Estimating Missing Features to Improve Multimedia Information Retrieval

Abraham Bagherjeiran Nicole S. Love Chandrika Kamath *

Abstract

Retrieval in a multimedia database usually involves combining information from different modalities of data, such as text and images. However, all modalities of the data may not be available to form the query. The retrieval results from such a partial query are often less than satisfactory. In this paper, we present an approach to complete a partial query by estimating the missing features in the query. Our experiments with a database of images and their associated captions show that, with an initial text-only query, our completion method has similar performance to a full query with both image and text features. In addition, when we use relevance feedback, our approach outperforms the results obtained using a full query.

Keywords: multimedia information retrieval, text and image mining.

1 Introduction

A common problem in multimedia retrieval is that of finding a face for a name so that we can determine if the face is present in other images without the associated name. A related problem is one where we are given a face, and we are interested in associating a name with the face so we can determine if the name appears in any text documents.

This task of finding a name given a face or a face given a name in a database of documents, each containing one or more captioned images, can be difficult for many reasons. If we know the name of a person, we could perform the retrieval by focusing on just the text in the document, ignoring any images and their associated captions. This could return incorrect results in two common cases. First, the document contains the name, but the associated image does not contain the picture of the person. Second, the image of the person is present, but the associated text does not mention the person by name.

Instead of the text of the entire document, we could consider just the text in the caption. This is a reasonable assumption as captions often provide a concise summary of the events in the image. It also reduces the search space of available words considerably. So, if we are seeking a face to match a name, or a name to match a face, considering just the associated caption with the image may suffice.

It is easy to see why we need to consider both the image and the text in the caption. In some cases, a caption may include the name of a person, but the associated image may not include the person. For example, Figure 2(b) might have a caption "President Bush applauds the observance of Financial Literacy month", while the associated image is a letter from the President to the effect, rather than an image of the President. Focusing on the image, and the faces therein, is not a viable alternative either as there could be many faces in the image, not all of which are listed by name in the caption. Further, it is well known that recognizing faces can be very difficult, given the variation due to changing illumination, different poses and view angles, changing hairstyles, and the presence or absence of makeup or accessories. Also, in a corpus of documents, there are likely to be images where the intensity of pixels is similar to that of a face. An automatic face detector will identify these regions as faces, as shown in Figure 2(a). In light of these considerations, it makes sense to exploit both the text in the caption and the information in the image to improve the performance of retrieval.

Ideally, when we combine the information in both images and the text in the caption of the image, we would expect the best results when we use this combined information in both the query and in the retrieval process. Unfortunately, in many cases, we have only a partial query, where either the text or the image is missing. In this paper, we show how we can estimate the missing part of a query and use it to improve our retrieval results. We focus on the specific problem of retrieving documents composed of images of faces and their associated captions. The query can be just a name or an image containing a face.

This paper is organized as follows: in Section 2, we survey the related work in multimedia retrieval, specifically techniques to combine images and text. In Section 3, we describe the process we use to extract the text features from the caption, followed by the process used to identify faces in images and the features used to represent the faces. In Section 4, we discuss the process of retrieval, including the combination of the text and image features, the similarity measure used in the retrieval, and our approach to query completion. Section 5 describes the data set used in our experiments

*Lawrence Livermore Laboratory, Livermore, CA 94551.

and presents our results, including a simple analysis of the data to understand the effectiveness of the text and the image features. We conclude in Section 6 with a summary and plans for future work.

2 Related work

The increasing availability of documents containing multimedia content has led to research in finding ways of exploiting this complementary information. In particular, recent work has focused on combining the text documents and images to improve the performance of retrieval [7, 10, 23].

One approach is to first obtain the ranking results using separate image and text queries, and then combine the rankings by maximum, minimum, or a linear combination [6, 7, 10, 23]. This was shown to improve performance compared to retrieving either image or text separately.

Efforts have also been made to estimate image queries from text information and estimate text features from the image (auto annotation). These methods primarily focus on modeling the joint probability distribution between words and image regions. In such methods, one assumes that images consist of clusters called blobs and that these blobs follow a known distribution. If we know this distribution, we can do both annotation and retrieval. Although an image is associated with a caption, the same words correspond to blobs in different images. For example, one expects that an image with a tiger has the word tiger in the caption whether or not the tiger is in front of a tree.

Lin *et al.* pose the image-text retrieval problem as one of image-text translation [19]. Rather than using images for queries directly, they translate query words to image blobs and then perform image retrieval. The dataset consists of images with captions in one of two languages: Chinese or English. Since queries can come in either language, image blobs are used as an intermediate language. A probabilistic model links words in captions, separately for each language, to images in the database. A new image is the result of translating the original text into a set of image blobs.

Jeon and Lavrenko assume that text terms depend probabilistically on an image, and an image depends on blobs [13, 17]. By modeling each document as a term-blob distribution, the distance between the query and a document is given by the Kullback-Leibler divergence of their distributions. For separate image or text queries, they estimate values from their model to add text or images before retrieval [13]. They report results for both retrieval and annotation.

Forsyth *et al.* apply several statistical models and use the EM algorithm to learn the association within an image [2, 8]. In their model, an image is a collection of blobs and words. Given the model, they can predict likely words for blobs.

Li and Wang use 2D hidden Markov models to represent clusters of images and associate words with each cluster [18]. A 2D hidden Markov model is trained on each image in the database. The models for several images are clustered, and words from the images are associated with each cluster. To predict words for a new image, the probability that the image is from each cluster is computed. Words are then sampled according to the image and word probabilities.

A popular application of image annotation is to find names for human faces in news photos [3, 9]. The data consist of news photos and their captions. One difficulty with news captions is that they typically describe several people who may or may not be in the photo. To alleviate this problem, names and photos are clustered and common names are associated with each cluster. Similar images and similar names are then linked [3]. A related approach uses co-occurrence statistics to predict when image clusters refer to the same topic in the news [9]. A user can then browse similar articles by looking at the pictures.

