In both cluster intersection and union, the *title* and the *snippet* of the resulting items are obtained by selecting either $i_1.title$ or $i_2.title$, and either $i_1.snippet$ or $i_2.snippet$, respectively.

In particular, to obtain the *title* and the *snippet* of the items belonging to the clusters of the resulting groups we select as resulting *title* and *snippet*, those belonging to the document having the smallest (in the case of Cluster Intersection) or the greatest (in the case of Cluster Union) value of *Irank*, without making any changes. The rationale of this choice is the fact that in the aggregation based on the intersection (union), we want to represent the document by its worst (best) representative, in accordance with the modelling of the AND and the OR within fuzzy set theory.

6.2. Group operators

The basic operations among clusters are the basis for the definition of operators that combine and generate groups.

- **Group intersection** Group intersection is defined to support the straightforward wish of users to intersect clusters in two groups, in order to find more specific clusters. The assumption is that the more search engines (or the more distinct queries) retrieve the same document, the more the document content is worth analyzing.

  DEFINITION 5: *The* Group intersection *operator generates a new group composed of all the combination of clusters in the original groups having a non empty intersection.*
  Given g_1 and g_2, the groups of clusters to intersect, the resulting group g' is composed of all the clusters c' such that: $c' = \text{ClusterIntersection}(c_1, c_2)$ *with* $|c'| \neq 0$.

- **Group join** A key operator of the language, closely related to the previous one, is the *Group join*. It lets the user expand the original clusters in a group with clusters, possibly belonging to another group, that share one or more documents. The *Group join* operator can be used to explicit indirect correlations between the topics represented by the clusters in the two input groups. The basic idea underlying its definition is that if two clusters have a non empty intersection (i.e. have some common items), this means that the texts of their items are related with both topics represented by the clusters. This may hint the existence of an implicit relationship between the topics of the two clusters.
  By merging the two overlapping clusters into a single one, the more general topic representing the whole content of the new cluster can be revealed, which subsumes, as most specific topics, those of the original clusters.

[5] This is also consistent with the definition of union of fuzzy sets.

DEFINITION 6: *The* Group join *operator allows the user to obtain, from two or more input groups, a resulting group composed by the union of all those pairs of original clusters that present a non empty intersection.*
In particular, given g_1 and g_2, the input groups, for each pair of clusters $c_1 \in g_1$ and $c_2 \in g_2$, the cluster

$$c' = \mathit{ClusterUnion}(c_1, c_2) \in g',$$

if and only if $\mathit{ClusterIntersection}(c_1, c_2) \neq \emptyset$, with g' the resulting group.

- **Group refinement** The *Group refinement* operator is aimed at refining clusters in a group, based on clusters in another group. While the group join operator generates a cluster representing a more general topic than the topics in both the original clusters, the *refinement* operator can be regarded as generating clusters making more specific the topics of the clusters in the first group on the basis of the topics of any cluster in the second group.
The idea underlying this operator is that we want to collect, in a unique cluster, the items (that are considered by the user as more interesting) which belong to both a cluster c_1 of the first group g_1 and any of the second group g_2. In this way, by eliminating some items from c_1, we generate a cluster representing a more specific topic w.r.t. c_1, but not necessarily more specific w.r.t. the clusters of the second group.

  DEFINITION 7: *The* Group refinement *operator allows the user to keep, from the original group g_1, only the clusters c_i containing documents present in at least one of the clusters c_j of the most interesting group g_2.*
  In particular, given g_1 (group of clusters to refine) and g_2 (interesting group), and c_1 being a cluster such that $c_1 \in g_1$, for each cluster $c_j \in g_2$ we compute the cluster union of the intersections $\overline{c}_j$, i.e.,
  $\overline{c}_j = \mathit{ClusterIntersection}(c_1, c_j)$.
  If the union c' of $\overline{c}_j$ is not empty, then $c' \in g'$.

The operators so far introduced constitute the core of our proposal; the others are sketched hereafter.

- **Group union.** The *Group union* operator unites together two groups. It generates the resulting group g' in such a way it contains all clusters in the input groups g_1 and g_2.
- **Group coalescing.** Complex processing of retrieved documents may need to be performed by fusing all clusters in a group into one global cluster. The **Group coalescing** operator generates a resulting group g' in such a way that g' contains only one cluster, obtained by uniting together all clusters in the input group g.
- **Reclustering.** After complex transformations, it might be necessary to reapply the clustering method to a group. In fact, reclustering documents

in a group may let new and unexpected semantic information emerge. The *Reclustering* operator coalesces all clusters in the input group g and generates a new group g' in such a way that it contains all the clusters obtained by clustering all items.

