A Cluster Manipulation Paradigm for Mobile Web Search Interaction

Gloria Bordogna  Alessandro Campi  Stefania Ronchi  Giuseppe Psaila
CNR-IDPA  Politecnico di Milano  Università di Bergamo
Via Pasubio 5  Piazza L. da Vinci 32  Viale Marconi 5
24044 Dalmine (BG)  20133 Milano  24044 Dalmine (BG)
Italy  Italy  Italy
gloria.bordogna@idpa.cnr.it  campi@elet.polimi.it  ronchi@polimi.it
psaila@unibg.it

ABSTRACT
This paper describes a new interaction paradigm well suited to perform web searches though a mobile device. The prototypical system that implements this novel interaction framework is named Matrioshka, that is a multi-modal system. In this paper we focus on the interaction framework and will introduce briefly an overview of the mobile version of Matrioshka. This framework is based on cluster manipulation operations. The results of a user request, yielded by one or more search engines, are organized into labelled clusters. Then, some manipulation operators can be applied to re-rank clusters or to combine them to generate new clusters. These facilities allow the user to capture the relevant documents hidden in the large set of retrieved ones in the first ranked clusters.

Keywords
Web searches, mobile information retrieval, results clustering, ranking strategies

1. INTRODUCTION
The large diffusion of Internet connections from anywhere at anytime has arisen the problem of more effective ways of searching the Web from mobile devices. In this paper, a mobile interaction framework for web meta-searching is proposed, whose definition is motivated by the observation that the visualization method based on the ranked list of web pages is too long to fit small screens such as those of mobile devices. Further, with the aid of a mobile keyboard, the usual way of interacting with search engines based on repeated cycles of query reformulation imposes too much burden to the user. At the same time, it is too expensive in terms of the high cost of mobile connections. In fact, if users do not find what they are looking for in the first one or two result pages, they are more keen to reformulate a new query than to analyze successive pages, or to submit the current query to another search engine.

To overcome these drawbacks, some services such as vivisimo, clusty, Snaket, Ask.com (at [1]), MS AdCenter Labs Search Result Clustering, etc., proposed to cluster the results of Web searches. W.r.t. the ranked list, clustered results are more compact and offer an overview of the main topics dealt with in much more documents than those contained in the first few pages, that would be missed otherwise [8, 16, 11]. As far as we know, in the literature we found only one academic mobile search engine, named Credino [2] that exploits clustered results.

On the other side, one problem users encounter with such clustered results, is the inability of fully understanding the contents of the clusters. This is mainly due to the short and sometimes bad quality of the clusters’ labels, which generally consist of a few terms, or individual short phrases, which are automatically extracted from the documents within clusters. Often, several clusters have similar labels, which differ just for a single term. To effectively explore the cluster contents, users have no other means than clicking on the cluster labels and browsing the clusters themselves. On a mobile device, this modality would again require too much scrolling.

The idea of our proposal is to maintain the result clustering paradigm, and to provide users with a language to manipulate clusters. Both several ranking criteria to differently order the clusters, and operators to combine the clusters themselves are defined whose final aim is to make possible the exploration of the retrieved contents.

The literature on mobile search engines mainly focuses on modelling the user context, considering primarily the user geographic location, in order to filter the retrieved results [10]; other topics are the summarization of documents [7], and the definition and use of data visualization schemes [13]. In [6] the clustering of retrieved results is proposed as a useful way of presenting the search results on small screens, but, to the best of our knowledge, only the mobile search engine Credino [2] performs clustering. The manipulation language as a basis for a flexible interaction makes our proposal substantially different from Credino [2], where the focus is the clustering algorithm it adopts w.r.t. other clustering methods, and does not offer criteria to explore the cluster contents.

A motivation of utility of the manipulation language can be found in [12] which advocates the need of tools for giving the user more immediate control over the clusters of retrieved web documents. Our proposal can be particularly useful when groups of clusters with same or almost same labels are generated by distinct requests or by the same query submitted to distinct search engines. In such situations it
becomes necessary to explore the contents of the clusters and their relationships in terms of number of contained documents, relevance of contents, homogeneity of contents, or common and distinct contents with other clusters. This task an exploratory task, that may last for a long time, and may require to reuse the intermediate results several times. For this reason, storing of the intermediate results into a database is essential for successive manipulation. Furthermore, the local manipulation of results avoids the useless overloading of both the network and the search engines. In fact, in current practices, several modified queries are submitted to the search engines, trying to capture relevant documents in the first positions of the ranked list; note that most of these documents were already retrieved by the previous queries, although hidden to the user since they did not occur in the first positions.

