Quick and Easy Implementation of Approximate Similarity Search with Lucene*

Giuseppe Amato, Paolo Bolettieri, Claudio Gennaro, and Fausto Rabitti

ISTI - CNR, Pisa, Italy
{giuseppe.amato,paolo.bolettieri,
claudio.gennaro,fausto.rabitti}@isti.cnr.it

Abstract. Similarity search technique has been proved to be an effective way for retrieving multimedia content. However, as the amount of available multimedia data increases, the cost of developing from scratch a robust and scalable system with content-based image retrieval facilities is quite prohibitive.

In this paper, we propose to exploit an approach that allows us to convert low level features into a textual form. In this way, we are able to easily set up a retrieval system on top of the Lucene search engine library that combines full-text search with approximate similarity search capabilities.

1 Introduction

Very often multimedia content is not associated with any text or metadata, therefore traditional search techniques cannot be used and content-based retrieval or similarity-based retrieval is the only way to access this information. Moreover, even when textual information is available, the combination of similarity search with the full-text search is very useful.

However, if the digital data we want to search for similarity are just a few thousand, a sequential search could be enough. But when the amount of data becomes large (hundreds of thousands), a single similarity search can last minutes.

On the other hand, the continuous price reduction of digital production tools, such as cameras, camcorders, and smartphones, is driving the demand for content-based retrieval tools.

Several attempts are currently being made to provide these capabilities, for instance Google images allows the user to upload a photo to find out similar images in the web. However, the cost of developing and deploying from scratch a robust and reliable system with content-based image retrieval facilities could not be within the range of possibilities for everyone.

But how easy is it to add these features to an existing Digital Library Management System? In this paper, we would like to approach the problem of similarity

* This work was partially supported by the ASSETS project funded by the European Commission.

M. Agosti et al. (Eds.): IRCDL 2012, CCIS 354, pp. 163–171, 2013.

search by enhancing the full-text retrieval library Lucene[1] with content-based image retrieval facilities. Apache Lucene is a high-performance, full-featured text search engine library written entirely in Java that is suitable for nearly any application requiring full-text search abilities.

In particular, we use a technique for approximate similarity search when data are represented in generic metric spaces. The metric space approach to similarity search requires the similarity between objects of a database to be measured by means of a distance (dissimilarity) function, which satisfies the metric postulates: positivity, symmetry, identity, and triangle inequality. The advantage of the metric space approach to the data searching is its "extensibility", allowing us to potentially work for a large number of existing proximity measures as well as many others to be defined in the future. In contrast, many approaches need objects to be represented as vectors and cannot be applied to generic metric spaces.

The basic idea exploited in our approach has been independently introduced by Amato et al [1] and Chavez et al. [4] and consists on observing that two objects x_1 and x_2 are very similar (which in metric spaces means that they are close one to each other), if their view of the surrounding world (their perspective) is similar as well. This implies that, if we take a set of objects from the database and we order them according to their similarity to x_1 and x_2, the obtained orderings are also similar. Therefore, we can approximatively judge the similarity between any two arbitrary objects x_1 and x_2, by comparing the ordering, according to their similarity to x_1 and x_2, of a group of reference objects, instead of using the actual distance function between the two objects.

Clearly, it is possible to find some special examples where very similar (or even identical) orderings correspond to very dissimilar objects. For instance, if reference points are all positioned on a line, two objects that are positioned on another line orthogonal to the first one will produce the same ordering of the reference points, independently of their actual position. However, as it has been proved in [1], even with a random selection of the reference points, the accuracy of this approach is very good.

Capitalizing on the work of Amato et al [1], we also use the inverted files in our research. Another similar approach, called PP-Index [5,6], uses a compact prefix tree for estimating the real distance order of the indexed objects with respect to a query. All these above mentioned approaches make use of index methods completely designed and developed from scratch. Although the results of these systems are quite impressive[2], they probably will not easily move from research prototypes to commercial applications due to the strong effort required to maintain and support such information systems. Consider, for example, Lucene: at the time of this writing, Lucene's core team includes about half a dozen active developers. In addition to the official project developers, Lucene has a fairly large and active technical user community that frequently contributes patches, bug fixes, and new features.