- Finally, two operators that do not combine groups are available, they are the **Group selection** and the **Group deletion**. The *Group selection* operator permits to select the clusters in a group. Similarly, the *Group deletion* operator allows the user to delete clusters.

The operators are intrinsically based on a soft approach, for several reasons. In fact, clusters can be regarded as fuzzy subsets where the *Irank*s are the membership degrees of the ranked items. Clusters intersection and union are defined as the intersection and union of fuzzy sets. Thus, also the Group operators, relying on the cluster intersection and union, are defined based on fuzzy operators. Finally, the combined ranking properties are defined based on a prioritized fuzzy operator used to aggregate mandatory and optional constraints.

Finally, notice that the *Closure property of Group operators* holds: operators are defined on groups and generate groups (Bordogna et al., 2008a).

An example In order to organize a trip to visit London, let us submit the query `"visit London"` to the search engines *Google*, *Yahoo!* and *MS Bing*. Groups g_1, g_2, g_3 in Fig. 3 are the resulting groups clusters; the three groups being generated by the same query `"Visit London"` have the same label.
Terminated the inspection of clusters in the groups, we can interactively ask for executing some operators, in an attempt of obtaining clusters with labels that more closely meet our needs. At first, we ask to intersect the three groups to retrieve the most reliable documents. Notice that in the resulting group g_4 we have cluster *cl.1* labeled `"Visit London"`, as in groups g_1 and g_2: this means that after the intersection, the document that gave the label to the two clusters in g_1 and g_2 (that is their centroid), is the centroid of cluster *cl.1* in g_4 as well.

By observing all clusters in group g_4, we then decide to request a join of the three original groups g_1 g_2 and g_3, in order to expand the contents obtained by the intersection. A new group g_5 is generated with more populous clusters: these clusters are the union of the original clusters that share some common document. We can see that the obtained clusters are identified by labels, which hint the presence of new correlated contents w.r.t. the labels of the clusters obtained by the intersections of the same groups (see group g_4 vs group g_5 in Fig. 3).

7. Using Matrioshka as an exploratory environment

In this section, we analyze the impact on the user of some exploratory tools made available by Matrioshka to explore the Web. To carry on the analysis, we performed three different tests, hereafter reported.

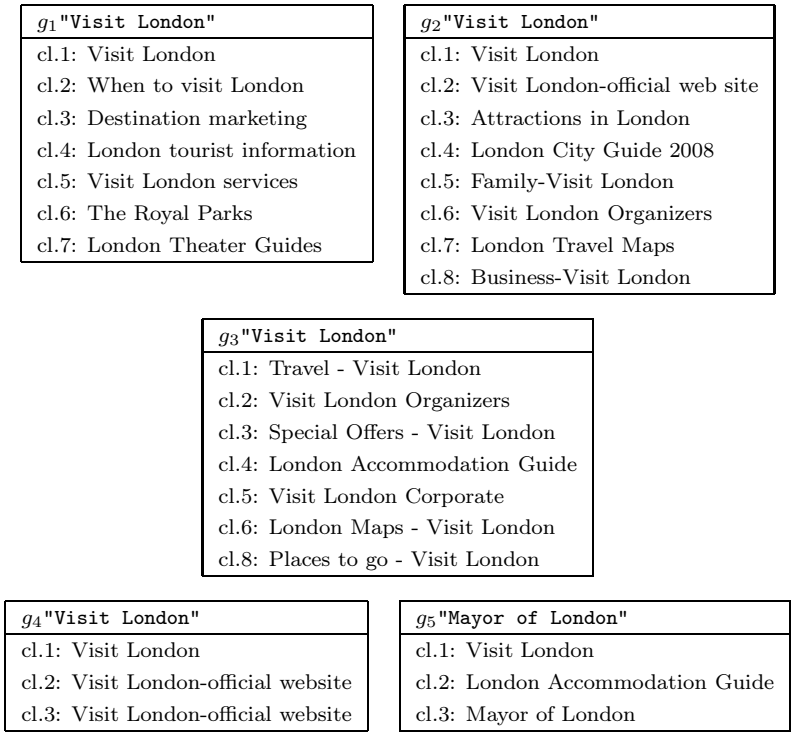

Figure 3. Resulting groups from the query *Visit London* submitted to Google (group g_1), Yahoo! (group g_2), and MS Bing (group g_3); *Group intersection* and *Group join* of groups g_1, g_2 and g_3 yield groups g_4 and g_5, respectively

7.1. Query suggestions

We wanted to analyze the automatic query reformulation feature provided by *Matrioshka* starting from a query and then following a path of suggested queries (see Table 1 and Table 2).