In [4] and [5], we proposed and defined the operators for combining the clusters for revealing their implicit relationships. In [3] a prototypal mobile meta search system was proposed that allows easily using the combination operators. In this paper we propose an extension of the manipulation language by introducing a ranking operator that makes possible the exploration of the cluster contents based on distinct properties of the clusters.

2. THE INTERACTION FRAMEWORK

Data Model. Here we describe the data model on which the proposed interaction framework is based. We start considering a query q submitted to a search engine; its result is a ranked list of documents, that we call items.

Definition 1: Item An item i represents (an instance of) a document retrieved by a web search. It is described by the following attributes: uri, which is the Uniform Resource Identifier of the ranked web document; title and snippet which are, respectively, the document title and snippet\(^1\); finally, irank is a score (in the range \([0, 1]\)) that expresses the estimated relevance of the retrieved document w.r.t. the query.

The same document (web page) may be represented by distinct items in distinct result lists. In facts, we assume that a document is uniquely identified by its uri [9], while it may have distinct snippets, irank and title, when retrieved by different search services (or by different queries). We assume that irank is a function of the position of the item in the query result list.

In our system, the results of a user request (or exploration) are not simply a ranked list of documents, but they are gathered in ordered clusters.

Definition 2: Cluster A cluster c is a set of items, having a rank. It is defined by two attributes: label is a set of terms that semantically synthesizes the main content of the cluster; crank is a score (in the range \([0, 1]\)) depending on some property of the cluster.

A cluster label is automatically generated by a specific labelling algorithm on the basis of frequent terms in cluster items [3].

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 following attributes:

\(^1\)The snippet is an excerpt of the document, made by a set of sentences that may contain the keywords of the query label, a set of terms that semantically synthesizes the main content of the group; s, the name of the search engines used to retrieve the items in the clusters of the group. □

Finally we define the users’ History repository.

Definition 4: History A history H is a set of items. It can be the empty set, at the beginning of a search session, and it can be updated by explicit action of the user when he/she decides to save a retrieved document. □

Manipulating Clusters. The procedure that generates a group is initially activated by a search operator, named CQuery, that allows users to query a search engine (e.g., Google, Yahoo!, MSN Search) and to cluster the results. In the implementation we considered a maximum of N documents, with \(n \geq 30\), i.e., a number of documents greater than that retrieved in the first three pages, those usually analyzed by a common user.

On this basis, for each retrieved document, the operator builds an item i, whose irank value depends on the position of the document in the result list: \(i\.\text{irank} = (N - \text{Pos}(d) + 1)/N\) (where Pos(d) is the position of the document in the query result list). In this way, a document in the first positions has a rank \(i\.\text{irank}\) very close to 1. This is done in order to achieve independence and comparability of the ranking produced by distinct search engines.

The ranked list obtained as a result by the search operator, is then clustered by applying the Lingo algorithm [14]. Lingo is used to perform a flat crisp clustering of the query results on the basis of their snippets and titles. Once clusters are obtained, they are labelled. Finally also the groups are labelled (see [3] for the labelling algorithm) to synthesize the most central contents retrieved by all their clusters.

Successively, one can decide either to explore the groups of clusters retrieved by a single query by applying some ranking operation described in 2.1 which evaluates a cluster property, or one can generate other groups by combining the obtained ones through the operators defined in Section 2.2.

2.1 Cluster Ranking Methods

Once the results of a query are obtained as a group of ranked clusters, in which the default crank score is computed as the average of the irank of its documents, the user has the possibility to re-rank the clusters based on the evaluation of some other clusters’ property. This allows to obtain, in the first positions of the ranked list of clusters, those clusters that previously could appear in the last positions. This is the novel contribution of the paper w.r.t. our previous work: the user is this way provided with the possibility of evaluating groups by different perspectives.