[1] `http://lucene.apache.org`
[2] `http://mipai.esuli.it/`
`http://mi-file.isti.cnr.it/CophirSearch/`

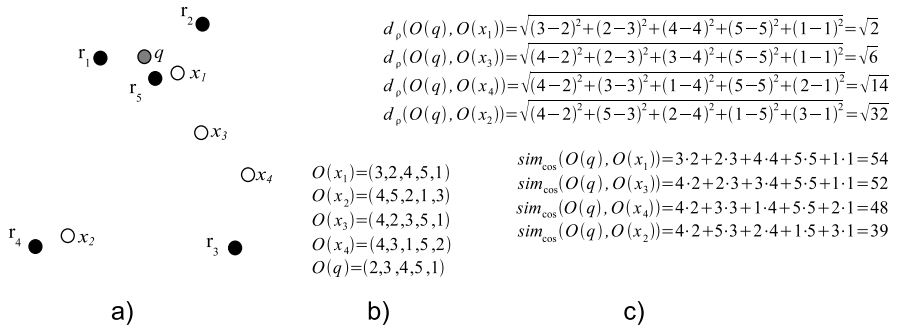

Fig. 1. Example of perspective based space transformation. a) Black points are reference objects; white points are data objects; the gray point is a query. b) Encoding of the data objects in the transformed space. c) Distance d_ρ and similarity s in the transformed space.

Moreover, only the approach in [5] provides a full-text search on descriptive textual metadata, which is, however, not combined with the content-based similarity search. Our approach instead since it is built on top of Lucene provides complex query processing by combining similarity search with the full-text search.

The structure of the paper is as follows. Section 2 formalizes the idea of searching by using the perspective of the objects and shows how this idea can be efficiently supported by the use of the Lucene library. Section 3 proposes a preliminary performance evaluation of the proposed solution.

2 Perspective Based Space Transformation

Let $\mathcal{D}$ be a domain of objects and $d : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ be a metric distance function between objects of $\mathcal{D}$. Let $R \in \mathcal{D}^m$, be a vector of m reference objects chosen from $\mathcal{D}$.

Given an object $x \in \mathcal{D}$, we represent it as the ordering of the reference objects R according to their distance d from x. More formally, an object $x \in \mathcal{D}$ is represented with $O(x)$, where $O(x)$ is the vector of ranks of all objects of R, ordered according to their distance d from x.

We denote the rank in $O(x)$ of a reference object $r_i \in R$ as $O_i(x)$. For example, if $O_4(x) = 3, r_4$ is the 3rd nearest object to x among those in R.

Figure 1 exemplifies the transformation process. Figure 1a) sketches a number of reference objects (black points), data objects (white points), and a query object (gray point). Figure 1b) shows the encoding of the data objects in the transformed space. We will use this as a running example throughout the remainder of the paper.

As we anticipated before, we assume that if two objects are very close one to each other, they have a similar view of the space. This means that also the orderings of the reference objects according to their distance from the two objects should be similar. There are several standard methods for comparing two ordered lists, such as as *Kendall's tau*, the *Spearman Footrule Distance*, and the *Spearman Rho Distance* [7]. In this paper, we concentrate our attention on the latter distance, which is also used in [4]. The reason of this choice (explained later on) is tied to the way standard search engines process the similarity between documents and query. Given two ordered lists $O(x)$ and $O(q)$ (x, $q \in \mathcal{D}$), containing the ranks of all objects of R, the Spearman Rho Distance d_ρ between $O(x)$ and $O(q)$ is computed as in the following:

$$d_\rho(O(x), O(q)) = \sqrt{\sum_{i=1}^{m} (O_i(x) - O_i(q))^2} \tag{1}$$

where m is the dimension of the vector R. This distance measures the degree in which rankings correspond with each other and it can be used in place of the metric distance d (see Figure 1c)).