Just by looking at the cluster labels and associated suggested queries obtained by having initiated the Web exploration with the original query "web intelligence", submitted to *Google*, *Yahoo! API*, *Bing* and *Google Scholar*, reported in Table 1 and Table 2, we learned some interesting relations of the query topic with other topics.

- There are courses that give certificates on the theme of web intelligence.
- Web intelligence is strongly related with semantic web approaches.
- Web intelligence has links with brain data mining.
- There is a branch of Web intelligence applying fuzzy ontologies.
- Web intelligence is applied in social networks to identify drug trafficking and terrorists.

Table 1. Matrioshka suggested queries: Depth-First Paths (part 1)

Queries suggested by Matrioshka and resubmitted at each cycle			
Search Engine: *Google* on 10/22/2010 (80 top ranked results) **Original Query**: Web intelligence			
Query 1	**Query 2**	**Query 3**	**Query 4**
Journal emerging technology	Technologies journal emerging Web intelligence	Web learning technologies	Web learning technologies publications semantics
Certificate web intelligence	Certificate training web intelligence	Web Intelligence courses On line ubc	
Web Intelligence international conference	Web intelligence international advances conferences	19th Web intelligence international advances	Conferences publications semantic web lab
Web customizing intelligence	Business web intelligence customizing objects		
Search Engine: *Yahoo! API* on 10/22/2010 (50 top ranked results) **Original Query**: Web intelligence			
Query 1	**Query 2**	**Query 3**	**Query 4**
International conference web intelligence	Electronic edition wi iat intelligence		
Search Engine: *Bing* on 10/22/2010 (780 top ranked results) **Original Query**: Web intelligence			
Query 1	**Query 2**	**Query 3**	**Query 4**
Web intelligence certificate program	Intelligence certificate web program	Certificate geospatial intelligence graduate program	

Table 2. Matrioshka suggested queries: Deep-First Paths (part 2)

Search Engine: *Google Scholar* on 10/22/2010 (50 top ranked results) **Original Query**: Web intelligence			
Query 1	**Query 2**	**Query 3**	**Query 4**
Web intelligence ontologies semantic networks			
Web intelligence knowledge management semantic	Intelligence Web ontology semantic education	Models personalizations users semantic web	Users ontologies personalized semantics learning
Web intelligence brain	Web intelligence computational brain data	Intelligence brain web data mining	Web intelligence Data mining research
		Computational intelligence Web brain techniques	Computational intelligence granular uncertainty → fuzzy systems based personalization granular
Web intelligence sites	Web intelligence business exploration	Semantic web adding business intelligence	Semantic triple space Web computing → space based semantic web computing
Web intelligence fuzzy world knowledge	Semantics fuzzy ontology web knowledge	Fuzzy ontology OWL description logic	Fuzzy ontology description logic reasoner
Web intelligence social networks	Network analysis criminal social applications	Knowledge discovery terrorism criminal networks	
		Drug analysis social network trafficking	

We also noticed that several clusters, in distinct groups, are generated with the same labels.

Table 3 reports the list of homonymous clusters of the search process reported in Table 1 and Table 2; nevertheless their associated disambiguated queries are different since they reflect the paths of the submitted queries that have been followed to generate them. Query suggestions do not just support users in specifying more refined and focused queries, but also serve the purpose of "highlighting" the main retrieved topics in each cluster by focusing on what is of interest to the user; thus they offer a personalized synthesis of clusters.

Table 3. Homonymous clusters with distinct associated disambiguated queries

Cluster label	**Disambiguated Queries for Different Clusters**		
Web service	Semantic web triple space computing	Space based web semantic computing	Semantic web service discovery based
Semantic Web	Web semantic social network	Web Semantic fuzzy ontology knowledge	
Social networks	Web intelligence social networks	Web intelligence social semantic wi	
Sap businessobject web	Sdk sap intelligence businessobject	Sap search business object	Web intelligence sap businessobject
Data mining	Web intelligence integrated data	Web intelligence mining usage based	
Search	Web semantic search wi publish	Web search intelligence finding site	Searching social network web computing
IEEE WIC ACM international conference	International conference web intelligence	IEEE web intelligence conference	

