Integrating text retrieval and image retrieval in XML document searching

Integrating Text Retrieval and Image Retrieval in XML Document Searching

D. Tjondronegoro, J. Zhang, J. Gu[1], A. Nguyen, S. Geva

Queensland University of Technology
2 George Street, GPO Box 2434, Brisbane, QLD 4001 Australia
{dian, jinglan.zhang, j2.gu, an.nguyen, s.geva}@qut.edu.au

Abstract. Many XML documents contain a mixture of text and images. Images play an important role in webpage or article presentation. However, popular Information Retrieval systems still largely depend on pure text retrieval as it is believed that text descriptions including body text and the caption of images contain precise information. On the other hand, images are more attractive and easier to understand than pure text. We assume that if the image content is used in addition to the pure text-based retrieval, the retrieval result should be better than text-only or image-only retrieval. We test this hypothesis by doing a series of experiments using the Lonely Planet XML document collection. Two search engines, an XML document search engine using both content and structure based on text, and a content-based image search engine were used at the same time. The results generated by these two search engines were merged together to form a new result. This paper presents our current work, initial results and vision into future work.

1 Introduction

Researchers have studied text retrieval for decades [1]. When text is input into a web browser, the search engine will output something related with the text. However, a traditional query for simple text documents can only return the entire document. In order to improve the functionality and precision of simple text retrieval, the Extensible Markup Language (XML) has been adopted as an industry standard for document formatting by W3C. The XML format contains the document content and metadata. It marks up semantic document elements such as document, paragraph, images, maps, etc. It contains structures and allows exploiting the internal structure in order to find keywords in certain document elements. For example, a traditional document based query can be “return a document that contains an image like this one”. An XML document can support the same query. However, it can also support other types of queries requesting more details, e.g. “return the paragraph that contains an image like this one” or “return the images that are similar to the sample image”. Structured document searching has attracted intensive research [2-3].

[1] Supported by Jiangsu Government Scholarship of Overseas Studies.

A picture is worth a thousand words, and many documents therefore contain a mixture of text and images. For example, almost every webpage contains text and images so that they are more attractive and easier to understand than a page full of text. Images play an important role in webpage or article presentation. However, this information has not been explored sufficiently in traditional Information Retrieval systems, which largely depend on text searching. Many research issues are still open.

Although images can be retrieved using metadata such as captions, authors, date, text description, etc, text-based image retrieval has obvious disadvantages. It is not easy to define a unique set of keywords to one image and when the image database is large it is time consuming to add the keywords. Images can also be retrieved based on its content using features such as colour, texture, and shape extracted from the image and stored in the database. If multiple features are used, the result should be fused to form a single result. Some fuzzy-neural techniques by fusion of texture and colour have been explored in image retrieval [4].

In general, the more information that is used to generate a query, the stricter it is and the more precise the searching results should be. For example, “an excellent IT student” is stricter than “an IT student” which is again stricter than “a student”. The text descriptions in an XML document, including the text content or the caption of the image, usually contain more precise information about an image. However, often more information is captured in the image than described in the caption. We assume that if the image content is used in addition to the pure text-based retrieval, the retrieval result should be better than text-only or image-only retrieval in terms of precision/recall. As image specification in the query makes the query stricter than the query without the image, we assume image elements can be treated as if they were text elements containing ordinary keywords.

We propose to use two search engines, an XML document search engine using both content and structure based on text, and a content-based image search engine, at the same time. Multiple tests and image processing algorithms can be implemented. The results generated by these two search engines should be merged together to form a new result. The goal is to include and integrate multiple media types into the retrieval process and to produce a meaningful ranking of documents. This paper presents our current work, initial results, some findings and vision into future work.

The remainder of this paper is organised as follows: Section 2 presents the overall system framework and Section 3 briefly discusses the database management. Section 4 then discusses the XML document searching techniques, while the image processing techniques used are described in Section 5. The result fusion technique used is presented in Section 6, and the implementation and testing is given in Section 7. Finally, Section 8 concludes the paper with a discussion on the evaluation, main lessons learned from the experiments, and vision for future work.

