STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVIII**, Number 4, 2013

IMAGE SEARCH RESULTS DIVERSIFICATION

CLAUDIU EPURE, ALEXANDRA-MIHAELA SIRIŢEANU, AND ADRIAN IFTENE

Abstract. In the last years, the volume of data (text, image or multimedia) has grown exponentially. In this context, the information retrieval task has become more and more difficult. This paper aims to present an application that performs image retrieval based on Flickr network image collection. After the user fills in the query, the system is responsible with (1) query processing, (2) image retrieval and (3) results displaying. Before retrieving the results, the system clusters the images based on several criterions and shows to the user one representative image for every cluster it creates. In this way, the results displayed to the user are diverse and he still has the possibility to view all the images from a cluster, when clicking on the representative image. The system achieves diversity in two ways: (i) at query processing, when the user adds from WordNet the lexical family corresponding to query keywords, and (ii) when the results are displayed and the clusters are created.

1. Introduction

The need of humans to socialize and share information has led to a constantly growing Web, which has become a support for social media. Every day, worldwide users are pushing multimedia data towards their family, friends and the world at large. This is the reason why, web search has also become the main method for people to fulfill their information needs. While web consists in huge amounts of multimedia documents, searching for relevant information is not that straightforward, since user queries are often ambiguous and have more than one interpretation. For instance, the word Turkey is mostly used to refer to the country, but it can also refer to the bird from the Meleagrididae family. Trying to overcome this issue, the search engine could either rank its results based on the "best guess" probability or try maximizing the probability of a user in finding at least one relevant document on the results page. In

Received by the editors: November 14, 2013.

2010 *Mathematics Subject Classification.* 68T50, 68P20, 91F20.

1998 *CR Categories and Descriptors.* H.3.1. [**Information Systems**]: Content Analysis and Indexing – *Linguistic processing*; J.5. [**Computer Applications**]: Arts and Humanities – *Linguistics*.

Key words and phrases. Image Retrieval, Diversification.

this context, diversity appears to be a trade-off between having more relevant results of the most likely intent and having diverse results that cover different query interpretations.

The common modality used for image search on the web is based on text, the assumption being that the tags and the textual descriptions associated with photos are powerful ways to describe and retrieve images. In this way, the better the textual metadata captures the content of the image, the better the system performance is. Yet, "a picture is worth a thousand words" [1], simply because complex ideas can convey in a single image. This explains the efforts made in content-based image retrieval (CBIR), moving from simple low-level features to high-level semantics.

Over time, various theories involving search results diversification have been developed, theories that have been taken into consideration [4]: (i) *content* (or similarity) [7], i.e. how different are the results to each other, (ii) *novelty* [2], [3], i.e. what does the new result offer in addition to the previous ones, and (iii) *semantic coverage* [19], i.e. how well covered are the different interpretations of the user query.

The rest of the paper is structured as follows. In Section 2, we survey the various ways of diversifying search results, in Section 3 we present existing data clustering algorithms and we review the algorithm used by our system, in Section 4 we describe our system, an image search engine based on Flickr data, which diversifies the results, using both images meta-data and images content, in Section 5 we show how our system works based on two case studies and finally, in Section 6 we present our conclusions.

2. Related work

Generally, the problem of diversifying search results can be seen as a compromise between finding relevant elements to the query and achieving variety in the result set. Given a set $X = \{x_1, ..., x_n\}$ of n items, a query q, and an integer k, $k \leq n$, we want to find $Y \subseteq X$ of size $|Y| = k$, such that every element from Y is relevant to q with respect to a similarity function $\delta_{sim} : q \times X \rightarrow \Re^+$ and, at the same time, diverse among other elements in Y with respect to the diverse function $\delta_{div} : X \times X \rightarrow \Re^+$.

In this section, we describe various dimensions considered for diversification in the multimedia retrieval literature [4] and we structure our presentation based on how diversity is defined: (i) content, (ii) novelty and (iii) semantic coverage.