7.2. Use of group operators to filter out relevant results retrieved by most search engines

In the search process described in Section 7.1, we were overloaded with too many results (see Table 1 and Table 2). Consequently, the second test we performed was aimed at *identifying* the most relevant results among those already retrieved. To achieve this goal, we regarded the search engines as experts in finding relevant information: a document that was retrieved as relevant by all the search engines, should have been among the most relevant retrieved documents. So, to filter out the documents retrieved by all the search engines *Google*, *Google Scholar*, *Yahoo! API* and *Bing* as a result of the query “web intelligence”, we

Table 4. Intersection of the groups retrieved by the query "web intelligence" to a search engine (part 1)

Google Scholar, *Yahoo! API*: Empty Group
Google Scholar, *Google*: Empty Group
Google Scholar, *Bing*: Empty Group
Google, *Yahoo! API*: 1. http://www.sap.com/solutions/sapbusinessobjects/ large/business-intelligence/qra/web_intelligence/index.epx 2. http://unex.uci.edu/certificates/it/web_intel/ 3. http://wi-consortium.org/ 4. http://en.wikipedia.org/wiki/Web_intelligence 5. http://www.sap.com/solutions/sapbusinessobjects/ large/business-intelligence/qra/web_intelligence/featuresfunctions/index.epx 6. http://www.arisey.com/web-intelligence-101
Google, *Bing*: 1. http://www.sap.com/solutions/sapbusinessobjects/large/ business-intelligence/qra/web_intelligence/index.epx 2. http://unex.uci.edu/certificates/it/web_intel/ 3. http://wi-consortium.org/ 4. http://en.wikipedia.org/wiki/Web_intelligence 5. http://www.sap.com/solutions/sapbusinessobjects/ large/business-intelligence/qra/web_intelligence/featuresfunctions/index.epx 6. http://www.yorku.ca/wiiat10/ 7. http://www.informatik.uni-trier.de/ ley/db/conf/webi/index.html 8. http://www.csdw.status.dhhs.state.nc.us/userguides /WebIntelligenceUserGuide.pdf 9. http://www.sap-press.com/products/ SAP-BusinessObjects-Web-Intelligence.html

performed their group intersection and, finally, a coalescing in order to have a group with one single cluster (see Table 4 and Table 5). The order of application of the intersection is irrelevant, since the operator is associative (Bordogna et al., 2008b).

We noticed that *Google Scholar* did not retrieve anything in common with the other search engines, while we obtained 5 documents common for the result lists provided by *Google*, *Yahoo! API* and *Bing* (reported in the last row of Table 5).

Further, we noticed that some retrieved groups had similar labels. We decided to analyze if they had some common and correlated contents (see Table 6). We observed that they share very few documents, at most one, while their documents share correlated contents; so, we decided to unite them into a single group.

Table 5. Intersection of the groups retrieved by the query "web intelligence" to a search engine (part 2)

Bing, *Yahoo! API*:	
1.	http://unex.uci.edu/certificates/it/web_intel/
2.	http://www.sap.com/solutions/sapbusinessobjects/large/business-intelligence/qra/web_intelligence/index.epx
3.	http://en.wikipedia.org/wiki/Web_intelligence
4.	http://wi-consortium.org/
5.	http://www.sap.com/solutions/sapbusinessobjects/large/business-intelligence/qra/web_intelligence/featuresfunctions/index.epx
6.	http://www.checkpoint.com/products/web_intelligence/index.html
7.	http://unex.uci.edu/certificates/it/web_intel/courses.asp
8.	http://help.sap.com/businessobject/product_guides/boexir3/en/xi3_web_intelligence_rich_client_en.pdf
9.	http://www.sdn.sap.com/irj/boc/webi
10.	http://www.businessobjects.com/pdf/products/queryanalysis/ds_web_intelligence.pdf
Google, *Bing*, *Yahoo! API:*	
1.	http://www.sap.com/solutions/sapbusinessobjects/large/business-intelligence/qra/web_intelligence/index.epx
2.	http://unex.uci.edu/certificates/it/web_intel/
3.	http://wi-consortium.org/
4.	http://en.wikipedia.org/wiki/Web_intelligence
5.	http://www.sap.com/solutions/sapbusinessobjects/large/business-intelligence/qra/web_intelligence/featuresfunctions/index.epx

7.3. Use of cluster reordering facilities to explore cluster contents

Finally, we wanted to analyze how the reordering facilities of clusters based on their properties could be exploited in an exploratory task.