These probabilistic methods tend to be rather computationally expensive. In contrast, we present an approach to estimating missing features which is both simple and computationally inexpensive. It allows us to generate a query containing both image and text features from one containing only image features or only text features.

3 Extracting Text and Image Features

In our study, we focused on the images in a document and their associated captions. As our interest is in retrieving related documents given either a name or a face, each item in our database corresponds to a single face in an image. The item is represented by a feature vector v:

$$v = \left(\hat{I}, \vec{t}\right)$$

where $\hat{I}$ is the set of features extracted from a face image I and $\vec{t}$ is the text feature vector derived from the set of text tokens, c, derived from the associated caption. In other words, we combine the text and image features by a simple concatenation. If an image has several faces in it, each face in the image has a copy of the text features derived from the caption, but different image features derived from the individual faces.

3.1 Text Features Given a caption, we extract the text features by stemming the words in the caption, thus

reducing the word to its root form. This is done using Porter's stemming algorithm [25] via the `doc2mat` script described by Karypis et al. in [15]. This script removes common words and finds the root words or tokens in the caption. For example, the caption "President George W. Bush..." becomes the set of tokens $w = \{\text{presi}, \text{georg}, \text{bush}\}$. The stemming algorithm removes single characters such as "W" and converts "president" to "presi". To simplify lookup and further processing, we sort the tokens in alphabetical order. This gives us the sequence of tokens $\langle \text{bush}, \text{georg}, \text{presi} \rangle$. We use these sorted tokens to extract the values of the text features.

We process the tokens using the term frequency inverse document frequency (TFIDF) representation from text retrieval [30]. This represents the text part of the feature vector, $\vec{t}$, as:

$$\begin{aligned} \vec{t} &= (t_1, \ldots, t_{|T|}) \\ t_i &= \begin{cases} 1 - \frac{\log_2 df_i}{\log_2 n} & c_i \in c \\ 0 & c_i \notin c \end{cases} \end{aligned}$$

where T is the set of all tokens in the data set and $c_i \in T$ is the ith token in the set of all tokens for the data set, c is the current set of tokens for the caption, n is the number of documents in the data set, and df_i is the number of documents that contain the token c_i.

3.2 Image Features As mentioned earlier, each item in the database corresponds to a face in an image. The image part of the feature vector is obtained by first finding all the faces in an image, and then extracting the appropriate face features.

Face Detection The first step in extracting the image feature vector is finding the faces in an image. We accomplish this using Mikolajcyzk's face detector [22]. This uses orientation features from the gradient and second derivatives of the Laplacian to form object parts. The object parts are learned using AdaBoost and combined with geometric constraints to determine the likelihood of a face. The detector outputs image blocks that are likely to contain faces. Recall that in our work, each candidate face in the image corresponds to a different item in the database and therefore a different feature vector.

Figure 1 shows an example of the results obtained using the face detector, with the face regions outlined using a square box. We observe that the detector finds the face of President Bush and those of several people in the background. However, it also identifies other regions as faces, as in Figure 2(a), where one of the five regions is not a face. Another typical error occurs when the face detector is applied to the image of a text document as shown in Figure 2(b). Here, non-faces which have a similar intensity distribution as the eyes and mouth in a face are identified as faces.

Figure 1: Sample faces detected in an image. The face regions are highlighted by a box.

Image Feature Extraction Once a candidate face is detected, the face is scaled to 128 × 128 pixels and low-level image features are extracted from the face. These low level features do not entail finding facial features such as the eyes, they are not unique to faces, and they can be extracted easily from any image. The image feature vector, $\hat{I}$, is a concatenation of these low-level image features.

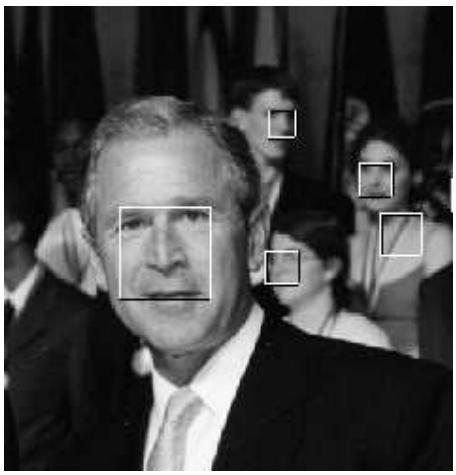
There are several features commonly used in image retrieval, such as histograms or texture features. Based on our early experimentation with retrieving faces in an image database, we found that the best combination was one where we combined the features derived from the angular radial transform, the histogram, the Gabor texture descriptor, the gray level co-occurrence matrices, and the power spectrum of the face region. Each of these features has several components as follows:

1. **Angular radial transform** [21] projects a face onto a set of orthogonal basis functions in 12 angular and 6 radial directions. (*71 features*)

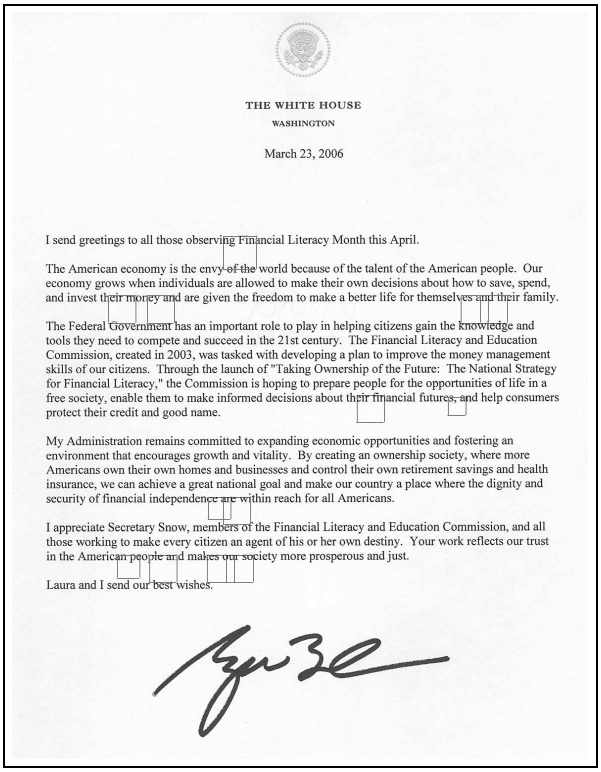
2. A normalized **histogram** of the pixel intensities of the face is used as features. The histogram is divided into 16 uniform-size bins. (*16 features*)