[1] Napoleon Bonaparte

2.1. **Content-based definitions.** Content-based definitions describe diversity as an instance of the max-sum dispersion problem, where given a set of n elements in some space, the aim is to select k points, such that the sum of distances between pairs of selected points is maximized [13]. The diversity problem consists in selecting a subset $Y \subseteq X$ of size $|Y| = k$ that maximizes the sum of inter-element distances among items of Y. The k-diverse set Y is formally defined in [17] as:

$$\begin{aligned} Y = & \underset{X' \subseteq X, |X'| = k}{argmax} \; div(X'), where \\ div(X') = & \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \delta_{div}(x_i, x_j), x_i, x_j \in X'. \end{aligned} \tag{1}$$

Content-based diversity is commonly used along with user's query relevance criterion. The relevance problem is another instance of the max-sum dispersion problem and aims to find a subset $Y \subseteq X$ of size $|Y| = k$ with the largest sum of similarity distances among all sets of size k in X. The k-similar set Y is formally defined in [17] as:

$$\begin{aligned} Y = & \underset{X' \subseteq X, |X'| = k}{argmax} \; sim(q, X'), where \\ sim(q, X') = & \sum_{i=1}^{k} \delta_{sim}(q, x_i), x_i \in X'. \end{aligned} \tag{2}$$

The result diversification problem is further defined in [17] as computing a set $Y \subseteq X$ of size $|Y| = k$ with a trade-off λ, $0 \leq \lambda \leq 1$ between having k elements that are both similar to the query q (1) and also diverse to each other (2). The k-similar diversification is formally defined as:

$$\begin{aligned} Y = & \underset{X' \subseteq X, |X'| = k}{argmax} \; F(q, X'), where \\ F(q, X') = & (k-1)(1-\lambda) \cdot sim(q, X') + 2\lambda \cdot div(X') \end{aligned} \tag{3}$$

Intuitively, $F(q, Y)$ measures the amount of "attractive forces", between q and k elements in Y, and "repulsive forces", among elements in Y.

2.2. **Novelty-based definitions.** Novelty and dissimilarity are different though related notions. The novelty quality of a document refers to how different it is with respect to the other documents previously seen, while dissimilarity applies to a set of items, and is related to how different the items are with

respect to each other. This is related to novelty in that when a set achieves dissimilarity, each document contains a piece of novel information with respect to the rest of the set.

In [3], a document is defined as a collection of information nuggets from the space of all possible nuggets. A nugget is interpreted broadly as a binary property, which may represent a fact, an answer to a question, a topicality or other similar piece of information. A document d is considered to be relevant if it contains any relevant information, which means at least one nugget that is also contained in the user's query q. Given a results list retrieved by an IR system for query q, the probability that the k^{th} document is both novel and diverse from the $k-1$ appearing before it can be expressed as the probability of that document containing a nugget that cannot be found in the previous $k-1$ documents.

2.3. **Coverage-based definitions.** Queries submitted to a retrieval system are often ambiguous. For a user to find at least one relevant result, it is desirable to diversify search results so that top-ranked documents can cover different query interpretations. In [19], the general idea of modeling diversity is to retrieve k documents that cover many interpretations of query q, especially interpretations that are considered important. Intuitively, the more subtopics Y covers and the more important the covered subtopics are, the higher diversity score Y has. Taking into consideration the relevance criterion, the k-similar diversification set Y contains k elements from X that maximize $F(q, Y)$:

$$F(q,Y) = \lambda \sum_{x_i \epsilon X} \delta_{sim}(q, x_i) + (1-\lambda) \sum_{s_i \epsilon S(q)} \delta_{weight}(q, s_i) \sum_{x_j \epsilon Y} \delta_{cov}(x_j, s_i) \tag{4}$$

where $\delta_{weight}(q, s_i)$ defines the importance of the suptopic s_i and $\delta_{cov}(x_j, s_i)$ measures the coverage of the subtopic s_i in the document $x_j \epsilon$ Y.

3. Data clustering

3.1. **Existing algorithms.** In this section we formally describe the clustering problem and we present several clustering techniques considered in the literature. In [18], clustering is defined as the process of identifying a finite set of categories, classes or groups in the dataset such that data objects in the same cluster should be similar to each other, while data object in different clusters should be dissimilar from one another. Clustering techniques are generally classified as partitional clustering and hierarchical clustering, based on the properties of the generated clusters [6], [10]. Han and Kamber [9] suggested categorizing the methods into additional three main categories: density-based methods, model-based clustering and grid-based methods.

Partitional clustering algorithms assign a set of data points into non-overlapping subsets without any hierarchical structure, such that each group contains at least one point, each data object is in exactly one subset and the union of all classes form the dataset [16].

Hierarchical clustering algorithms organize data into a hierarchical structure according to the proximity matrix, the result being a cluster hierarchy, also known as dendrogram. Hierarchical clustering methods are categorized into agglomerative (bottom-up) and divisive (top-down) [12]. The merging or division of clusters is performed according to some similarity measure, which divides the hierarchical clustering methods into: single link, average link, and complete link.