In order to reduce the search cost and also, as we will see, the size of the index, it is convenient to take just the closest reference objects to represent any object that has to be indexed. Let $k_x \leq m$ be the number of reference objects used for representing the objects. Note that, in this case, different objects will be typically represented by different reference objects, given that different objects will have different neighbor reference objects. This idea can be extended also to the query, for which we can exploit a number $k_q \leq k_x$ of nearest reference objects. If we define two approximate version of the vectors $\widetilde{O}^k$, such that $\widetilde{O}^k_i(x) = k + 1$ for all i such that $O_i(x) > k$ (with either $k = k_x$ or $k = k_q$), we can still use the distance in Eq. (1), i.e:

$$d_\rho(\widetilde{O}^{k_x}(x), \widetilde{O}^{k_q}(q)) = \sqrt{\sum_{i=1}^{m} (\widetilde{O}^{k_x}_i(x) - \widetilde{O}^{k_q}_i(q))^2}. \tag{2}$$

In this case, we assume that x belongs to the dataset and q is the query. This is a generalization of the Spearman Rho Distance with location parameter for the special case $l = k_x = k_q$ [7], which evaluates the distance (or dissimilarity) of two top-k ranked lists.

Up to now, we have discussed how to compare two partial rankings of reference objects corresponding to objects and query. However, we did not say how to implement the proposed idea into a standard full-text search engine.

Most text search engine, including Lucene, use the Vector Space model to represent text. In this representation, a text document is represented as a vector of terms each associated with the number of occurrences of the term in the document. Therefore, we have to define a textual representation each metric object of the database so that the inverted index produced by Lucene looks like the one presented above and that its built-in similarity function behaves like the

Spearman Similarity rank correlation used to compare ordered lists. This can be achieved in several ways, in the following we outline our solution.

First, we associate each element $r_i \in R$ with a unique alphanumeric keyword τ_i. Then we use the function $t^k(x)$, defined in the following, to obtain a space-separated concatenation of zero or more repetitions of τ_i words:

$$t^k(x) = \bigcup_{i=1}^{i} \bigcup_{j=1}^{k+1-\widetilde{O}_i^k(x)} \tau_j$$

where, by abuse of notation, we denote the space-separated concatenation of words with the union operator $\bigcup$. The function $t^k(x)$ returns a text representation of x such that, if r_i appears in position p in the list of the k reference objects nearest to x, then the term τ_i is repeated $(k+1)-p$ times in the text. The function $t^k(x)$ is used to generate the textual representation of the object x to be used for both indexing and querying purposes. Specifically, we use $k = k_x$ for indexing and $k = k_q$ for querying.

In our case, this means that, if for instance term τ_i corresponding to the reference descriptor r_i $(1 \leq i \leq m)$ appears n times, the i-th element of the vector will contain the number n, and whenever τ_i does not appear it will contain 0. To summarize, we finally get the vectors of size m, $\widetilde{O}^{k_x}(x)$ and $\widetilde{O}^{k_q}(q)$, which correspond to $t^{k_x}(x)$ and $t^{k_q}(q)$, respectively. The cosine similarity is typically adopted to determine the similarity of the query vector and a vector in the database of the text search engine, and it is defined as:

$$sim_{cos}(\widetilde{O}^{k_x}(x), \widetilde{Q}^{k_q}(q)) = \frac{\widetilde{O}^{k_x}(x) * \widetilde{Q}^{k_q}(q)}{\|\widetilde{O}^{k_x}(x)\| \; \widetilde{O}^{k_q}(q)\|},$$