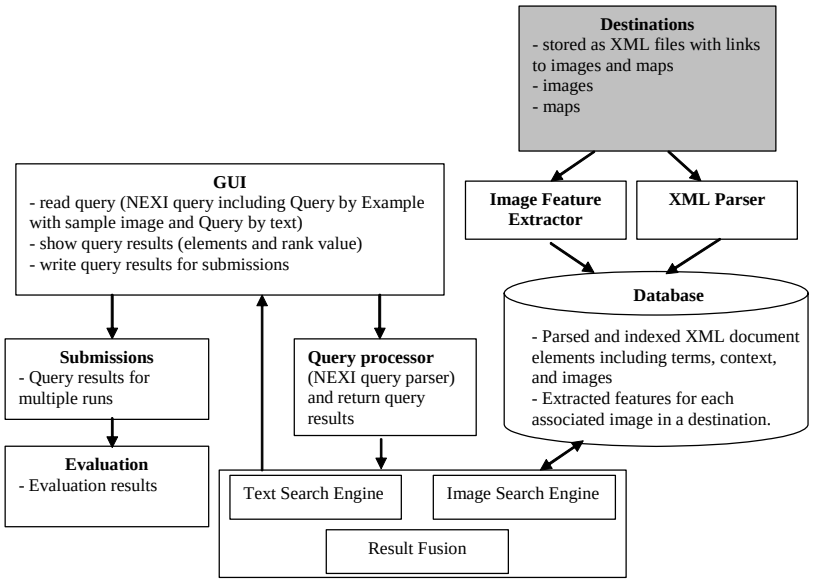

Fig. 1. System Framework

2 System Framework

The diagram shown in Fig. 1 shows the framework and data flow of the prototype system used for the work reported in this paper.

3 Database Management of XML Documents

An existing relational database management system is used to manage the XML documents. XML documents are parsed and document elements are stored and indexed in a database. The structure of the database is shown in Fig. 2. It contains six tables. The **Terms** table is linked to the **List** table through the List_Position field. The document ID in the **Document** table and the context ID in the **Context** table are the foreign keys of the Doc_ID and Context_ID fields in the **List** table respectively. The context ID is also the foreign key of the Context_ID field in the **ImageContext** table. The image file name in the **Images** table is a foreign key of the Term field in the **ImageContext** table.

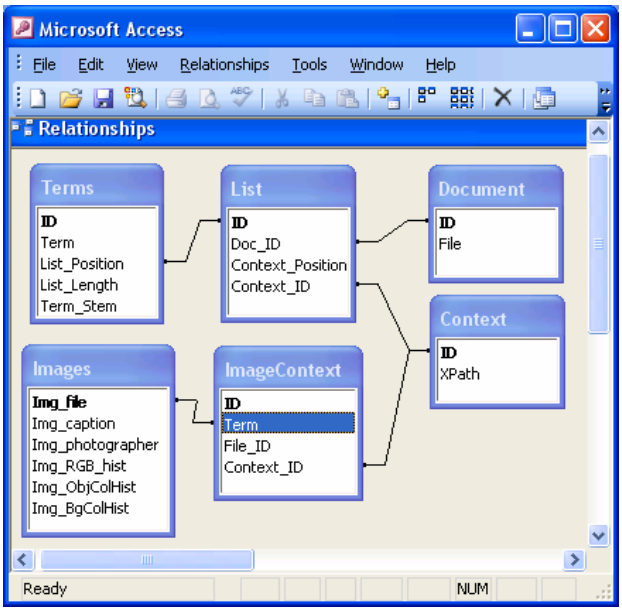

Fig. 2. Database Structure for text- and image-based XML Document Retrieval.