Density-based clustering is presented in [18] as a technique capable of finding arbitrarily shaped clusters, where clusters are defined as dense regions separated by low-density regions. A density based algorithm is often applied when clusters are irregular or intertwined, and when noise and outliers are present. Specifically, the algorithm DBSCAN (Density Based Spatial Clustering of Applications with Noise) [5] implements the concept of **density-reachability** and **density-connectivity** to define clusters. Letting $N_\epsilon(p) = \{q \in P | \delta_{dist}(p,q) \leq \epsilon\}$ be the ϵ-neighborhood of a data point p (where ϵ is a given radius and δ_{dist} a distance function), a point p is considered to be **density-reachable** from a point q if there is a chain of points $p_1 ... p_n, p_1 = q$ and $p_n = p$ such that for each p_{i+1}: (1) $p_{i+1} \in N_\epsilon(p_i)$, and (2) $|N_\epsilon(p_i)| \geq min$, where min is a prespecified parameter that specifies the minimum number of data points that must form the neighborhood of a data point in a cluster. If two points are found on the border of a cluster, they are possibly not density reachable from each other and their relation can be described using density-connectivity. A point p is considered to be **density-connected** to a point q, if there is a point r such that p and q are both density-reachable from r.

In **model-based clustering** [11], it is assumed that the data are generated by a mixture model of several probability distributions, each with its own set of parameters. Rather than performing classical clustering, the goal is typically to estimate the parameters of each model and compute the cluster assignment. The estimation is usually done using an iterative scheme similar to those used by partitioning methods. The main assumption is that each point is associated with a hidden class and belongs to only one cluster.

As density-based methods, **grid-based clustering** are popular for mining classes in large multidimensional space, classes being perceived as denser regions than their surroundings. A typical grid-based clustering algorithm consists in dividing a data space into a set of cells, calculating the cell density for each cell, sorting the cells according to their densities, identifying the cluster centers and traversing the neighbor cells [8].

3.2. **Proposed algorithm.** We propose a density based algorithm inspired by DBSCAN [5], from which we take the ideas of *distance* and *dynamically creation of clusters*. The minimum number of images for the creation of a cluster is set to one, reason why, after clustering, isolated images (noise) cease to exist. Another difference concerns the input of the search, where the first image from the request is taken and not a random one, like in the case of DBSCAN.

In this approach, the notion of *distance* changes according to the type of request specified by the user: based on the *text* or on the *content*. In the first case, we use the Levenshtein algorithm [15] to calculate the distance between two strings, applied on the image's annotations (title, description, tags). In the second case, we compute the distance between images at the pixel level, by applying the algorithms provided by the AForge.NET[2] framework.

3.3. **Application of the algorithm on annotations.** When the user chooses to cluster the images based on text, an analysis of the image associated annotations (title, description, tags) is carried out. For the algorithm to be efficient (to properly group the similar images), the difference between the lengths of the images titles must not be very long[3]. This thing requires a preprocessing of the input data aiming to classify the images based on the title length. In this way, several thresholds are introduced regarding the length of an image title.

The process begins by classifying the images in collections intuitively called *albums*, in such a way that every album has a *threshold* associated and the title of any image from this album comply with these conditions:

- it does not exceed the threshold value;
- it is bigger than the threshold value of the previous album or it is bigger than 0 if the album is the first one.

The values for the chosen thresholds are 20, 40 and 60. In the end, every album consists in a list of clusters and the final result is obtained by concatenating these lists. The algorithm is applied in in the same manner for every album, as follows:

(1) A similarity parameter (*sim*) is set; for example, half of the threshold's length corresponding to the album;
(2) The first image is chosen as representative of the first cluster.
(3) For each remaining image *img*, the following steps occur:

[2]AForge.NET: http://www.aforgenet.com/framework/

[3]for example, let's assume that for the input "car" will be three images: two similar (1), (2) having the titles: car_expo... (45 characters) and car_expo... (10 characters); a different image (3): red_car (7 characters); the algorithm will group the images (2) and (3) which is incorrect

(a) The list of the existing clusters is iterated and the minimum of the Levenshtein distances between the title of the image *img* and the title of the representative image of the cluster is calculated, only if that Levenshtein distance is smaller than the similarity parameter *sim*. The cluster C in which the minimum has been found is remembered.
(b) If there is a minimum, the image is added to the cluster C where the minimum has been found. Otherwise, a new cluster is created having the image *img* as the representative image. The new cluster is added to the clusters list.