where $*$ is the scalar product. sim_{cos} can be used as a function that evaluates the similarity of the two ranked lists in the same way as $d_\rho(x, q)$ defined in Eq. (2) does (although it is defined as a distance), and it is possible to prove that the first one is an order reversing monotonic transformation of the second one, and then that they are equivalent for practical aspects[3]. This means that if we use $d_\rho(\widetilde{O}^{k_x}(x), \widetilde{O}^{k_q}(q))$ and we take the first k nearest metric objects from dataset (i.e, from the shortest distance to the highest) we obtain exactly the same descriptors in the same order if we use $sim_{cos}(\widetilde{O}^{k_x}(x), \widetilde{Q}^{k_q}(q))$ and take the first k similar objects (i.e., the greater values to the smaller ones). This is illustrated in Figure 1c). The proof of this proposition is omitted due to space limitations of this paper but may be demonstrated using simple mathematical steps. To have an idea on how these textual representations look like, consider the example reported in Figure 1, and let us assume $\tau_1 = \text{RO1}$, $\tau_2 = \text{RO2}$, etc. The function t will generate the following output

[3] To be precise, it is possible to prove that $sim_{cos}(x, q)$ is an order reversing monotonic transformation of $d_\rho^2(x, q)$. However, since $d_\rho(x, q)$ is monotonous this does not affect the ordering.

$t^5(x_1)$ = "RO5 RO5 RO5 RO5 RO5 RO2 RO2 RO2 RO2 RO1 RO1 RO1 RO3 RO3 RO4"
$t^5(x_2)$ = "RO4 RO4 RO4 RO4 RO4 RO3 RO3 RO3 RO3 RO5 RO5 RO5 RO1 RO1 RO2"
$t^5(x_3)$ = "RO5 RO5 RO5 RO5 RO5 RO2 RO2 RO2 RO2 RO3 RO3 RO3 RO1 RO1 RO4"
$t^5(x_4)$ = "RO3 RO3 RO3 RO3 RO3 RO5 RO5 RO5 RO5 RO2 RO2 RO2 RO1 RO1 RO4"

and for the query q:

$t^5(q)$ = "RO5 RO5 RO5 RO5 RO5 RO1 RO1 RO1 RO1 RO2 RO2 RO2 RO3 RO3 RO4"

If we exploit the idea of taking just the closest reference objects to represent any object that has to be indexed, and assuming, for instance, $k_x = 3$ (the number of reference objects used for indexing), and $k_q = 2$ (the number of reference objects used for generating the query), the textual representations become:

$t^3(x_1)$ = "RO5 RO5 RO5 RO2 RO2 RO1"
$t^3(x_2)$ = "RO4 RO4 RO4 RO3 RO3 RO5"
$t^3(x_3)$ = "RO5 RO5 RO5 RO2 RO2 RO3"
$t^3(x_4)$ = "RO3 RO3 RO3 RO5 RO5 RO2"

and for the query q:

$t^2(q)$ = "RO5 RO5 RO1"

This representation of an object will be clearly smaller than using all reference objects. In addition, this has also the effect of reducing the size of the inverted file. In fact, every object will be just inserted into k_x posting lists, by reducing their size and by also reducing the search cost.

2.1 Reordering Search Result

The algorithms described so far use an object representation in a transformed space and an object similarity measure based on a variation of the d_ρ measure to order the objects in the dataset in decreasing similarity with respect to the query. The result is an approximation of the exact result set that would have been obtained if the ordering of the objects was performed using the original distance d in the original data space.

Suppose we are searching for the k most similar (nearest neighbors) objects to the query. We can improve the quality of the approximation by re-ranking, using the original distance function d, the first c ($c \geq k$) objects from the approximate result set at the cost of c more disk accesses and c distance computations. We will show that this technique significantly improves the accuracy, though only requiring a very low search cost. In fact, when c is much smaller than the size of the dataset, this extra cost can be considered negligible with respect to the cost of accessing the inverted file. For instance, when k is 10 and $c = 1000$, with a dataset size of 1,000,000 it means that we have to reorder a number of objects equivalent to just 0.1% of the entire dataset. Usually, as we will see in the experiments, this is not true for other access methods, for instance tree-based access methods, where the efficiency of the search algorithms strongly depends on the amount of objects retrieved.