3. The **Gabor** features used are the mean and standard deviation of Gabor filtered images of 5 scales and 6 orientations [20]. (*60 features*)

4. The **gray level co-occurrence matrices (GLCM)** contain the number of pixels of intensity I_x which are a particular offset from a pixel of intensity I_y. Each matrix has a different offset and is normalized, with the entries summing to 1.



(a) One of the five regions is not a face.



(b) Non-faces in a letter are identified as faces.

Figure 2: Examples of non-faces detected as faces by the face detector

The GLCM is obtained using a *quantized face*, where the original face region has been quantized to 16 intensities. The features are taken from four offsets: (1,0), (1,1), (0,1) and (-1,1). The GLCM features used are the angular second moment, contrast, correlation, inverse difference moment, and entropy [12]. (*20 features*)

5. The **power spectrum** features [1] are the maximum, average, power, and variance of the Fourier transform amplitudes. An additional 16 values are obtained by block averaging the squared magnitude of the Fourier transform into 4 by 4 blocks. (*20 features*)

These low-level image features are combined to generate, $\hat{I}$, *187 image features* for a face in the feature vector.

4 Retrieval Process

Having extracted the text features from the caption and the image features from each face in the image, we create a feature vector for each data item (*i.e.* face) by concatenating the image and text features. Recall that when there is more than one face detected in an image, each of these faces will have the same text features as they have the same caption. Once the database is constructed, we normalize the feature vectors so that each feature is in the range $[0, 1]$.

4.1 Similarity Measure The retrieval process essentially takes the query and searches through the database for similar items. As a measurement of similarity between the feature vector representing the query and feature vectors representing the documents in the data set, we use a weighted cosine distance. This function computes the angle between the two weighted feature vectors:

$$\begin{aligned} d(x,y) &= \arccos(\cos(Wx, Wy)) \\ \cos(a,b) &= \frac{a \cdot b}{\|a\| \, \|b\|} \\ W &= \mathbf{diag}(w) \end{aligned}$$

where x and y are column vectors and W is a diagonal matrix of weights. Typically, the weight vector is normalized to sum to 1:

$$\sum_i w_i = 1.$$

In the retrieval process, the distances from the query document to all other documents in the database are obtained and the closest k documents to the query are returned. In the case of a tie, indicating two documents which are at equal distance from the query, we randomly choose the ordering of the documents. Note that this is different from the approach used in

`trec_eval` [4, 11], a common program used in information retrieval for evaluation, which uses the document number to determine the order of documents returned. We found that this biased the results towards documents with higher numbers and can significantly affect the results, especially when a large number of documents have the same distance to the query. This situation arises for text-only queries, when there are several faces with the same caption.

4.2 Partial Queries As a document is described by both an image and a caption, an ideal query has both image and text features. Often however, users can only provide either the image or the text query terms. We call these *partial queries* as the query is missing important features. A partial query which is missing image features will result in face and non-face images with the same caption having the same distance from the query. Similarly, a partial query with missing text features may return several faces of different people. This is because our low-level image features cannot adequately distinguish between faces, as discussed in Section 5.1.

To attain the performance of a full query given only a partial query, we present a query completion method to estimate values for the missing features. Query completion creates a sequence of completed queries from the original partial query. Each query in the sequence is updated using the retrieved documents from the previous iteration of retrieval. The query completion stops when the maximum number of iterations have been reached or the query has not significantly changed. We next describe two ways in which we accomplish the query completion.

Simple query completion method A simple approach to completing a partial query is based in the query refinement approach. This idea is based on the assumption that the user's query does not adequately express the user's true intent. By refining the query iteratively, we can improve the retrieval results [5, 24].

Consider the initial partial query to be $q = \langle a_q, ? \rangle$ where a is the set of features provided by the user and ? indicates that the query is incomplete with missing features. The first step of retrieval ignores the missing features and returns a set of documents $R(q)$:

$$\begin{aligned} R(q) &= \{r_1, \ldots, r_k\} \\ r_i &= \langle a_i, m_i \rangle \end{aligned}$$

where $k \leq n$ is the number of retrieved documents, r_i, which is typically less than the size of the database n. The m_i are the features in r_i that were missing from the query. In contrast to the query, each retrieved document has values for all features. The original query is then updated to form a new query $\hat{q} = \langle a_q, \hat{m}_q \rangle$ which consists of the original query values a_q and an estimate of the missing values $\hat{m}_q$. The estimate is a weighted average of the values for each of the retrieved documents $R(q)$:

$$\hat{m}_q = \frac{\sum_j \gamma_j m_j}{\sum_j \gamma_j}$$

where γ_j is a weight proportional to the similarity of each retrieved document r_j:

$$\begin{aligned} \gamma_j &= \delta_j e^{-\alpha \hat{d}(r_j, q)} \\ \delta_j &= \begin{cases} 1+\beta & j \leq k^\star \\ 1 & j > k^\star \end{cases} \end{aligned}$$

As the top $k^\star$ documents tend to be the most relevant, we increase their weight by $\delta \in [1, 2]$ such that $0 < \beta < 1$. The parameter $\alpha \geq 1$ weights the distance values such that even the least relevant documents have weight > 0.001. We used parameters $\alpha = 2$, $\beta = 0.1$, and $k^\star = 100$ for our experiments, which we determined empirically. The distance function $\hat{d}$ is scaled to be in $[0, 1]$:

$$\begin{aligned} \hat{d}(r, q) &= \frac{d(r, q) - \min_r d(r, q)}{\max_r d(r, q) - \min_r d(r, q)} \\ d(r, q) &= \begin{cases} d\left(\langle a_r \rangle, \langle a_q \rangle\right) & \text{if } q = \langle a_q, ? \rangle \\ d\left(\langle a_r, m_r \rangle, \langle a_q, \hat{m}_q \rangle\right) & \text{otherwise} \end{cases} \end{aligned}$$

where d is the attribute-level distance of the component values. This approach to estimating the missing features ensures that more similar documents contribute more to the new query.