In other words, every image is grouped along with the images which closely resembles (in the context of distance between titles) when it satisfies the similarity condition (*sim*) or it becomes the core of a new cluster when it does not satisfy the similarity condition.

3.4. **Application of the algorithm on content.** If the user chooses to cluster the images based on their content, this is carried out by applying an algorithm that analysis the images' pixels. To accomplish this, the *open-source* library *AForge.NET* has been used, which provides an algorithm for computing the similarity between two images. The similarity is represented by a real number between 0 and 1 (the similarity of an image with herself is 1) and is being used as the *distance* parameter. After performing some tests on the test data we have decided to set the *sim* parameter to the value of 0.66.

Having the *distance* and the *sim* parameters specified, the clustering algorithm continues in the same manner as the one presented above. The algorithm used by the library for analyzing the pixels measures the distance between two pixels, namely the difference between the component values (red, green, blue). For example, the distance between the colors *green* RGB (0, 255, 0) and *blue* RGB(0, 0, 255) is $|0-0| + |255-0| + |0-255| = 0 + 255 + 255 = 510$; with respect to RGB(255, 255, 255), the result is $510/765 = 0.66$. The similarity between pixels is further computed as: $1 - 0.66 = 0.33$. Obviously, the similarity between *white* RGB(255, 255, 255) and *black* RGB(0, 0, 0) is $1 - (255 + 255 + 255)/765 = 1 - 1 = 0$. The average of the all similarities between pixels results in the similarity of the compared images.

3.5. **The retrieval of images based on a lexical database.** There are situations when the user does not precisely know or does not remember the term to use for his interrogation, but he has some information about the term like: a synonym, an antonym or its word family. For solving this issue, a lexical database can be used. A lexical database is a database that contains

information about words and about the relations that exist between these words and other words in their word family.

According to *WordNet* (WordNet, 2012) between the words of a lexic there are relations that can help at the creation of hierarchies and different properties can be highlited. The main relation between words is the *synonymy*. In *WordNet*, the synonyms are grouped into sets of synonyms (*synsets*). Between these sets there are many types of relations, the most important being those of hypernyms and hyponyms. Other types of relations are: meronymy, holonymy and for adjectives, anyonymy.

Example: Let's assume that the user wants to get information about *double-decker* and he doesn't know exactly the term but he knows that it is in the *bus* category. Searching for *bus* using the option of *hyponym*, he will be able to access the desired word. With the *meronym* option, he could obtain words like *wheel, engine, air-bag*.

4. System architecture

An *Image Retrieval* application must be capable of fast query processing, conveniently storing results in a database and provide the user a good experience in terms of human-computer interaction. In the case of results diversification, applications deal with large datasets, thus they have to implement very efficient algorithms in which the comparison is the predominant operation. Even so, for best results, they rely on powerful hardware processing units.

The proposed solution is a web application which is simultaneously accessible for plenty of users. The application uses the ubiquity of public web services in order to obtain images, metadata and ontology datasets. The necessity of using a *Cloud Computing*[4] solution, has led us to choosing the *Windows Azure* [1] platform, which takes care of data storage and application hosting.

Our system is a *Client-Server* application and, from an architectural point of view, it follows the *Model-View-Controller* pattern. The main functionalities are: (1) Querying and getting data, (2) Diversification and saving images and (3) Displaying the results. The main components are shown in Figure 1:

- the **Server**: contains both the classes responsible for getting the images and the main algorithms of classification;
- the **Database**: consists of a *SQL* server for storing users and management data, and a *NoSQL* server for storing images metadata based on the previously imposed diversification type;

[4]*Cloud Computing* the practice of storing regularly used computer data on multiple servers that can be accessed through the Internet: http://www.merriam-webster.com/dictionary/cloud%20computing

- the **Client**: contains the graphic user interface consisting of several web pages.