The text-based searching can start from Term in the **Terms** table, i.e., keywords provided in the query such as mountain, water, etc. Following the links in Fig. 2, we can find where the terms appear, how many times they appear, and retrieve the context of each occurrence. The image-based searching will start from the **Images** table. Given an image name, we can retrieve all its features and calculate the similarity between a sample image and an image in the database. If an image is desirable, we can return the image itself or the context of that image. We assume the same image can appear in multiple documents but within one document, an image just needs to appear once.

4 XML Searching Techniques

4.1 XML File Inversion

In our scheme each term posting in the collection consists of three path elements: the file name, the absolute XPath context, and the ordinal position within the XPath context. The entire collection is inverted and the indexing structure supporting access to

the terms inverted lists is stored in a MS Access database (see Fig. 2). Details of the file inversion technique can be found in [5].

4.2 Processing INEX Queries

Processing of complex NEXI expressions is based on parsing of the expression and the incremental construction of a result-tree. The result-tree consists of all the elements in the collection that contains at least one of the keyword in the query (or a synonym or any other term deemed relevant). Each node in the result tree contains the necessary information to allow the computation of a score, using a TF-IDF variant (described in Sect. 3). After the result-tree is constructed, a traversal of the result-tree generates the score for each node, from the leaves to the root node. These results are then organized as a list and sorted by score, with the top N results returned (N = 250 for the MM track).

When a NEXI expression contains multiple filters, the system constructs a result-tree for each of the filters. After the score of each node in all trees is determined, the scores of support elements (i.e. elements that satisfy a support filter in the NEXI expression) are used to boost the score of result elements. In this manner, elements with support tend to be ranked higher than elements without support, everything else being equal. More specific details can be found in the paper describing our submission to the ad hoc track, in these proceedings.

4.3 Ranking Scheme

Elements are ranked according to a relevance score. In our scheme, leaf and branch elements need to be treated differently. Data usually occurs at leaf elements, and thus, our inverted list mostly stores information about leaf elements. A leaf element is considered candidate for retrieval if it contains at least one query term. A branch node is candidate for retrieval if it contains a relevant child element. Once an element (either leaf or branch) is deemed to be a candidate for retrieval its relevance score is calculated. A heuristically derived formula (1) is used to calculate the relevance score of leaf elements. The same equation is used for both return and support elements. The score is determined from query terms contained in the element. It penalizes elements with frequently occurring query terms (frequent in the collection), and it rewards elements with more unique query terms within a result element.

Equation 1: Calculation of a Leaf Element's Relevance Judgment Score

$$L = K^{n-1} \sum_{i=1}^{n} \frac{t_i}{f_i} \qquad \textbf{(1)}$$

Here n is the number of unique query terms contained within the leaf element, K is a small integer (we used K = 5). The term K^{n-1} scales up the score of elements having

multiple distinct query terms. The system is not sensitive to the value of K – we experimented with $K = 5$ to 25 with little difference in results. The sum is over all terms where t_i is the frequency of the i^{th} query term in the leaf element and f_i is the frequency of the i^{th} query term in the collection. This sum rewards the repeat occurrence of query terms with uncommon terms contributing more than common terms.

Once the relevance scores of leaf elements have been calculated, they can be used to calculate the relevance judgment score of branch elements. A naive solution would be to just sum the relevance judgment score of each branch relevant children. However, this would ultimately result in root (i.e. article) elements accumulating at the top of the ranked list, a scenario that offers no advantage over document-level retrieval. Therefore, the relevance score of children elements should be somehow decreased while being propagated up the XML tree. A heuristically derived formula (2) is used to calculate the scores of intermediate branch elements.

Equation 2: Calculation of a Branch Element's Relevance Judgment Score

$$R = D(n)\sum_{i=1}^{n} L_i \qquad (2)$$

Where:

n = the number of children elements
$D(n)$ = 0.49 if $n = 1$
0.99 Otherwise
L_i = the i^{th} return child element

The value of the decay factor D depends on the number of relevant children that the branch has. If the branch has one relevant child then the decay constant is 0.49. A branch with only one relevant child will be ranked lower than its child. If the branch has multiple relevant children the decay factor is 0.99. A branch with many relevant children will be ranked higher than its descendants. Thus, a section with a single relevant paragraph would be judged less relevant than the paragraph itself, but a section with several relevant paragraphs will be ranked higher than any of the paragraphs.