We analyzed the distinct ranking of the group retrieved by *Google Scholar* as a result of "web intelligence": in Fig. 4, from the left to the right, the reader can see the *content relevance* ranking (that ranks first the results closest to the query), the *ponderosity* ranking (that ranks first the biggest clusters), the *heterogeneity* ranking (that ranks first clusters with most diversified contents) and, finally, on the right, the *combined* content ranking and possibly 80% heterogeneity ranking (that ranks documents based on a balance of their content relevance and heterogeneity, this last property weighted at 80% of importance of the first property). It can be observed that in the first three top ranked positions distinct clusters are present. Specifically, the clusters "intelligence wi" appears in two lists within the first 3 top positions (since it is very relevant to the query and heterogeneous too) and also "fuzzy logic" and "knowledge management" (since they are both very heterogeneous and relevant w.r.t. the query).

Table 6. Common contents of groups with similar labels

1sr Group Label:	Semantic fuzzy ontology web knowledge
	(16 ranked items)
2nd Group Label:	fuzzy ontology OWL description logic
	(18 ranked items)
Group Operation:	Intersection
Retrieved items:	1 ranked item
http://www.springerlink.com/index/d732510437u89537.pdf	
1sr Group Label:	Intelligence web brain data mining
	(48 ranked items)
2nd Group Label:	Computational intelligence brain techniques
	(7 ranked items)
Group Operation:	Intersection
Retrieved items:	1 ranked item
http://www.springerlink.com/index/aq8remqft6099w51.pdf	
1sr Group Label:	Web brain intelligence
	(57 ranked items)
2nd Group Label:	web Intelligence computational brain data
	(28 ranked items)
Group Operation:	Intersection
Retrieved items:	1 ranked item
http://www.springerlink.com/index/aq8remqft6099w51.pdf	
Group Operation:	Join
Retrieved items:	30 ranked item
Group Operation:	Union
Retrieved items:	85 ranked item
1sr Group Label:	Google.social networks
	(41 ranked items)
2nd Group Label:	Yahoo! API.social networks
	(59 ranked items)
Group Operation:	Intersection
Retrieved items:	4 ranked item
Group Operation:	Join
Retrieved items:	31 ranked item
Group Operation:	Union
Retrieved items:	72 ranked item

Figure 4. Group "web intelligence" retrieved by *Google Scholar* (ordered, from left to right, by *content relevance*, *ponderosity*, *heterogeneity*, *content relevance* and possibly 80% *heterogeneity*).

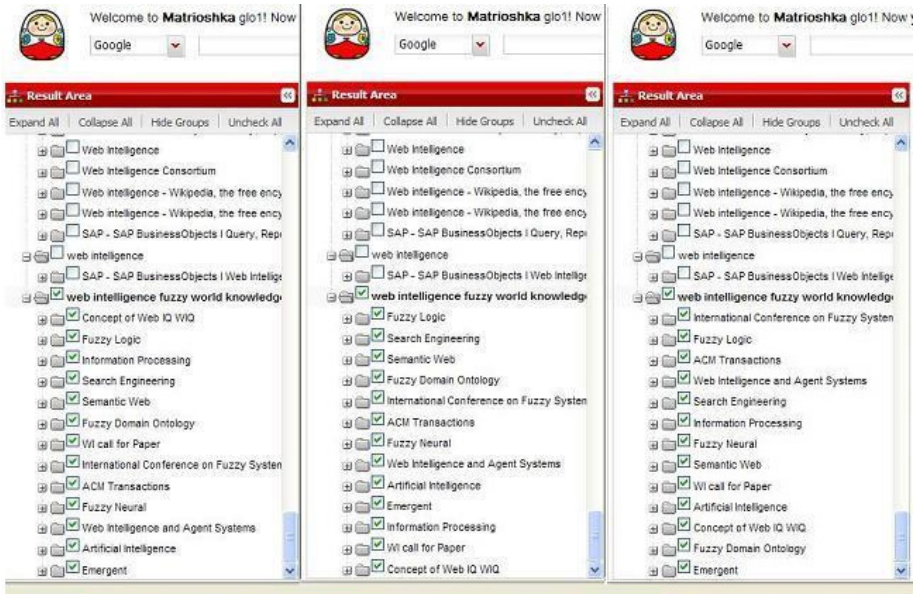

Figure 5. Group "web intelligence fuzzy world knowledge" retrieved by *Google Scholar* (ordered, from left to right, by *novely*, *content relevance*, *novelty* and possibly 100% *ponderosity*)

Then, we submitted the suggested query "web intelligence fuzzy world knowledge" disambiguating the content of cluster "fuzzy logic"; the resulting clusters were ordered first by their *novelty*, then by their *content relevance*, and finally by a *balance* of their *novelty* and possibly 100% *ponderosity*. As it can be seen in Fig. 5, the cluster with most novel documents is labeled "concepts of web IQ WIQ" and, surprisingly, this is also the least relevant to the query, the most relevant cluster to the query is the one labeled "fuzzy logic", while the cluster that is the most novel and biggest one is labeled "international conference on fuzzy systems".