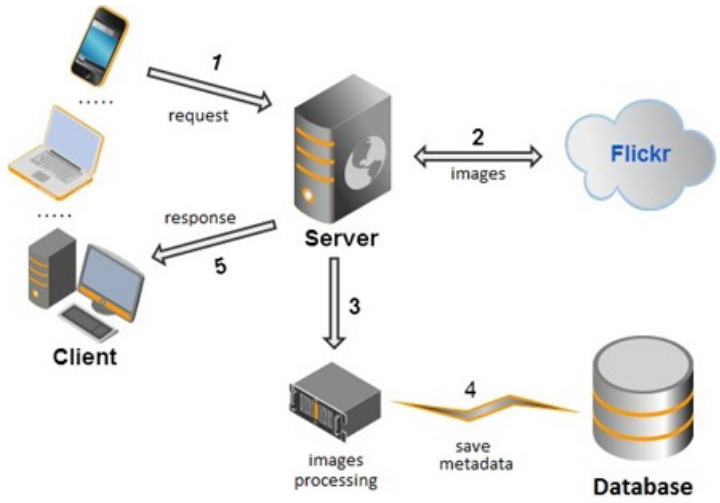

FIGURE 1. *System Architecture*

In the following paragraphs we will present each component's structure and functionality along with the technologies that we have used in our application.

4.1. **Server.** Following the Client-Server model, when a client sends a request (synchronous or asynchronous), the server executes an action and sends a response back to the client. When the query (a text with keywords) is received, the server updates the current user's query list and searches for results firstly in the database. When such results occur, they are sent to the client without other processing. If this is not the case, the query is run against a web service (e.g. Flickr) and the corresponding images metadata are received. Based on the query, the images are further processed according to the diversification algorithms at text level (title, description, annotations) or at content level (pixels similarity). The metadata is clustered and the obtained classes are stored in the database. Finally the images result list is displayed to the client.

The application was developed using Microsoft technologies and the solution contains four projects:

- two *Class Library* projects for storing the models and for implementing the access to the database;
- one *ASP.NET MVC 4 Web Application* project for the application itself;

- one *Windows Azure Cloud Service* project that encapsulates the application.

We have implemented the server using the .NET framework and we have imported in our project several libraries as follows:

- *Flickr.NET*[5] for easily accessing the Flickr API in .NET;
- *WordNet*[6] API for the English lexical database;
- *Antelope*[7], a framework for easily accessing the WordNet API in .NET;
- *AForge.NET* API for processing the images at pixels level.

4.2. **Database.** The data required by the application are stored in two different databases, providing specific functionalities:

- a relational database server administrated by Microsoft SQLServer for storing and handling of registered users and their data (queries, feedback, etc.);
- an non-relational database (NoSQL) administrated by the Windows Azure Storage for storing images meta-data, classified in tables of queries and clusters (Table Storage), but also for storing images (Blob Storage).

The **relational database** consists of the following tables: *UserProfile* (where the users are stored), *Keywords* (where the users' queries are stored), *KeywordUserProfiles* (where the users' different types of queries are specified), structured as it can be seen in Figure 2.

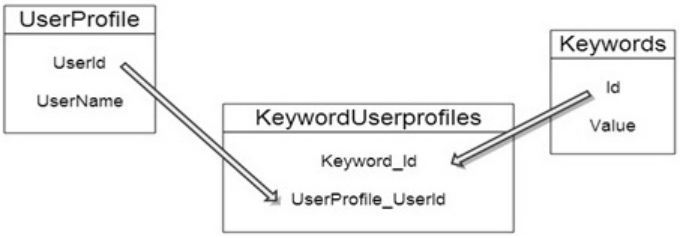

FIGURE 2. *SQL tables structure and relations*

The primary key id of the Keywords table is built based on the plain-text query, by applying a .NET predefined hash function and then concatenating the resulted string with the label defining the type of the diversification algorithm applied (text-based or content-based). This is necessary in order to

[5]Flickr.NET: http://flickrnet.codeplex.com/
[6]WordNet: http://wordnet.princeton.edu
[7]Antelope: https://www.proxem.com/en/technology/antelope-framework/

obtain a unique ID for each pair (query type + keyword) in part. This *id* is used afterwards to name a NoSQL table stored in TableStorage, which contains all the images metadata grouped in clusters, based on the type and the value of the query.

The **non-relational database** provides two types of storage:

- Table Storage - stores images metadata resulted after the clustering process in a convenient way for the following queries, which will contain the keyword and the clustering type (text-based or content-based);
- Blob Storage stores the images required for the content clustering process, so that the images would be downloaded a single time.

The *Storage* table can contain any number of tables named according to *Keywords* table id and has the following flexible[8] scheme: *PartitionKey* (cluster id), *RowKey* (image id), *Timestamp, Similarity, Title, UrlSmall, UrlLarge.*

The role of the database is to store both users and images information. The data obtained after the clustering process will be stored in tables from the *Table Storage* unit and the images used in the content-based clustering will be stored in the Images container from the *Blob Storage* unit.