Having computed scores for all results and support elements, the scores of support elements are added to the scores of the corresponding result elements that they support. For instance, consider the query:

//A[about(.//B,C)]//X[about(.//Y,Z)]

The score of a support element //**A//B** will be added to all result elements **//A//X//Y** where the element **A** is the ancestor of both **B** and **X//Y**.

Finally, the results consist of an entire recall tree for the query where each node is individually scored.

5 Image Processing Techniques

The image features, which have been selected for this experiment are: colour histogram, texture, and detectable-lines. In our database, for each image, we store these features to describe the whole image, the background and the foreground.

5.1 Colour Histogram ($p(r_k)$)

The histogram of a digital image is defined as:

$$h(r_k) = n_k$$

Where r_k is the k^{th} intensity level in the interval [0, *H*] and n_k is the number of pixels in the image with intensity level equal to r_k . The value of *H* depends on the image class (i.e. the number of bits used); for example, *H* = 255 when 8 bits are used to define a colour. It is common for the histogram to have *N* (< 256 for an 8 bit/pixel image) equally spaced bins, each representing a range of data values. The histogram, in this case, would calculate the number of pixels within each range.

A normalized histogram is obtained by dividing all elements of $h(r_k)$ by the total number of pixels in the image, which is denoted by n :

$$p(r_k) = \frac{h(r_k)}{n}$$

Since colour images were used, $p(r_k)$ was computed for each of the red (R), green (G), and blue (B) components. Sixteen bins (*N* = 16) were used such that 48 histogram features were generated.

5.2 Texture ($\Im$)

Texture analysis is frequently based on statistical properties of the intensity histogram. One of the principal approaches for describing the shape of a histogram is by using its *central moments*.

Let r_i be a discrete random variable that indicates intensity levels in an image, and $p(r_i)$ be the corresponding normalized histogram. Given that the range of r_i is 0 to *G*-1 where *G* is the number of possible intensity values, a histogram component, $p(r_i)$, is an estimate of the probability of occurrence of intensity value r_i. Thus, central moments can be defined as:

Table 1. Moments used for texture measurement

Moment	Texture Measurement	Equation
Mean	Average intensity	$m = \sum_{i=0}^{G-1} r_i p(r_i)$
Standard deviation	Average contrast	$\sigma = \sqrt{\mu_2(r)}$
Smoothness	Relative smoothness of the intensity of a region (0 is for constant intensity and closer to 1 for regions with large excursions in the intensity levels)	$\zeta = 1 - \frac{1}{(1+\sigma^2)}$
Third moment	Skewness of histogram (0 is symmetric, negative means skewed to the left whereas positive means skewed to the right)	$\mu_3 = \sum_{i=0}^{G-1} (r_i - m)^3 p(r_i)$
Uniformity	Maximum when all gray levels are equal (maximum uniformity) and decreases otherwise	$U = \sum_{i=0}^{G=1} p^2(r_i)$
Entropy	Measurement of randomness	$\Re = \sum_{i=0}^{G=1} p(r_i) \log_2 p(r_i)$

$$\mu_n = \sum_{i=0}^{G-1} (r_i - m)^n p(r_i)$$

Where, *n* is the order of the moment, and *m* is the mean.

As we are using normalized histograms, the sum of all its components is 1, thus $\mu_0 = 1, \mu_1 = 0$, and:

$$\mu_2 = \sum_{i=0}^{G-1} (r_i - m)^2 p(r_i)\text{; and so on.}$$

The higher order moments which can effectively measure texture is described in Table 1. Hence texture is represented by a feature vector of:

$$\Im = [m, \sigma, \zeta, \mu_3, U, \Re]$$

5.3 Detectable Vertical-Horizontal Lines (α)

Hough transform computes projections of an image along specified directions. We can detect straight lines which are formed by strong edges on the edge image (which can be produced by *Sobel* transform).