The cluster properties that can be considered for the ranking are the following:

- Relevance: this is defined as the average of the relevance scores of documents belonging to the cluster and is the default property for the ordering of clusters; the relevance scores of clusters are the irank values computed as previously defined from the documents’ positions in the ranked list returned by the search engine. Ordering clusters by decreasing values of their relevance means being interested primarily in the relevance of documents contained in the clusters.
- Ponderosity: this is defined as the cluster cardinality, and it measures how many documents belong to the clusters; the ranking of clusters in decreasing order of their ponderosity can be useful for users interested in high recall.
- **Heterogeneity**: this is defined as the variance of the documents vectors, 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, defined as the average vectors of all the documents vectors belonging to the cluster. The greater the variance the more heterogeneous is the cluster: by choosing to rank clusters in increasing order of their heterogeneity means being interested in contents focalized 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, by choosing to rank clusters in decreasing order of their heterogeneity means being more tolerant on the meaning expressed by the cluster label; this can be useful when one is unsure to have expressed by the query the actual information needs and wants to soften the selection conditions.

- **Novelty**: this is defined as the proportion of novel documents contained in the cluster w.r.t. previously already seen documents, that the user has saved in the history repository; choosing a novelty ranking means being interested in new documents on the topics of a search and can be useful in the context of bibliographic surveys.

In order to rank clusters of a group based on one of the above properties the operation ClusterRank is defined:

\[ g' = \text{ClusterRank}(g, \text{property}, \text{order}) \]

in which \( g \) and \( g' \) are the input and output groups of clusters, \text{property} takes values in a set of strings \{ \text{Relevance}, \text{Pondosity}, \text{Etherogeneity}, \text{Novelty} \} denoting a cluster property; \text{order} \in \{ \text{increasing}, \text{decreasing} \} indicates the desired ordering, i.e., increasing and decreasing w.r.t. the value of the specified cluster property, respectively.

\( g' \) has the same label of \( g \) and contains the same clusters of \( g \) with the only difference that the clusters’ crank scores are computed based on the specified \text{property} of the clusters:

\[ \text{crank}_i = \frac{\text{ranks}(\text{property}(c_i))}{\text{MAX}(\text{ranks}(\text{property}(c_i)))} \]

### 2.2 Combining Groups of Clusters

The system 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 [4]: they are inspired by the operators provided by the Relational Algebra (i.e. intersection, join, union etc.), thought 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 ore more clusters; it can also be empty, in the case no common items are detected.

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 \( \text{uri} \) of the items of two input clusters, assuming that \( \text{uri} \) is the document’s unique identifier. The rationale of this assumption is the fact that the same document, retrieved by two different search services, may have different title and snippet, but maintains the same \( \text{uri} \). Consider the intersection of two clusters \( c_1 \) and \( c_2 \), denoted as:

\[ c' = \text{ClusterIntersection}(c_1, c_2). \]

The \( \text{irank} \) of \( i' \in c' \), the cluster resulting from the intersection, is defined as the minimum \( \text{irank} \) value of \( i_1 \) and \( i_2 \).\(^2\) In the case of cluster union, denoted as \[ c' = \text{ClusterUnion}(c_1, c_2), \]
the \( \text{irank} \) of \( i' \) is the maximum \( \text{irank} \) value of \( i_1 \) and \( i_2 \).\(^3\) In both cluster intersection and union, the title and the snippet of the resulting items are obtained by selecting either \( i_1 \text{title} \) or \( i_2 \text{title} \), and either \( i_1 \text{snippet} \) or \( i_2 \text{snippet} \), respectively.

In particular, to obtain the title and the snippet of the items belonging to the clusters of the resulting groups we select as resultant \text{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 \( \text{irank} \), 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.

#### 2.2.1 Group Operators

The first group operators we describe are not properly combination operators: they are the **Group Selection** and the **Group Deletion**. The **Group Selection** operator allows to select the clusters in a group. In the resulting group, the selected clusters maintain the original order. Similarly, the **Group Deletion** operator allows the user to delete clusters. Like for the Cluster Selection operator, the original order is maintained in the resulting group.

The following operators combine and generate groups.

- **Group Intersection**. Group Intersection is defined to support the straightforward wish of users to intersect clusters in two groups, to find more specific clusters. The assumption is that the more search services (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 not empty intersection. In particular, given \( g_1 \) and \( g_2 \) the groups of cluster 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 more specific topics, those of the original clusters.