For the relational database, the Microsoft SQL Server technology has been used. With the help of the Entity Framework, the entire database structure has been created and specific methods have been used for the general data manipulation. By using the "code first" technique, the database model has been created first: classes represented the tables and the relations among them. By activating the *Enable Migrations* option, it allowed us an automatic update of the database structure whenever the model was modified and, in addition, it offered the possibility of defining a Seed method, which populated specific tables with default data every time the database structure was updated.

Regarding the non-relational database, the solutions offered by the Microsoft Windows Azure Storage - Table Storage and Blob Storage - have been chosen. These solutions are usually used when storing specific data that do not require a fixed schema, being for this reason suitable for web applications. Another reason taken into consideration was the need to store large amounts of data (images) and the Blob Storage has been found appropriate to store this kind of data.

In order to be used, the application does not require the user authentication. Nevertheless, logging into an account will let the system learn from the user previously queries and offer suggestions for the future ones.

[8]The non-relation databases are characterized by not using a fixed scheme to create a table. Thus, the table can contain different types of entries (both in termes of number and type), being a heterogeneous structure of data.

4.3. **Client.** The client is the part of the application that interacts with the user and consists in an ASP.NET-MVC4 website. Its interface provides the possibility to register, to log in and to configure the search process (see Figure 3). When a user registers and loges in, his configurations and the queries he previously run against the system are used to build in a user profile. In this way, the search process efficiency increases by obtaining the desired results, which are classified using the stored preferences.

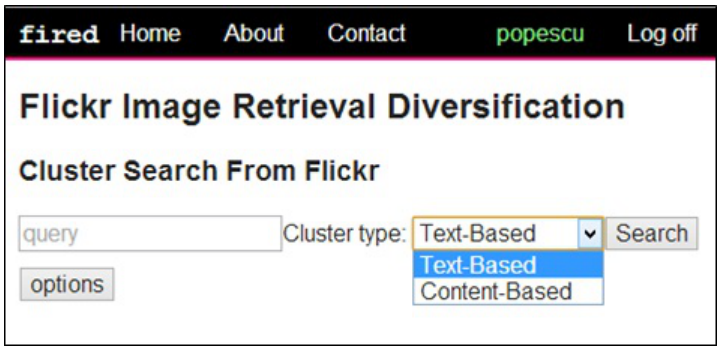

FIGURE 3. *Search elements*

The graphical interface is made up of a series of web pages, through which the user can interact with the application. The application provides *responsive design*, a modern concept, which provides different perspectives of the graphic interface depending on the screen size of the device used to access the application [14]. This aspect is achieved by using CSS media queries feature.

Communication with the server is done using two request-response methods. The navigation, the registration and the authentication services make use of the *PostBack* method. For sending the queries and receiving the results, the asynchronous PostBack method is used, known by the name AJAX[9]. The advantages of using this method are: low data traffic by avoiding reloading the page, better interactivity and easy navigation. There is however a drawback: the *Refresh* and the *Back* buttons cannot be used to refresh the results or to go back to a previous search.

By filling in and submitting a simple form, consisting in a query input and several options for the search process, the user receives a set of images grouped in several clusters. By clicking an image, an asynchronous request is made to the server and the images belonging to the same cluster are retrieved and displayed. The image associated metadata (title, description, tags, URL)

[9]AJAX - Asynchronous JavaScript and XML

obtained from the Flickr social network can be viewed by accessing directly the image.

In case of a search based on lexical analysis, the server response consists in a list of related words, which by direct access, regular request (like the one presented in the previous paragraph) with the selected word in the query are made to the server.

For creating the graphic interface, several web technologies have been used, both classic (HTML 5, CSS 3, JavaScript) and specific: ASP.NET-MVC 4 (layout pages, shared sections, Razor sintax, html helpers). The skeleton is built in HTML 5, using *Boilerplate*. For the design part, the CSS3, Initializr[10], jQueryUI[11] and Bootstrap have been used. The functionality is provided by pure JavaScript, but also by using JavaScript libraries such as jQuery[12], Modernizr and Bootstrap.

5. Case studies

5.1. **Search based on simple text.** A user wants to query the server in order to obtain relevant images from the collection provided by the Flickr social network. The query is *text-based* and the result will contain only images having in their metadata (title, description, tags) keywords matching the query. After receiving the images from Flickr, the clustering process begins. The clustering technique is chosen based on the option selected by the user (Figure 3). It can be an algorithm that clusters the images based on their annotations or their content.