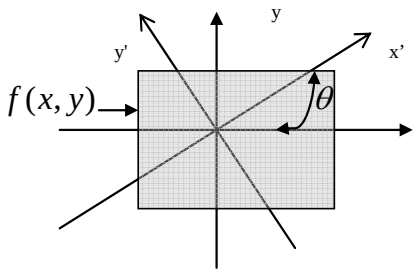

Fig. 3. Geometry of Hough Transform.

A projection of a two-dimensional function $f(x, y)$ is a line integral in a certain direction (see Fig. 3). Projections can be computed along any angle θ. The Hough transform of $f(x, y)$ is the line integral of f parallel to the y' axis.

$$H_\theta(x') = \int f(x'\cos\theta - y'\sin\theta, x'\sin\theta + y'\cos\theta)dy'$$

For the purpose of distinguishing images with building(s) as the main object, we have used the statistics of the horizontal and vertical lines. As shown by examples in Fig. 4, images with buildings have stronger detectable horizontal-vertical lines compared to the images with no buildings. It should be noted, nonetheless, that this feature is robust for measuring similarity of other image types, not just for distinguishing buildings. The edge images generated by the process captures the structure of objects and thus the vertical-horizontal line features can be also be used to distinguish between non-building objects.

To capture the characteristics of vertical-horizontal lines of an image, we have used:

$$\alpha = [Vert\{avg, \min, \max, std\}, Horz\{avg, \min, \max, std\}]$$

Where *avg*, *min*, *max*, and *std* are the average (mean), minimum, maximum, and standard deviation of the vertical (*Vert*) and horizontal (*Horz*) line's strength. Since the Hough transform returns two vectors containing the strength and the corresponding coordinates along the x'-axis (i.e. the angle), the vertical and horizontal line features are obtained when the angle, θ = 180° and 90° respectively.

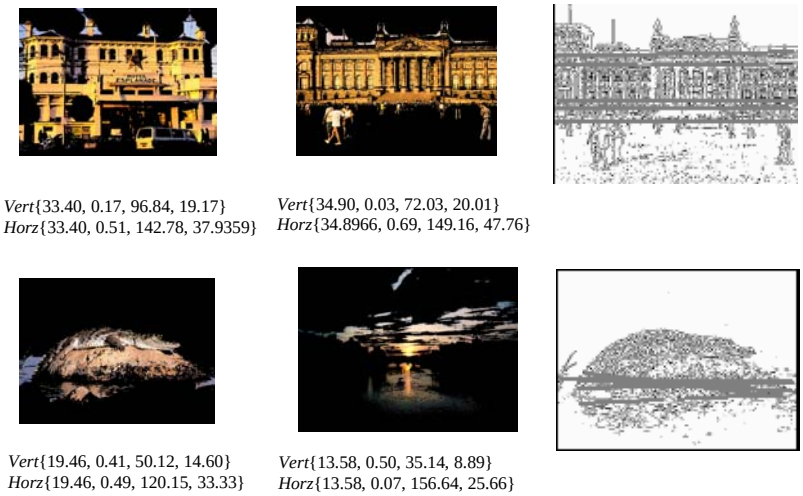

Fig. 4. Vertical-Horizontal Line Features in Images.