3 A Real Application and Performance Evaluation

In this section, we report the results of an experimental evaluation of the proposed method. For both testing and demonstration, we developed a web user interface to perform image content based retrieval on the CoPhIR dataset [3], which consists of 106 millions images, taken from Flickr (`www.flickr.com`), described by MPEG-7 visual descriptors. Content based retrieval can be performed by using similarity functions of the visual descriptors associated with the images.

We have indexed the whole CoPhIR dataset and for each image, we created five Lucene fields which can be queried separately or in combination. The first field contains the unique identifier of the Flickr image. The second field maintains the textual information taken from title, and tags of the original Flickr image. The other three fields contain the content generated by the t function explained above for searching on three different pre-combined visual features. In particular, in order to support content based search, the CoPhIR project extracted several MPEG-7 visual descriptors from each image, three descriptors for describing the colors (SCD, CSD, and CLD) and two for describing textures (EHD and HTD). We have indexed three different aggregations of those descriptors, the first one combining the three color descriptors, the second one combining the two texture descriptors, and the third one combining all five descriptors. In this way we leave the possibility to the user to search for colors and textures independently or to search all the descriptors together. The weights used for aggregating the descriptors are the ones suggested in [2].

At the address `http://lucignolo.isti.cnr.it/` a demo web application of the developed search engine can be found. From that page it is possible to perform a full-text search, a similarity search starting from one of the random selected images. Besides the three types of visual similarities, thanks to the search functionality of Lucene, it also provides complex query processing by combining any of the three types of similarity search with the full-text search on descriptive metadata.

We conducted our experiments using the combination of all visual descriptors, with 20,000 reference objects and by setting $k_x = 50$ during the indexing. We used the measure of the recall to assess the accuracy of the method. Specifically, given a query object q, the recall is defined as $R = \frac{\#(S \cap S^A)}{\#S}$, where S and S^A are the ordering of the k closest objects to q found respectively by the exact similarity and by the proposed method. In practice, we compare the efficacy of our solution with an algorithm that exploits a sequential scan of the whole database. The comparison was made at the same conditions, using only the similarity obtained as combination of all five MPEG-7 descriptors, without exploiting the textual content. For this purpose 100 queries were randomly selected from the database. Results are shown in Figure 2. The graphs show the recall varying the number of items retrieved k for various options of the $k_q \leq k$. The graph on the left shows the recall of the basic implementation without reordering. The graph on the right shows the performance of the recall when the reordering strategy is used, with $c = 2,000$.

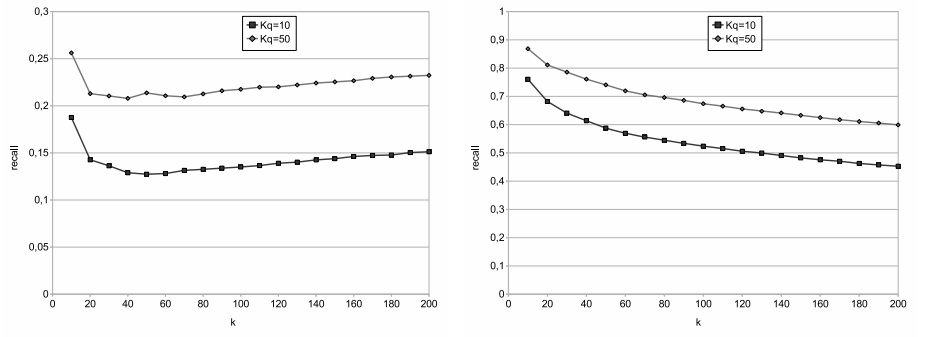

Fig. 2. Recall varying the number k for different values of k_q parameter. Basic (left), and Reorder (right).