The query is repeatedly updated using the documents retrieved in response to the previous query. The final query is determined by the maximum number of iterations or the convergence of the query. The query converges if the distance between the previous query and the current query is less than $\epsilon = 0.001$.

Query completion with relevance feedback In the simple query completion, we assume that the first few documents are the most relevant ones. However, we know that this is often not the case. In an interactive retrieval environment, a user can improve the retrieval results by specifying which of the returned documents are the more relevant ones. We can exploit this relevance feedback [10, 23, 29] and assist the query update by determining which documents to use in estimating the missing values. To compute the updated query, we use a subset of the retrieved results $R'(q) \subseteq R(q)$ from the previous query q. The subset is defined as follows:

$$R'(q) = \{r \mid r \in R(q) \wedge cl(r) = cl(q)\}$$

where $cl(x)$ is the class label of a document x and $R(q)$ is the set of documents retrieved for a query q. Each document has a class label which we can use to mimic user relevance feedback. The weight for each document in $R(q)$ depends on the class label:

$$\gamma_j = \delta_j e^{-\alpha \hat{d}(r_j,q)}$$
$$\delta_j = \begin{cases} 1 & r_j \in R'(q) \wedge j \leq k^\star \\ 0 & \text{otherwise} \end{cases}$$

where $k^\star = 100$ is the maximum number of documents to consider in estimating the missing features. By limiting the number of documents we essentially simulate a user's attention span as we can expect that a user is unlikely to provide feedback on more than 100 documents. Here, the updated query, $\hat{q}$, replaces the missing values and updates the original query values a_q as well, resulting in the new query: $\hat{q} =< \hat{a_q}, \hat{m_q} >$. Note that unlike the simple query completion, the relevance feedback not only allows us to estimate the missing features using more relevant documents, but also allows us to change the "non-missing" features by updating them. This query modification is another common approach to relevance feedback [29]. In the special case that $R' = \emptyset$, we revert to the simple completion method without relevance feedback.

5 Experiments

We conducted our experiments in estimating missing feature values using a data set of documents which comprised images and their associated captions. The documents in the data set were obtained as the result of several text queries using the Google Images Agent [27]. As we were interested in faces, many of the text queries were names of people in the news. Given the text query, an image was relevant if it was near a concentration of query terms in the HTML source code of the web page. The caption for the image was the text near the image in the HTML source. Although a web page could contain several images, the image search returned only the image that was near a high concentration of query terms.

The data set was then reduced based on several criteria. Any word in the caption that occurred in at least 3 documents was included in the feature vector. This reduced the number of text features from 37,000 to 619. Some documents had no text features. Most of these were errors such as "This page contains characters that cannot be displayed." or captions in another language. All documents with no words in the feature vector were removed. We also removed all documents whose class had fewer than 100 documents. The resulting data set consists of 5910 documents. These documents were manually labeled for analysis. This allowed us to evaluate the quality of the retrieval results and also enabled us to mimic a user's input for relevance feedback.

Label	Google Query (Name)	# of Documents
A	Non Faces	2497
B	Unknown Faces	1366
C	George H. W. Bush	162
D	President Bush (George W. Bush)	219
E	Jeb Bush	140
F	William Jefferson Clinton	211
G	Hillary Rodham Clinton	286
H	Bill Gates	219
I	Aaliyah	227
J	Ben Affleck	182
K	Andre Agassi	187
L	Christina Aguilera	214

Table 1: Labels and distribution of documents for the data set.

The ten text queries used to generate the data set and the distribution of the documents with class labels are shown in Table 1. Approximately 42% of the data are non-faces, 23% are unknown faces and the remaining 35% are divided among ten known faces. The non-faces correspond to errors in the face detector. The face detector requires a threshold to distinguish faces from non-faces. We chose a low threshold to ensure a large number of faces were detected. This lowered the precision of the face detector, increasing the number of non-faces detected.

5.1 General Observations on the Data First, to gain insight into the contributions of the text features and image features, we used the k-nearest neighbor classifier, where $k = 1$ and examined the confusion matrices. Tables 2, 3, and 4 show the confusion matrices obtained using text-only features, image-only features, and combined image and text features, respectively. The elements along the diagonal of the matrix are those correctly identified as belonging to a certain class. The false positives for a class are the documents assigned to that class but are actually from another class. These are indicated along a column of the confusion matrix (values not in bold). The false negatives for a class are the values along a row (excluding the diagonal) and indicate those documents have been assigned to the wrong class.

For example, in Table 2, of all the non-face documents (row **A**), 1,836 were correctly classified (column

	Classifier Assigned											
	A	B	C	D	E	F	G	H	I	J	K	L
A	**1836**	187	20	61	10	20	99	21	74	65	44	60
B	453	**503**	17	53	12	21	105	22	28	65	59	28
C	105	20	**12**	14	1	8	1	0	0	0	0	1
D	80	37	4	**90**	2	3	0	0	1	1	0	1
E	81	50	1	3	**5**	0	0	0	0	0	0	0
F	79	76	9	0	2	**28**	13	1	2	0	1	0
G	51	44	1	2	1	15	**170**	0	1	0	1	0
H	152	29	0	0	0	1	0	**34**	0	1	1	1
I	24	13	0	2	1	1	1	1	**179**	3	0	2
J	19	33	0	0	0	1	0	0	4	**123**	1	1
K	39	24	0	1	0	1	2	1	0	1	**118**	0
L	27	7	1	3	0	0	0	1	2	0	0	**173**

Table 2: Confusion matrix for text-only classification. Each entry is the number of documents known to be from the class in the row that were assigned to the class in the column. The entries in bold are the number of documents assigned to the corrected label.