5.4 Object Extraction

Object(s) can be extracted from an image using the following algorithm:

1. Create a morphological structuring element (*SE*) with a radius of *R* pixels (experimentally set to disk-shaped with *R* = 20).
2. Temporary background (*BG'*) is created by removing objects in original image (*I*) with a radius less than *R* pixels by opening it with the structuring element created in step 1. This operation also estimates the background illumination.
3. Subtract *BG'* from *I* to create the temporary foreground (*FG'*).
4. Binary foreground (FG") is generated by converting FG' into binary by applying *Otsu*'s global image threshold[2].
5. To create the final foreground image (*FG*):
   - For every pixels in FG", if the value = 1, restore the original intensity value of the pixel (based on *I*); Else preset the intensity value to a non-FG value such as black (e.g. R = 0, G = 0, B = 0).
6. To create the final background image (BG):
   - For every pixels in FG", if the value = 0, restore the original intensity value of the pixel (based on *I*); Else preset the intensity value to a non-BG value such as black (e.g. R = 0, G = 0, B = 0).

[2] Otsu, N., "A Threshold Selection Method from Gray-Level Histograms," IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, no. 1, 1979, pp. 62-66.

Fig. 5. The Process of Object(s) Extraction.

The effectiveness of the object extraction algorithm can be demonstrated by the sample scenarios shown in Figs 5 to 8.

5.5 Similarity Calculation and Ranking

Although the physical meanings of different image features are different, they can all be represented by vectors. We used one vector to describe each feature of the image. Thus distance calculation approach can be applied uniformly for different features. We simply use the Euclidean Distance between two vectors, i.e. $d(v_1, v_2) = \sqrt{\sum_i (v_1(i) - v_2(i))^2}$ for all elements in the two vectors. As all images are transformed to the same size, the distance calculation is size-invariant and the two vectors are always the same length. All distances are normalized to the range of [0, 1].

6 Result Fusion

The additional complication that is introduced through the addition of image retrieval cues in the NEXI expression is that we need to combine the scores of image elements with the scores of text elements. Rather than re-write much of our code we have decided to treat image elements as if they were text elements containing ordinary keywords. When a NEXI filter specified an image as a retrieval cue we proceeded to generate element scores in two steps. In the first stage, we used image processing techniques to rank the images in the collection for similarity (using features described in Sect. 5). Images were ranked in the range [0, 1]. In the second stage, we heuristi-

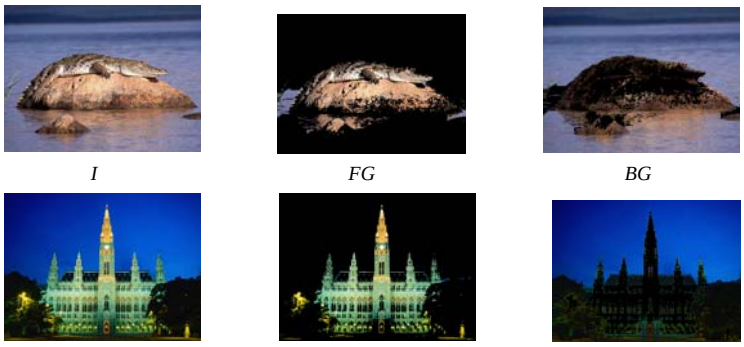

Fig. 6. Images with one dominant object.

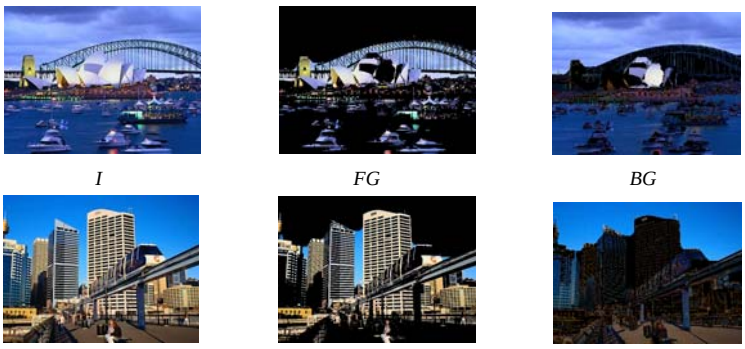

Fig. 7. Images with multiple objects of interest.