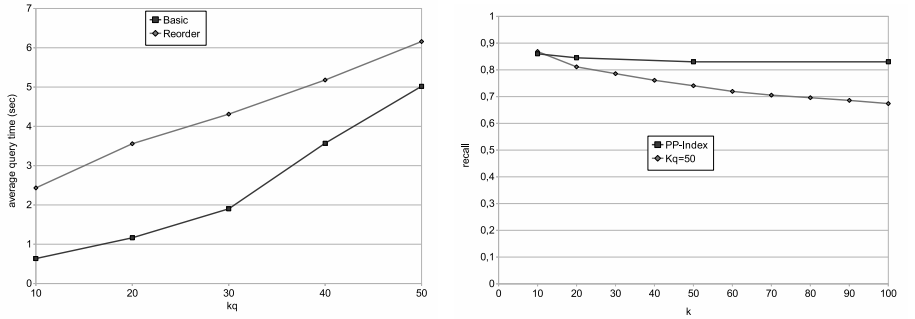

Fig. 3. Query time for different values of k_q parameter (left) and comparison between our approach and PP-Index on the same data set (right)

Figure 3 (left graph) also shows the average query processing times as function of k_q, with and without reordering. As expected, the search cost is worse when use the reordering strategy but still acceptable, also considering the big improvement in terms of recall.

3.1 Comparison with PP-Index

A similar approach [6] (based on the on representing any indexed object with its view of the surrounding world), called Permutation Prefix Index (PP-Index), uses an index data structure that supports efficient approximate similarity search.

Figure 3 (right graph) shows the comparison of the recall between our approach and PP-Index on the CoPhIR dataset. Actually, PP-Index exhibits better performance. However, as explained in the introduction, the aim of our approach is to provide a tool for rapid development and integration of a multimedia object retrieval system with other digital libraries based on text. As a result, along

with the obvious advantage of having a system which relies upon an open source library that is constantly expanding, our method provides content based search combined with textual metadata.

4 Conclusions and Future Work

In this paper we presented an approach to approximate similarity search in metric spaces based on a space transformation that relies on the idea of perspective from a data point. We proved through a concrete implementation that the proposed approach has clear advantages over other methods existing in literature in terms of easiness in implementation. A major characteristic of the proposed technique is that it can be implemented by using inverted files, thus capitalizing on existing software investments.

This approach can take advantage of parallelism of Lucene and easily scales up to any desired dataset size. This can be obtained by distributing the inverted index in multiple Lucene segment, and exploding parallel search facilities of Lucene. For instance, our index consists of ten separated Lucene indexes each one including about 1/10 of the whole dataset. If the indexes reside on different physical disks, we may obtain performance improvements; however, in our tests conducted with a single physical disk, the performance with multi-thread search was slightly better than with a single-thread search.

References

1. Amato, G., Savino, P.: Approximate similarity search in metric spaces using inverted files. In: Proceedings of the 3rd International Conference on Scalable Information Systems (InfoScale 2008), pp. 1–10. ICST (2008)
2. Batko, M., Kohoutkova, P., Novak, D.: Cophir image collection under the microscope. In: International Workshop on Similarity Search and Applications, pp. 47–54 (2009)
3. Bolettieri, P., Esuli, A., Falchi, F., Lucchese, C., Perego, R., Rabitti, F.: Enabling content-based image retrieval in very large digital libraries. In: Second Workshop on Very Large Digital Libraries (VLDL 2009), pp. 43–50. DELOS (2009)
4. Chavez, E., Figueroa, K., Navarro, G.: Effective proximity retrieval by ordering permutations. IEEE Transactions on Pattern Analysis and Machine Intelligence 30, 1647–1658 (2007)
5. Esuli, A.: Pp-index: Using permutation prefixes for efficient and scalable approximate similarity search. In: Proceedings of the 7th Workshop on Large-Scale Distributed Systems for Information Retrieval (LSDS-IR 2009), pp. 17–24 (2009)
6. Esuli, A.: Use of permutation prefixes for efficient and scalable approximate similarity search. Information Processing & Management (2011)
7. Fagin, R., Kumar, R., Sivakumar, D.: Comparing top-k lists. SIAM J. of Discrete Math. 17(1), 134–160 (2003)