8. Conclusions

In this paper, we presented the application of soft approaches to model web search explorations developed within *Matrioshka*, a prototype system for meta web searches.

Matrioshka implements a novel interaction framework for personalizing exploration of the results of several web searches. The work is motivated by the idea that many queries to search engines produce similar results, and that users can more effectively find the relevant information they need by exploring the already retrieved items. We thus cluster the results of each query in order to have similar documents in the same cluster, and then in *Matrioshka* we provide tools to conduct complex exploratory searches of the clusters contents, in order to discover hidden relevant information.

Tools developed within *Matrioshka* heavily rely on soft approaches, both to compute some cluster properties and to manipulate and analyze clusters contents. Throughout the paper, we presented our soft techniques to evaluate qualities of clusters, for computing preference-based combinations of quality measures, to generate disambiguated queries and to manipulate clusters. The whole process and the intermediate results are stored in the process repository, so as to support long lasting activities.

References

Agrawal, R., Gollapudi, S., Halverson, A. and Ieong, S. (2009) Diversifying search results. In: *Proceedings of ACM WSDM 2009, the Second International Conference on Web Search and Data Mining.* ACM, New York, NY, 5–14.

Balfe, E. and Smyth, B. (2005) An Analysis of Query Similarity in Collaborative Web Search. In: *Advances in Information Retrieval, 27th European Conference on IR Research, ECIR 2005*, Santiago de Compostela, Spain. **LNCS 3408**, Springer Verlag, 330–344.

Bordogna, G., Campi, A., Psaila, G. and Ronchi, S. (2008a) An interaction framework for mobile web search. In: *Proceedings of MoMM-2009,*

the 6th International Conference on Advances in Mobile Computing and Multimedia. Linz, Austria. ACM, 183–191.

Bordogna, G., Campi, A., Psaila, G. and Ronchi, S. (2008b) A language for manipulating clustered web documents results. In: *Proceeding of CIKM 2008, the 17th ACM conference on Information and knowledge management*. Napa Valley, California. ACM, 23–32.

Bordogna, G., Campi, A., Psaila, G. and Ronchi, S. (2009a) A flexible language for exploring clusterd search results. In: A. Laurent and M.J. Lesot, eds., *Scalable Fuzzy Algorithms for Data Management and Analysis*. IGI Global, 179–213.

Bordogna, G., Campi, A., Psaila, G. and Ronchi, S. (2009b) Query Disambiguation Based on Novelty and Similarity Users Feedback. In: *Proceeding of FQAS 2009, the 8th International Conference on Flexible Querying Answering Systems*. Roskilde, Denmark. Springer Verlag, 179–190.

Brin, S. and Page, L. (1998) The anatomy of a large-scale hypertextual Web search engine. In: *WWW-7: Proceedings of the seventh international conference on World Wide Web*. Brisbane, Australia, 107–117.

Carbonell, J. and Goldstein, J. (1998) The use of MMR, diversity-based reranking for reordering documents and producing summaries. In: *Proceedings of ACM SIGIR 1998, the 21st Annual International Conference on Research and Development in Information Retrieval*. Melbourne, Australia. ACM, 335–336.

Card, S.K., Mackinlay, J.D. and Shneiderman, B. (1999) *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers Inc. San Francisco, CA.

Carpineto, C., Osinski, S., Romano, G. and Weiss, D. (2009) A Survey of Web Clustering Engines. *ACM Computing Survey* **41** (3).

Chen, H. and Dumais, S. (2000) Bringing Order to the Web: Automatically Categorizing Search Results. In: *Proceedings of ACM CHI 2000, the International Conference on Human Factors in Computing Systems*. The Hague, The Netherlands. ACM, 145–152.

Chirita, P., Firan, C. and Nejdl, W. (2007) Personalized query expansion for the web. In: *Proceedings of ACM SIGIR 2007, the 30th Annual International Conference on Research and Development in Information Retrieval*. Amsterdam, The Netherlands. ACM, 7–14.

Chung, W., Chen, H. and Nunamaker, J.J. (2003) Business intelligence explorer: a knowledge map framework for discovering business intelligence on the Web. In: *Proceedings of HICSS 2003 the 36th Annual Hawaii International Conference on System Sciences*, vol. 1, p. 10.2, Big Island, Hawaii, USA.