Fig. 8. Images with background as a more dominant feature.

cally assigned to each image element a pseudo-term frequency, just like we assign a term frequency to every node that contains a given term when we process a textual node. With this, an image node that was ranked first on the basis of image features

can be made more dominant by being assigned a pseudo term frequency of say 5, while an image node ranked 250 might be assigned a node frequency of 1. In this manner, we were able to compute a node score as if it consisted of text elements, and we could then seamlessly apply the text ranking scheme that we used for text retrieval.

Table 2. Summary results for INEX MM submissions.

Measures†	QUTAU-0	QUTAU-1	QUTAU-2	QUTAU-3	QUTAU-4	QUTAU-5
topics	19	19	19	19	19	19
Retrieved	3767	3793	3366	3882	4009	4132
Relevant	448	448	448	448	448	448
Rel. ret.	297	297	303	300	285	266
map	0.1995	0.2064	0.2711	0.1844	0.2037	0.2066
R-prec	0.2094	0.2116	0.2641	0.1892	0.1986	0.2181
bpref	0.6507	0.6518	0.6516	0.6647	0.6501	0.6319
Recip_rank	0.4657	0.504	0.5414	0.4561	0.4901	0.5134

† *Retrieved* – Total number of document fragments retrieved over all queries; *Relevant* – Total number of relevant document fragments over all queries; *Rel. ret.* – Total number of relevant document fragments retrieved over all queries; *map* – Mean Average Precision; *R-prec* – R-Precision (Precision after R (= number of relevant for topic) document fragments retrieved); *bpref* – Binary Preference (top R judged non-relevant); *recip_rank* – Reciprocal rank of top relevant document.

In our submissions we experimented with several variations over the set of image features used, the length of the candidate images list to be considered, and the pseudo term frequency. The length of candidate images list was varied because the list was always long, but you would run out of relevant or really similar images pretty quickly. Therefore, there was no point in considering too many image elements (though there were only a couple of thousand images in the collection anyway). The pseudo term frequency was varied to provide different weightings between the image elements relative to the text elements. These submissions basically varied in the above parameters and were labelled QUTAU-0, QUTAU-1, QUTAU-3, QUTAU-4, and QUTAU-5. We also submitted a baseline text only submission (QUTAU-2) that did not take into account any image features. The hypothesis is of course that the use of image features would improve precision/recall.

The assessment of the fusion quality was conducted manually based on statistical (e.g. term frequency) and visual criteria (e.g. image) by human assessors through comparing the retrieval results of different combinations between text and image retrieval with varying weight factors. Along the spectrum of this combination, the two extreme ends are the results of text-retrieval-only and image-retrieval-only. A few discrete points in between were also defined.

An online evaluation tool provided by INEX2005 organizers was used as the evaluation environment. Fig. 9 shows the Interpolated Recall Precision Averages (IRPA), which shows the precision at various recall levels for the different submissions. A summary of the results is tabulated in Table 2.

The text-retrieval-only system can be observed to be superior at all recall levels compared to the fused text- and image-retrieval systems. This is due to the text in the document body and captions of being too well tagged and annotated, and thus high

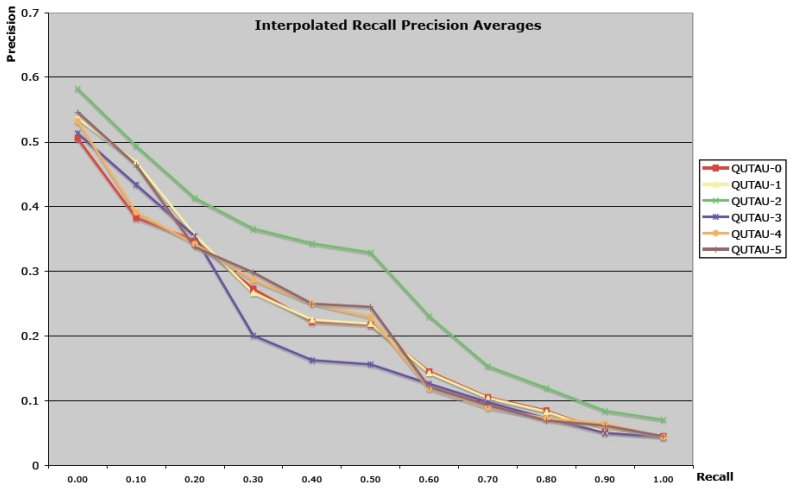

Fig. 9. Recall-Precision results for INEX MM submissions.