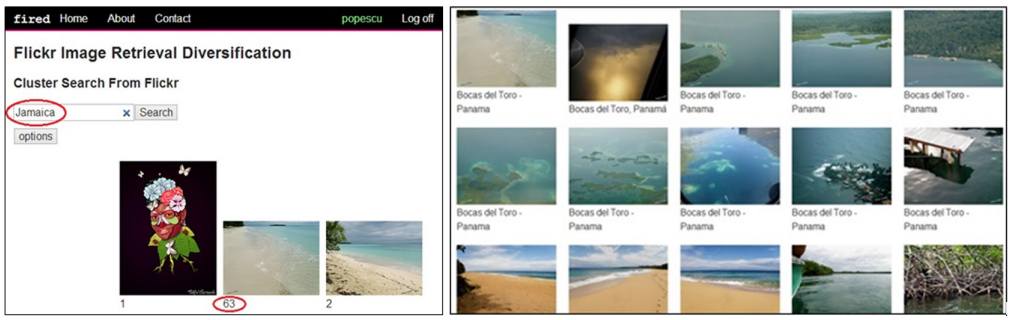

Figure 4. For keyword "Jamaica": a) *Image clusters (in the left)* b) *The images from a cluster (in the right)*

[10]Initializr: http://www.initializr.com/
[11]jQueryUI: http://jqueryui.com/
[12]jQuery: http://jquery.com

The results, grouped in clusters, are retrieved to the client. Every cluster is defined by a single image on the results page and can be further visualized by clicking on the represented image (see Figure 4 a)). Every image in the result set is characterized by several general features such as title, description, URL, that can be viewed when the image is directly accessed (Figure 4 b)).

5.2. **Search based on WordNet.** In this case the user wants to perform a search when he does not know or he does not remember the specific keyword that characterizes the images he is searching for. For this situation he can perform a search using word family of the keyword. In order to do that, the user can select one of the relations available in the interface: synonym, antonym, meronym, holonym, hypernym and hyponym (see Figure 5).

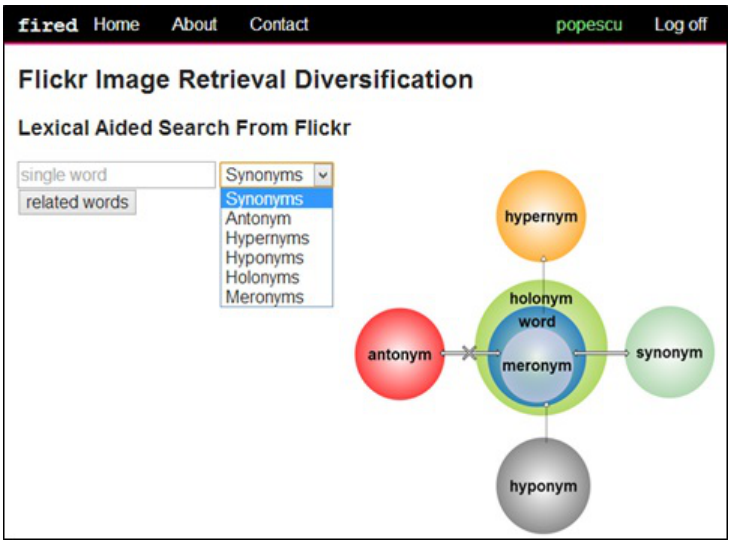

FIGURE 5. *Available options for lexical family of a keyword*

In an initial step, the lexical family is extracted from WordNet accordingly with the selected option. In the next step the user selects the set of words that will be used in the searching process. After that, the search process is similar with the above case (see Figure 6).

6. CONCLUSIONS

In this paper we propose a new clustering algorithm used in the information retrieval task. The aim of this algorithm is to achieve diversification in the results set. The most significant contribution comes from the clustering