Coates, T., Connolly, D., Dack, D., Daigle, L., Denenberg, R. Durst, P.G., Hawke, S., Iannella, R., Klyne, G., Masinter, L., Mealling, M., Needleman, M. and Walsh, N. (2001) URIs, URLs, and URNs: Clarifications and recommendations 1.0. Technical report, World Wide

Web Consortium, URI Planning Interest Group W3C/IETF, http://www.w3.org/TR/2001/NOTE-uri-clarification-20010921/.

Cutting, D., Karger, D., Pedersen, J. and Tukey, J. (1992) Scatter/ Gather: a cluster-based approach to browsing large document collections. In: *Proceedings of SIGIR 1992, the 15th Annual International Conference on Research and Development in Information Retrieval*. Copenhagen, Denmark. ACM, 318–329.

Dubois, D. and Prade, H. (1985) A Review of Fuzzy Set Aggregation Connectives. *Information Sciences* **36** (1-2), 85–121.

Fellbaum, C., ed. (1998) *WordNet An Electronic Lexical Database*. The MIT Press, Cambridge, MA, USA.

Freyne, J., Smyth, B., Coyle, M., Balfe, E. and Briggs, P. (2004) Further Experiments on Collaborative Ranking in Community-Based Web Search. *Artificial Intelligence Reviews* **21** (3-4), 229–252.

Gollapudi, S. and Sharma, A. (2009) An axiomatic approach for result diversification. In: *Proceedings of WWW 2009, the 18th World Wide Web Conference*. Madrid, Spain, 381–390.

Hearst, M.A. and Pederson, J.O. (1996) Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In: *Proceedings of ACM SIGIR 1996, the Annual International Conference on Research and Development in Information Retrieval*. Zurich, Switzerland. ACM.

Jansen, B., Booth, D. and Spink, A. (2009) Patterns of query reformulation during Web searching. *JASIST Journal of the American Society for Information Science and Technology* **60** (7), 1358–1371.

Jansen, B. and Spink, A. (2006) How are we searching the World Wide Web? A comparison of nine search engine transaction logs. *Information Processing and Management* **42** (7), 248–263.

Jansen, B., Spink, A. and Saracevic, T. (2000) Real life, real users and real needs: A study and analysis of users queries on the Web. *Information Processing and Management* **36** (2), 207–227.

Kampanya, N., Shen, R., Kim, S., North, C. and Fox, E.A. (2004) Citiviz: A Visual User Interface to the CITIDEL System. In: *Proceedings of ECDL 2004, the 8th European Conference on Research and Advanced Technology for Digital Libraries*. **LNCS 3232**, Bath, UK. Springer Verlag, 122–133.

Lad, A. and Yang, Y. (2007) Generalizing from Relevance Feedback using Named Entity Wildcards. In: *Proceedings of ACM CIKM 2007, the Sixteenth Conference on Information and Knowledge Management*. Lisbon, Portugal. ACM, 721–730.

Lawrence, S. (2000) Context in Web Search. *IEEE Data Engineering Bulletin* **23** (3), 25–32.

Leouski, A. and Croft, W. (1996) An Evaluation of Techniques for Clustering Search Results. Technical Report of the Department of Computer Science of University of Massachusetts at Amherst. IR-76, 122–133.

Liu, F., Yu, C. and Meng, W. (2004) Personalized Web Search For Improving Retrieval Effectiveness. *IEEE Transactions on Knowledge and Data Engineering* **16** (1), 28–40.

Liu, Y., Jin, R. and Chai, J. (2006) A statistical framework for query translation disambiguation. *ACM Transactions on Asian Language Information Processing (TALIP)* **5** (4), 360–387.

Luca, E.D. and Nurnberger, A. (2005) A Meta Search Engine for User Adaptive Information Retrieval Interfaces for Desktop and Mobile Devices. In: *Proceedings of PIA 2005, the 1st International Workshop on New Technologies for Personalized Information Access, in conjunction with UM05, the 10th International Conference on User Modeling.* Edinburgh, UK, 23–34.

Ma, Z., Pant, G. and Sheng, O. (2007) Interest-based personalized search. *ACM Transactions on Information Systems* **25** (1).

Mihalkova, L. and Mooney, R. (2009) Learning to Disambiguate Search Queries from Short Sessions. In: *Proceedings of ECML-PKDD 2009, the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases: Part II.* Bled, Slovenia. Springer Verlag, 111–127.

Notess, G., ed. (2006) *Teaching Web Search.* Information Today Inc., New Jersey, USA.