recall performance can be achieved with the text alone. The addition of image retrieval results, on the other hand, seems to degrade performance. The difference between the various fused text and image systems are not statistically significant, but in general performed better when a higher text retrieval weighting was used. The results do not imply that image retrieval is not effective nor does it mean that we reject our hypothesis that image features would improve precision/recall; the text was simply too well annotated to allow image features to improve recall. Despite this, image retrieval cues showed potential from their recall performances. This is promising since no semantic or higher level image processing was incorporated into the image retrieval process.

7 Implementation and Testing

The prototype system was implemented on a PC using an existing database management system (Microsoft ACCESS). The C# programming language in the .NET framework was used to implement the text search engine. The image search engine is also implemented using C#, while the image feature extractor is implemented using the MATLAB programming language. Both MATLAB and C# support access to the database.

The test dataset used was a comprehensive database of XML documents containing text- and image- based information on holiday destinations, namely, the Lonely Planet XML document collection. This collection was provided by INEX2005 orga-

nizing committee. This database contains a total of 463 XML documents and 1947 images, which contain various types of contents such as landscape, people, and buildings.

8 Conclusion and Future work

The retrieval quality of the search engine was evaluated through standard evaluation measures using various precision/recall metrics. The retrieval results including text and images were manually (and blindly) scored by independent assessors. This is done through viewing the images and the text, and judging relevance against a stated information need that is associated with each query. An online evaluation tool provided by INEX2005 organizers was used as the evaluation environment.

XML document fusion usually refers to the fusion of the retrieval results of multiple searching algorithms on the same collection or the fusion of search results from multiple document collections. Our work involves XML document fusion in the former case as only one single document collection was used and each XML document was unique.

The experimental results show that the fusion of two result sets is not a trivial task. More complicated fusion algorithms such as Principal Component Analysis need to be explored in the future. A systematic research on metrics (criteria) for assessing fusion quality also needs to be conducted. It is still a great challenge on the objective and automatic assessment of the fusion quality.

In addition, the performance of the system at low recall levels need to be studied, since image similarity is a very limited retrieval cue and a long tail of similar images will unlikely produce improved results. Despite this, the experiments have shown that the image retrieval cues performed relatively well. Higher-level image understanding techniques will likely enhance the retrieval process and is a subject of future investigations.

The current INEX image collection also requires revision since it was not large enough, and secondly, it was too well annotated to allow image features to improve recall. As such no system was able to improve on the text-only alternative. However, it is envisaged that with further investigations on a larger database, the use of image content in addition to pure text-retrieval, should lead to significant improvement in retrieval performance results.

References

1. T. Bray, J. Paoli, and C. Sperberg-McQueen, Extensible Markup Language (XML) 1.0, W3C recommendation, (1998)
2. I. Schlieder and H. Meuss, Querying and ranking XML document. Journal of the American Society for Information Science and Technology, 53(6), (2002), 489-503
3. A. Trotman, Searching structured documents. Journal of Information Processing and Management 40, (2004), 619-632.

4. B. Verma, S.Kulkarni, A fuzzy-neural approach for interpretation and fusion of colour and texture features for CBIR system, Applied Soft Computing 5, (2004), 119-130.
5. S. Geva, GPX-Gardens Point XML Information Retrieval at INEX2004, Advances in XML Information Retrieval – Third International workshop of the Initiative for the Evaluation of XML Retrieval (INEX), (2004), 211-223.