A) but 99 were classified as Hillary Clinton (column *G*). Of all documents that the classifier labeled as Hillary Clinton (column *G*), 170 were really Hillary Clinton (row **G**) while 99 were non-faces (row **A**). Further, of all the documents that were really Hillary Clinton, 51 were identified as a non-face by the classifier.

Note that while the 1-nearest-neighbor classifier is somewhat simplistic, it gives us an insight into the effectiveness of the different features.

For example, using text features only, we know that documents with identical captions (after stemming, *etc.*) are considered equivalent. In our data set, it is common for a caption to have many of the correct words but be a non-face. For example, a letter written by President Bush and a photo of him may have the same caption, but only the latter is a face. We would expect based on just the caption, many of the non-face documents would be misclassified as a face and *vice versa.* Table 2 shows that the non-faces are spread across the first row of the matrix. The false positive rate of the non-face class (column *A*) is high. About 20% of all face documents were wrongly labeled as non-faces. This means that the classifier cannot distinguish faces from non-faces on the basis of text alone.

In the known face classes, only 5 documents of Jeb Bush (row **E**) and 12 of his father, George H. W. Bush (row **C**) were correctly classified. The remainder were split among the non-faces (column *A*) and unknown face (column *B*) classes. The reason for these low numbers is the large number of non-face documents with similar captions. Also, a large number of unknown faces in images which contain Jeb Bush and George H. W. Bush

	Classifier Assigned											
	A	B	C	D	E	F	G	H	I	J	K	L
A	**2026**	211	17	24	25	29	24	23	34	36	22	26
B	84	**661**	75	85	43	68	104	66	43	47	54	36
C	11	45	**44**	10	8	7	14	7	4	3	6	3
D	7	65	14	**65**	11	16	10	10	2	9	6	4
E	5	39	10	20	**23**	8	7	2	5	13	6	2
F	5	70	11	17	7	**51**	20	10	5	8	5	2
G	3	87	10	22	9	21	**87**	10	5	8	5	19
H	4	46	8	7	7	14	16	**99**	5	5	2	6
I	11	49	3	3	4	3	17	1	**99**	15	5	17
J	13	44	5	12	10	8	8	7	4	**61**	5	5
K	2	58	11	20	6	2	14	4	3	8	**54**	5
L	9	69	5	12	5	4	19	5	16	8	9	**53**

Table 3: Confusion matrix for image-only classification. Each entry is the number of documents known to be from the class in the row that were assigned to the class in the column. The entries in bold are the number of documents assigned to the corrected label.

	Classifier Assigned											
	A	B	C	D	E	F	G	H	I	J	K	L
A	**1936**	233	41	45	22	29	39	37	45	29	15	26
B	91	**737**	62	83	67	59	79	41	27	57	45	18
C	10	66	**57**	17	3	9	0	0	0	0	0	0
D	17	75	20	**93**	9	3	0	1	0	1	0	0
E	9	64	4	9	**49**	2	1	0	0	0	1	1
F	15	67	10	7	4	**82**	16	6	1	1	1	1
G	20	73	0	4	0	24	**164**	0	0	0	1	0
H	17	36	0	2	0	1	1	**161**	1	0	0	0
I	9	17	0	0	0	0	0	0	**198**	0	1	2
J	6	50	0	0	0	0	0	1	3	**122**	0	0
K	8	41	1	2	0	1	0	0	1	0	**133**	0
L	9	18	2	1	0	0	1	1	1	2	0	**179**

Table 4: Confusion matrix for image-text classification. Each entry is the number of documents known to be from the class in the row that were assigned to the class in the column. The entries in bold are the number of documents assigned to the corrected label.

are identified incorrectly as them because they have the same caption.

We also observe that several classes have very good performance with text features. These documents correspond to people with unique names (rows **I**-**L**) and are commonly in images by themselves, with no unknown faces.

Table 3 shows that image features can definitely distinguish non-faces (column A) from known or unknown face documents. But, the image features seem unable to distinguish different faces. Few face documents were classified as non-face (column A) using image features. Many face documents were classified as unknown faces (column B). In several classes, more were classified as an unknown face (column B) than the correct class (rows **C**, **E**, **F**, **K**, and **L**).

We deliberately chose our dataset so that there is more than one person with the same last name, for example, Bush (rows **C**, **D**, and **E**) and Clinton (rows **F** and **G**) and also the same first name, as in Bill (rows **F** and **H**). For some of these classes that share names, we observe that image-only features improved performance over text-only features. The main problem with text features is that they cannot distinguish non-faces from faces, while image features can do this easily.

We therefore expect that combining the image and text features would give us the best of both worlds and the results in Table 4 support this claim. It indicates that the performance of text features is improved when image features are added. For those classes with names in common (rows **C**-**H**), the addition of image features generally improves the overall performance. In particular, Jeb Bush (row **E**) and George H. W. Bush (row **C**) have a substantial improvement. With text-only features, both were often confused as non-faces (column A) or unknown faces (column B).

For people with unique names (rows **I**-**L**), text-only features do quite well and the addition of low-level image features aids in distinguishing faces from non-faces. Table 4 shows that for such people, the combined performance exceeds that obtained using text-only features and is a dramatic improvement over image-only features.

When we consider the overall performance (Table 5), we find that combining the image and text features improves the accuracy by about 10%. It is clear that combining image and text features is helpful. The text-only case benefits as the image features distinguish faces from non-faces and the image-only case benefits as the text features help in distinguishing names.