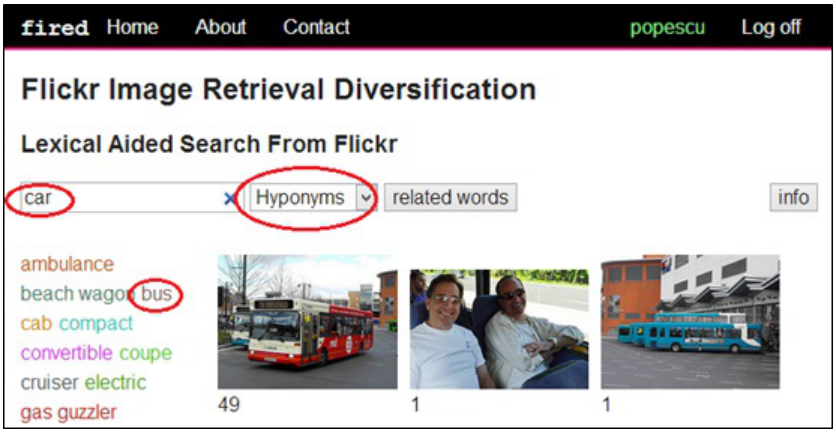

FIGURE 6. *Lexical family of word "car"*

method which is based on *DBSCAN* algorithm. The main differences with respect to this algorithm are elimination of the restriction of having to specify the minimum number of elements to form a cluster and selection of elements in an unelected way. Another contribution can be seen at the annotation level, where images are grouped in an initial phase in albums (depending on length of title) and in a second phase images from the same album are grouped using the *Levenshtein* distance.

We further combine the results obtained in the searching process with lexical analysis operations in cases where the number of results is insufficient. In this way, the user chances to obtain relevant results are increased and the results are more diverse in the end.

For future work, we intend to extend our application and add new functionalities. First of all, we want to use algorithms able to analyze the user query at the semantic level. Secondly, we want to extend our application to work with other languages using specific resources or using translation services such as Google Translate. Another future direction is building web services, which will make our main functionalities available for other applications to use.

ACKNOWLEDGMENT

The research presented in this paper was funded by the project MUCKE (Multimedia and User Credibility Knowledge Extraction), number 2, CHIST-ERA/01.10.2012.

References

1. R. Boucher, *Windows azure*, (2012).
2. J. G. Carbonell and J. Goldstein, *The use of mmr, diversity-based reranking for reordering documents and producing summaries*, SIGIR (1998), 335–336.
3. C. L. A. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Bttcher, and I. MacKinnon, *Novelty and diversity in information retrieval evaluation*, SIGIR (2008), 659–666.
4. M. Drosou and A. Pitoura, *Search result diversification*, SIGMOD (2010), 41–47.
5. M. Ester, P. Kriegel, H., J. Sander, and X. Xu, *A density-based algorithm for discovering clusters in large spatial databases with noise*, Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (1996), 226–231.
6. B. Everitt, S. Landau, and M. Leese, *Cluster analysis, 4th edition*, New York: Oxford University Press (2001).
7. S. Gollapudi and A. Sharma, *An axiomatic approach for result diversification*, WWW (2009), 381–390.
8. P. Grabusts and A. Borisov, *Using grid-clustering methods in data classification*, Proceedings of the International Conference on Parallel Computing in Electrical Engineering (2002), 425–426.
9. J. Han and M. Kamber, *Data mining: Concepts and techniques*, Morgan Kaufmann Publishers (2001).
10. P. Hansen and B. Jaumard, *Cluster analysis and mathematical programming*, Mathematical Programming, series B **79** (1997), 191–215.
11. R. Harpaz, *Model-based linear manifold clustering*, ProQuest (2008).
12. A. Jain and R. Dubes, *Algorithms for clustering data*, Prentice-Hall, Inc. (1988).
13. M. J. Kuby, *Programming models for facility dispersion: The p-dispersion and maxisum dispersion problems*, Geographical Analysis (1987), 315–329.
14. K. Leetham, *Designing responsively*, (2012).
15. V. I. Levenshtein, *Binary codes capable of correcting deletions, insertions and reversals*, Soviet Physics Doklady (1966), 707–710.
16. N. Tan P., M. Steinbach, and V. Kumar, *Introduction to data mining*, Pearson Addison Wesley (2006).
17. M. Vieira, M. Razente, H.and Barioni, M. Hadjieleftheriou, D. Srivastava, C. Traina, and V. Tsotras, *On query results diversification*, Proceedings of the ICDE (2011), 1163–1174.
18. R. Xu and D. Wunsch, *Clustering*, John Wiley & Sons, Inc. (2009).
19. W. Zheng, X. Wang, H. Fang, and H. Cheng, *Coverage-based search result diversification*, Journal Information Retrieval (2012), 433–457.

"Al. I. Cuza" University of Iasi, Faculty of Computer Science, Romania
E-mail address: {claudiu.epure,alexandra.siriteanu,adiftene}@info.uaic.ro