Osinski, S. (2003) An Algorithm for Clustering of Web Search Results. Master thesis, Department of Computing Science, Poznań University of Technology, http://project.carrot2.org/publications/osinski-2003-lingo.pdf.

Patwardhan, S., Banerjee, S. and Pedersen, T. (2005) SenseRelate:: TargetWord: a generalized framework for word sense disambiguation. In: *Proceedings of ACL 2005, the International Conference of the Association for Computational Linguistic, Interactive Poster and Demonstration Sessions.* Ann Arbor, Michigan. Association for Computational Linguistics, 73–76.

Radlinski, F., Bennett, P.N., Carterette, B. and Joachims, T. (2009) Redundancy, diversity and interdependent document relevance. *SIGIR Forum* **43** (2), 46–52.

Roussinov, D. and Chen, H. (2001) Information navigation on the web by clustering and summarizing query results. *Information Processing and Management* **37** (6), 789–816.

Sameh, A. and Kadray, A. (2010) Semantic Web Search Results Clustering Using Lingo and WordNet. *International Journal of Research and Reviews in Computer Science* **1** (2), 71–76.

Sanderson, M. (2008) Ambiguous queries: test collections need more sense. In: *Proceedings of ACM SIGIR 2008, the 31st Annual International Conference on Research and Development in Information Retrieval.* Singapore. ACM, Singapore, 499–506.

Sebrechts, M.M., Vasilakis, J., Miller, M.S. and J.V. Cugini, S.J.L. (1999) Visualization of Search Results: A Comparative Evaluation of Text, 2D, and 3D Interfaces". In: *Proceedings of ACM SIGIR 1999, the 22nd Annual International Conference on Research and Development in Information Retrieval*. Berkeley, CA, USA. ACM, 3–10.

Shen, X., Tan, B. and Zhai, C. (2005) Implicit user modeling for personalized search. In: *Proceedings of ACM CIKM 2005, the 14th International Conference on Information and Knowledge Management*. Bremen, Germany. ACM, 824–831.

Staley, E. and Twidale, M. (2000) Graphical interfaces to support information search. Technical report, University of Illinois, http://people.lis.uiuc.edu/ twidale/irinterfaces/bib-main.html.

Sugiyama, K., Hatano, K. and Yoshikawa, M. (2004) Adaptive web search based on user profile constructed without any effort from users. In: *Proceedings of WWW 2004, the 13th International Conference on World Wide Web*. New York, NY, USA. ACM, 675–684.

Sun, J., Zeng, H., Liu, H., Lu, Y. and Chen, Z. (2005) CubeSVD: a novel approach to personalized Web search. In: *Proceedings of WWW 2005, the 14th International Conference on World Wide Web*. Chiba, Japan. ACM, 382–390.

Sweeney, S., Crestani, F. and Losada, D.E. (2008) Show me more': Incremental length summarisation using novelty detection. *Information Processing and Management* **44** (2), 663–686.

Teevan, J., Dumais, S. and Liebling, D. (2008) To personalize or not to personalize: modeling queries with variation in user intent. In: *Proceedings of ACM SIGIR 2008, the 31st Annual International Conference on Research and Development in Information Retrieval*, Singapore. Singapore. ACM, 163–170.

Voorhees, E. (1993) Using WordNet to disambiguate word senses for text retrieval. In: *Proceedings of ACM SIGIR 1993, the 16th Annual International Conference on Research and Development in Information Retrieval*. Pittsburgh, PA, United States. ACM, 171–180.

Xu, Y. and Chen, Z. (2006) Relevance judgment: What do information users consider beyond topicality? *JASIST Journal of the American Society for Information Science and Technology* **57** (7), 961–973.

Zadeh, L. (1965) Fuzzy Sets. *Information and Control* **8**, 338–353.

Zamir, O. and Etzioni, O. (1999) Grouper: a dynamic clustering interface to Web search results. *Computer Networks* **31** (11-16), 1361–1374.

Zhai, C., Cohen, W. and Lafferty, J. (2003) Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In: *Proceedings of ACM SIGIR 2003, the 26th Annual International Conference on Research and Development in Information Retrieval*. Toronto, Canada. ACM, 10–17.

Zhang, B., Li, H., Liu, Y., Ji, L., Xi, W., Fan, W., Chen, Z. and Ma, W. (2005) Improving web search results using affinity graph. In: *Proceedings of ACM SIGIR 2005, the 28th Annual International Conference on Research and Development in Information Retrieval*. Salvador, Brazil. ACM, 504–511.

Zhang, Y. (2008) Complex adaptive filtering user profile using graphical models. *Information Processing and Management* **44** (6), 1886–1900.