5.2 Retrieval Performance We conducted our experiments on retrieval by selecting 5 queries from each of the ten known faces, resulting in 50 queries. The queries were selected to represent how a user might use an information retrieval system. Each query has a different but descriptive caption for the person. The captions contain the name of the individual or title. For example, for George H. W. Bush, example captions are "Former President Bush" and "George Herbert Walker Bush".

Features	% Accuracy
Text	55.34%
Image	56.2267%
Image and Text	66.176%

Table 5: Accuracy of the 1-NN classifier using image, text, and both image and text features.

5.3 Evaluation Method In text retrieval, precision-recall curves are often used to evaluate methods [13, 23, 24]. Precision is the ratio of the number of relevant documents retrieved to the number of retrieved documents. Recall is the ratio of the number of relevant documents retrieved to the total number of relevant documents in the database. When the number of retrieved documents is much smaller than the number of relevant documents, which is usually the case in large databases, the recall will always be low. An alternative evaluation method used in such cases is the precision-scope curve [14, 28] available in `trec_eval` [4, 11], where the scope is the number of retrieved documents. We use the precision-scope curve to evaluate our retrieval algorithms.

5.4 Full and partial query evaluation We examined eight scenarios for retrieving documents

1. A partial query with image-only features.
2. A partial query with text-only features.
3. A partial query with text-only features, completed using the simple method.
4. A partial query with image-only features, completed using the simple method.
5. A partial query with text-only features, completed using relevance feedback.
6. A partial query with image-only features, completed using relevance feedback.
7. A full query with both text and image features.
8. A full query with both text and image features, refined using relevance feedback.

We used 20 iterations of query completion without relevance feedback or until the queries converged. For query completion with relevance feedback, we used only 2 iterations as we do not expect the users to provide feedback on a large number of documents. The completion method converges if the difference between the current query and the previous query is less than $\epsilon = 0.001$.

In Figure 3, we compare the results for the text-only query with those obtained for the text-only partial query with simple query completion. For comparison, we have also included the results for a full query with both image and text features. The figure indicates that the text-only query has the poorest results when less than 90 documents are retrieved. The precision is relatively flat at 40%. This is due to the large number of documents with equal distance to the query, specifically, documents with the same caption but different faces. In the returned results, the order of retrieved documents with the same distance to the query is randomly chosen. To improve results, we need to break the ties among documents with identical captions. This is accomplished using the image features. The text-only partial query with estimated image features significantly improves the performance over the text-only query when less than 90 documents are retrieved. In fact, these results almost match the full query results. This is because the addition of image features enabled ties between faces and non-faces to be broken, thus improving the retrieval results.

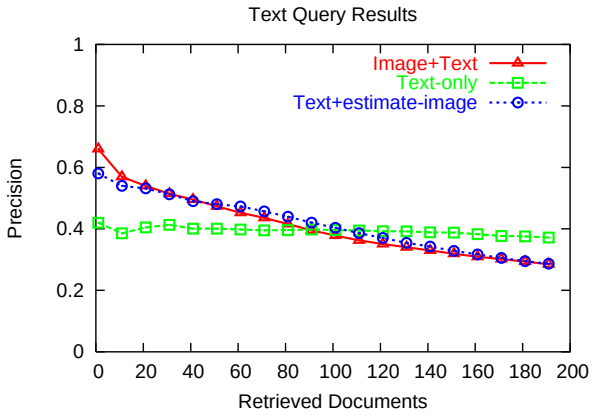


Figure 3: Precision-scope curves of text-only query, text query with image features estimated, and full query.

We expect that additional improvements can be made by including information from the relevance feedback in the completion of the partial text-only query and in the refinement of the full query with text and image features. Figure 4 shows these results when compared with the full query with no relevance feedback. We observe that the performance of the partial-query completion method with relevance feedback is similar to the full query with relevance feedback. This indicates that we do not lose by estimating the image features.

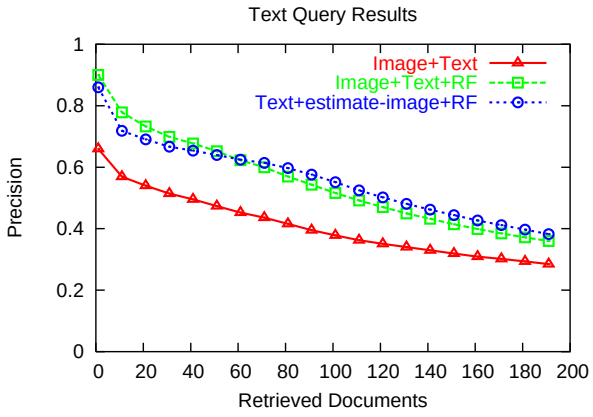


Figure 4: Precision-scope curves of full query, text query with image features estimated using relevance feedback and full query with relevance feedback.

We next consider a similar set of experiments conducted with image-only queries. Figure 5 indicates that the image-only partial query has a precision below 20%. As we expect, the low-level image features are poor at distinguishing one person from another. The addition of estimated text features using the simple query-completion method improves the results only slightly. The improvement is small because a large number of incorrect documents are combined to form the estimated text features. Essentially, the simple query completion assumes that the first few documents are the most relevant, which is not the case as our image features have poor retrieval result. When the text features are known,

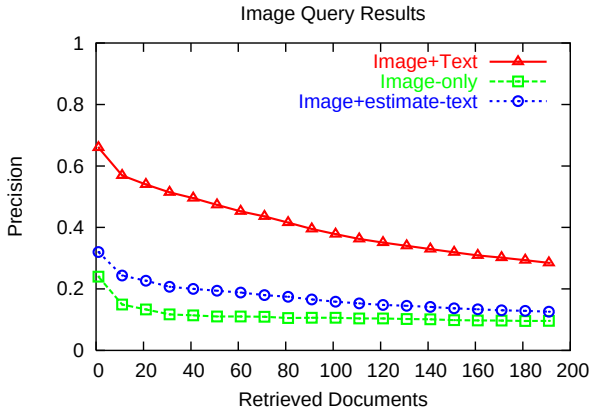


Figure 5: Precision-scope curves of image-only query, image query with text estimated, and full query