\(^2\)This definition is consistent with the definition of the intersection operation between fuzzy sets [15].

\(^3\)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 not 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
\[
\begin{align*}
\text{c' } & = \text{ClusterUnion}(c_1,c_2) \\
\text{if and only if } & \text{ClusterIntersection}(c_1,c_2) \neq \emptyset,
\end{align*}
\]
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 specializing 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 from the second group that are regarded as generating clusters specializing the topics of the original clusters.

Definition 7. The Group Refinement operator allows the user to keep, from the original group \( g_1 \), only the clusters \( c_i \) containing documents presents 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 being \( c_1 \) 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 \( c_j \cap c_1 = \text{ClusterIntersection}(c_1,c_i) \).

If the union \( c_i' \) of \( c_i \) is not empty, then \( c_i' \in g' \). □

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.

The Closure Property of Group Operators holds: operators are defined on groups and generate groups [5].

3. THE MOBILE SYSTEM MATRIOSHKHA

The interaction framework introduced in the previous section has been implemented in the mobile version of the prototypical system Matrioshka.

It is constituted by three main parts: the client side components handle the user interaction; the server side component interfaces the search engines and executes the clustering and the manipulation operations specified by the user; finally, the Communication Layer dispatches the messages between client and server. Specifically, the client provides a query editor for the user, the server either executes the queries and builds the groups of clusters or executes the operations on previously generated groups of clusters. Let us describe the functionality of each architectural component.

On the client side the Matrioshka User Interface collects users requests, displays the results of queries and/or the application of manipulation operations. The Client-side components are thin clients compliant, and communicate with the server-side by exchanging XML messages. Specifically, the component for mobile devices (called Mobile Matrioshka), is a Javascript application based on the AJAX (Asynchronous JavaScript and XML) web development technique.

The Server Side exposes a web service interface, based on XML messages: it receives requests to perform queries on search services, or to apply the operators; it replies with groups of clusters. All the data received from the search engines, and those resulting from the operations, are stored in an XML native database; this way, the entire process is stored and can be accessed to carry on the exploratory task.

The server side is entirely implemented in the Java Language. The interaction with search services usually exploits web service APIs provided by the search engines, otherwise the standard HTTP interaction model is exploited.

Document clustering is performed on the indexes extracted from the titles and snippets of retrieved documents (generated by using Lucene functions): the Lingo multilingual algorithm, provided by the Carrot2 libraries is used.

The interpreter of the combination operators has been implemented from scratch.

The Communication Layer is a pool of JSP scripts, executed on top of the Tomcat web server. It carries out the client/server communication through XML format messages, according with AJAX web development techniques, and by the support of the Tomcat Java servlet container.

When the user logs into the system, a specific instance of the database is created, in which the entire exploratory process performed by the user will be stored. When logged-in, the user has the possibility to submit queries to the chosen search engine (as shown in the left-hand side of Figure 1).

In order to organize a trip to visit London, let us submit the query "visit London" to the search engines Google, Yahoo! and MSN search. Groups \( g_1 \), \( g_2 \), \( g_3 \) in Figure 2 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. By observing clusters in the resulting group $g_3$, 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 (see the screen shots in Figure 1). A new group $g_4$ 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 groups $g_1$ vs group $g_3$ in Figure 2).

4. CONCLUSIONS

In this paper, we described a novel interaction framework for web searches implemented by the prototypal mobile version of the system Matrioshka. The features that make this framework particularly suitable for mobile searches are several: first, it presents clustered results of the searches so as to better render them on the small screen of mobile devices; it makes available ranking and combination operators defined for clusters manipulation which allow easily exploring the retrieved results, thus alleviating network overloading caused by the submission of repeated refined queries to search engines. The large number of documents retrieved by such engines constitute a serious obstacle for users of mobile devices, who generally engages long trial and error query reformulation phases to retrieve relevant results in first few positions. The operator provided by the interaction framework are the basis for complex exploratory tasks; users can issue operations through the mobile interface, but certainly they must be skilled users; certainly, generic users are in troubles. Currently we are performing an evaluation study to understand the effectiveness for end users, in order to define novel, more user friendly interaction paradigms on the client side, more suitable for generic users.

5. REFERENCES