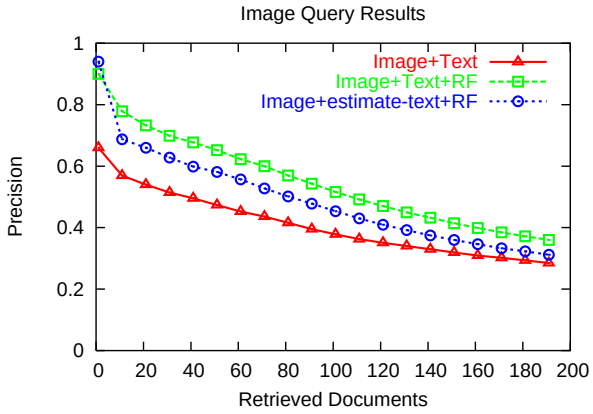


Figure 6: Precision-scope curves of full query, image query with text estimated using relevance feedback, and full query with relevance feedback.

as in the case of the full query, the results improve significantly. It is obvious that completing the image-only query using relevance feedback will improve the retrieval results by using only relevant documents to complete the query. These results of the image-only query with estimated text using relevance feedback are shown in Figure 6. The completed query improves performance beyond the full query results and approaches the performance of the full query with relevance feedback.

5.5 Other experiments We conducted additional experiments to attempt to enhance the retrieval performance and learn more about the relationship between image and text features. This section presents three topics we investigated: weight learning, word pairs, and automatic annotation.

- **Weight Learning**

  It is obvious that compared to either image or text features alone, combining the features increases the dimensionality of the feature vector. In this high-dimensional space, a distance function cannot adequately distinguish documents of different classes, especially if some of the features are irrelevant. A solution to this problem is to learn weights for the features, so that more relevant features can be assigned higher weights.

  We considered two approaches to learning the weights: the Relief-F algorithm which weights a feature based on its ability to distinguish between different classes [16] and the weights obtained through relevance feedback based on the standard deviation of the features in the relevant documents [26]. We did not observe any substantial improvement in performance using weights on the features.

- **Word Pairs**

  We also considered word pairs as an option to improve precision. Since full names commonly consist of at least two words, word pairs add additional constraints, helping to resolve conflicts, and leading to better performance. For example, the word "bill" is shared by two classes, but "bill clinton" is present in only one class.

  To incorporate word pairs into the text features, we extracted all word pairs that occur in the image caption and appended them to the single words in the document vector. Our results showed that addition of word pairs have no significant difference in the precision results.

- **Automatic Annotation**

  As a component of our query completion method, given an image, we estimate text features. In automatic annotation, the relationship between image and text is learned and for a new image, text is found to represent the image. Examining the text features that are estimated by our query completion method, we found the query completion without relevance feedback is limited by its assumption that the first few documents are relevant. But with relevance feedback, the query completion method appears promising as an automatic annotation method.

  Using relevance feedback, the annotation yields some surprising results. Sometimes, several new words appear as a result of query completion. Notably, we observed that "war" is added to George

W. Bush and "Florida" is added to Jeb Bush. The annotation also appears to discover that William Jefferson Clinton is often referred to as Bill Clinton. Although surprising at first, the words appear only because the documents that contain them were labeled as relevant after the initial retrieval results. This may allow us to connect documents which may originally appear to be unconnected.

6 Conclusions

In this paper, we investigated a query completion method which enabled us to generate a query containing both image and text features, given a text-only or an image-only query. Our approach was to iteratively estimate the values for the missing features using a weighted average of the features from the retrieved documents.

We evaluated our approach using a database of nearly 6,000 documents, each an image of a face with an associated caption. Our results showed that text-only queries performed better than image-only queries, which was expected as we were using low-level image features. Both image-only and text-only queries were improved by estimating the missing features. In fact, text-only queries with estimated image features performed as well as full queries with both features. The use of relevance feedback to improve the estimate of missing features led to further gains, exceeding the performance of a full query with text and image features.

We also observed that the text features performed well in distinguishing individuals with unique names but could not distinguish faces from non-faces. The low-level image features chosen distinguished faces from non-faces but was not able to recognize individual faces. Thus, combining the two sets of features allowed us to exploit the strengths of each. Our query completion method allowed us to estimate the missing features, ensuring improved queries with better performance.

We plan to extend this work in several ways. First, we will investigate further the possibility of using our technique for automated image annotation. Next, we will extend our analysis to retrieval queries other than faces. We are especially interested in seeing how our low-level image features, which are very general, work when the query is a general query. Finally, as we have observed, the image features cannot adequately distinguish between different faces. For face queries, this limits the utility of the image retrieval, the potential for improvement with query completion, and the application of automatic annotation. We will explore alternative image features which are more suitable for retrieval of faces.

7 Acknowledgments

We would like to thank Dale Slone for the scripts to obtain the images and associated captions and Krystian Mikolajczyk for the face detector software. We also acknowledge the contributions of the Sapphire team who developed the software used in this work.

Abraham Bagherjeiran is a graduate student at the University of Houston, this work was done while he was a summer intern at LLNL.

UCRL-CONF-225087: This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

References

[1] M. Augusteijn, L. Clemens, and K. Shaw, *Performance evaluation of texture measures for ground cover identification in satellite image by means of neural network classifier*, IEEE Transactions on Geoscience and Remote Sensing, 33 (1995), pp. 616–626.

[2] K. Barnard and D. Forsyth, *Learning the semantics of words and pictures*, in Proc. 8th Int'l Conf. on Computer Vision, vol. 02, Los Alamitos, CA, USA, 2001, IEEE Computer Society, pp. 408–415.

[3] T. L. Berg, A. C. Berg, J. Edwards, M. Maire, R. White, Y.-W. Teh, E. Learned-Miller, and D. A. Forsyth, *Names and faces in the news*, in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, vol. 2, 2004, pp. 848–854.

[4] C. Buckley, *trec_eval information retrieval evaluation package*, July 2006. Available from `http://trec.nist.gov/trec_eval/`.

[5] C. Buckley, G. Salton, J. Allan, and A. Singhal, *Automatic query expansion using SMART: TREC 3*, in Proc. 3rd Conf. Text REtrieval, November 1995, pp. 69–80.

[6] W. B. Croft, *Combining approaches to information retrieval*, in Adv. in Information Retrieval, Kluwer Academic Publishers, 2000, ch. 1, pp. 1–36.

[7] T. Deselaers, D. Keysers, and H. Ney, *Fire: Flexible image retrieval engine: Imageclef 2004 evaluation*, Lecture Notes in Computer Science, 3491 (2005), pp. 688–698.

[8] P. Duygulu, K. Barnard, J. F. G. de Freitas, and D. A. Forsyth, *Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary*, in Proc. 7th European Conf. on Computer Vision, vol. 2353, Copenhagen, Denmark, May 2002, pp. 97–111.

[9] J. Edwards, R. White, and D. Forsyth, *Words and pictures in the news*, in Proc. HLT-NAACL 2003 Workshop on Learning word meaning from non-linguistic data, Morristown, NJ, USA, 2004, Association for Computational Linguistics, pp. 6–13.

[10] J. Favela and V. Meza, *Image-retrieval agent: Integrating image content and text*, Intelligent Systems and Their Applications, 14 (1999), pp. 36–39.

[11] D. Grossman, *Notes on trec eval*, July 2006. `http://ir.iit.edu/~dagr/cs529/files/project_files/trec_eval_desc.htm`.

[12] R. Haralick, K. Shanmugam, and I. Dinstein, *Textural features for image classification*, IEEE Transactions on Systems, Man, and Cybernetics, SMC-3 (1973), pp. 610–621.

[13] J. Jeon, V. Lavrenko, and R. Manmatha, *Automatic image annotation and retrieval using cross-media relevance models*, in Proc. 26th Int'l Conf. on Research and Development in Information retrieval, New York, NY, USA, 2003, ACM Press, pp. 119–126.

[14] F. Jing, B. Zhang, F. Kin, W.-Y. Ma, and H.-J. Zhang, *A novel region-based image retrieval method using relevance feedback*, in Proc. of the 2001 ACM workshops on Multimedia: multimedia information retrieval, New York, NY, USA, 2001, ACM Press, pp. 28–31.

[15] G. Karypis, *Cluto: A clustering toolkit.*, Tech. Rep. Technical Report 02-017, University of Minnesota, Department of Computer Science, November 2003. `http://glaros.dtc.umn.edu/gkhome/cluto/cluto/download`.

[16] I. Kononenko, *Estimating attributes: Analysis and extensions of RELIEF*, in Proc. European Conf. on Machine Learning, Secaucus, NJ, USA, 1994, Springer-Verlag New York, Inc., pp. 171–182.

[17] V. Lavrenko, R. Manmatha, and J. Jeon, *A model for learning the semantics of pictures*, in Adv. in Neural Information Processing Systems 16, S. Thrun, L. Saul, and B. Schölkopf, eds., MIT Press, Cambridge, MA, 2004.

[18] J. Li and J. Z. Wang, *Automatic linguistic indexing of pictures by a statistical modeling approach*, IEEE Trans. on Pattern Analysis and Machine Intelligence, 25 (2003), pp. 1075–1088.

[19] W.-C. Lin, Y.-C. Chang, and H.-H. Chen, *From text to image: Generating visual query for image retrieval*, Lecture Notes in Computer Science, 3491 (2005), pp. 664–675.

[20] B. Manjunath and W. Ma, *Texture features for browsing and retrieval of image data*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 18 (1996), pp. 837–842.

[21] B. S. Manjunath, *Introduction to MPEG-7: Multimedia Content Description Interface*, Wiley, 2002.

[22] K. Mikolajczyk, C. Schmid, and A. Zisserman, *Human detection based on a probabilistic assembly of robust part detectors*, European Conference on Computer Vision, 1 (2004), pp. 69–82. `http://www.robots.ox.ac.uk/~vgg/research/affine/face_detectors.html`.

[23] M. Ortega, K. Porkaew, and S. Mehrotra, *Information retrieval over multimedia documents*, Tech. Rep. Technical Report, University of California at Irvine, 1999.

[24] K. Porkaew and K. Chakrabarti, *Query refinement for multimedia similarity retrieval in mars*, in Proc. 7th ACM Int'l Conf. on Multimedia, New York, NY, USA, 1999, ACM Press, pp. 235–238.

[25] M. Porter, *An algorithm for suffix stripping*, Program, 13 (1980), pp. 130–137.

[26] J. Rocchio, Jr., *Relevance feedback in information retrieval*, in The SMART Retrieval System: Experiments in Automatic Document Processing, G. Salton, ed., Prentice Hall, 1971, ch. 14, pp. 313–325.

[27] G. Rousse, *Google images agent*, 2005. Perl Script `http://search.cpan.org/~grousse/WWW-Google-Images-0.6.3`.

[28] Y. Rui and T. S. Huang, *Optimizing learning in image retrieval*, in Proc. of IEEE Conf. on Computer Vision and Pattern Recognition, 2000, pp. 236–243.

[29] Y. Rui, T. S. Huang, and S. Mehrotra, *Relevance feedback techniques in interactive content-based image retrieval*, in Storage and Retrieval for Image and Video Databases (SPIE), 1998, pp. 25–36.

[30] G. Salton and M. J. M. Gill, *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983.