Efficient Semantic Indexing for Image Retrieval

Chandrika Pulla, Suman Karthik, C. V. Jawahar
CVIT, International Institute of Information Technology
Hyderabad-500 032, INDIA

Abstract

Semantic analysis of a document collection can be viewed as an unsupervised clustering of the constituent words and documents around hidden or latent concepts. This has shown to improve the performance of visual bag of words in image retrieval. However, the enhancement in performance depends heavily on the right choice of number of semantic concepts. Most of the semantic indexing schemes are also computationally costly. In this paper, we propose a bipartite graph model (BGM) for image retrieval. BGM is a scalable datastructure that aids semantic indexing in an efficient manner. It can also be incrementally updated. BGM uses ***tf-idf*** *values for building a semantic bipartite graph. We also introduce a graph partitioning algorithm that works on the BGM to retrieve semantically relevant images from a database. We demonstrate the properties as well as performance of our semantic indexing scheme through a series of experiments.*

1. Introduction

In content based image retrieval systems, similar images are found by matching features of the query image with those in the database images. Most of the current image retrieval systems use visual bag of words model for efficient and effective retrieval [13]. Bag of words model encodes the similarity between descriptors by quantization of a very high dimensional space (using algorithms like k-means) to build a compact codebook. Images are then represented with the help of a set of visual words (members of the clusters). This approach, thus, relies only on the (quantized) low level features of the image. Thus it has an inherent problem of semantic gap.

To futher improve the retrieval performance, semantic indexing techniques like Latent Semantic Analysis (LSA), Probabilistic Latent Semantic Analysis (pLSA) [4] and Latent Dirichlet Allocation (LDA) [1] were designed. Semantic analysis can be viewed as an unsupervised clustering of the constituent words and documents around hidden or latent concepts. Semantic indexing techniques have been extended to various computer vision problems in the past [2, 7, 10, 12]. Both LSA and pLSA were originally developed for text retrieval [4] and later extended for visual data [2]. In LSA, the term document matrix, which encodes the frequency of the words in the document, is decomposed using singular value decomposition. The largest k Eigen values from the decomposed matrices are used to form a reduced matrix, which defines the dimensionality of the latent space.

pLSA is a generative model of the data with strong statistical foundation, where each document is represented by its word frequency. And the similarity between the documents is compared in a semantic space which is more reliable than original representation. Unfortunately, the performance depends on the number of semantic concepts one assumes. (See Figure 3). For most practical databases (like internet image collections), guessing the number of concepts is practically impossible. Also pLSA is computationally intensive. Most of the current databases are large in size and therefore the semantic indexing for such large database is not practically feasible.

Semantic indexing for dynamic databases where new images are constantly added to the image collection poses a considerable challenge, primarily due to its resource intensive matrix computations. An incremental variant of pLSA proposed by Wu *et al.* [14] tried to improve the computational efficiency. However they failed to address the issue of storage complexity. Even their performance depends on the right choice of number of concepts. (See the comparative study in Section 3).

In this paper, we propose a Bipartite Graph Model (BGM) for semantic indexing. BGM converts the vector space model into a bipartite graph which can be incrementally updated with *just in time* semantic indexing. BGM have been used successfully in unsupervised learning tasks like data clustering [15]. We further propose a graph partitioning scheme that traverses the BGM to retrieve relevant images at runtime. We compare the retrieval performance of BGM, pLSA and Incremental pLSA using the holiday dataset [5]. Our method is computationally efficient, and robust to the parameters of the model.

2. Bipartite Graph Model

Our *Bipartite Graph Model* (BGM) indexes the term document data in a scalable and incremental manner. The basic idea of bipartite graph model is to convert the term document matrix into a bipartite graph of terms and documents (See Figure 1). In BGM, the edges are weighted with term frequencies of words in the documents and each term is also associated with an inverse document frequency value. These values determine the importance of a word to a particular document. $G = (W, D, E)$ is the bipartite graph such that $W = \{w_1, w_2 \ldots, w_n\}$, $D = \{d_1, d_2 \ldots, d_m\}$ and $E = \{e^{d_1}_{w_1}, e^{d_2}_{w_7} \ldots, e^{d_m}_{w_n}\}$. Here the weight associated with $w_1 = IDF(w_1)$ and that of $e^{d_1}_{w_1} = TF(w_1, d_1)$. Thus the BGM encodes the co-occurrence data in the term document matrix without the need to project the database into a latent topic space.

As shown in Figure 1, the documents (images) are connected to words (quantized neighbourhood descriptors). An image may contain many words. A word may be present in many images. Similarity of two images can be measured in terms of the number of words they share. If images A and B as well as A and C are similar, then B and C are also similar. This gets reflected in the paths which traverse the graph from image to word and then back to images.

2.1 A Graph Partition Scheme

A vertex partitioning of $G = (W, D, E)$ denoted by (V_1, V_2) is defined as a partition of the vertex set D, such that vertex set V_1 contains vertices which are relevant to the query, and V_2 contains all other nodes. Our method is fundamentally a damped label propagation, which is a modification of the method suggested by Raghavan *et al.* in [9] (and also [6]). Our graph partitioning algorithm adapts their method by performing a single source label propagation, instead of multi-node propagation. This gives us the flexibility to gauge the label propagation through each node. When a query is given, the query node attaches itself to the nodes in the set W which are directly related to the query, with the relationship previously known. The node initially contains a fixed number of labels, which are partitionable. The node then distributes the labels based on the edge weight between the node and its neighbours, such that the received amount of labels is directly proportional to the edge weight. The query node is disconnected from the graph. The neighbours then propagate the labels to their neighbours. If the node is a document node, the distribution of the labels among its edges is determined according to the quantity which is proportional to the flow capacity calculated by the normalized Term Frequency (TF) value. If the node is a word node, then a penalty, which is proportional to the Inverse document Frequency(IDF) value of the word, is taken from the amount of label it receives and the rest is distributed like the document node based on the flow capacity of its edges. Hence higher the edge weights the more label is propagated to the relevant node. At each node the label is compared with a *cutoff* value which is the least amount of the label needed for a node to forward the label. Hence the label is propagated to relevant documents and terms until a cutoff value is reached at which point label is no longer propagated. The nodes receiving the most label are the most relevant documents. Thus, it divides the nodes in the bipartite graph into relevant and non-relevant sets similar to a graph cut algorithm.

Algorithm 1 Graph Partitioning Algorithm for Bipartite Graph

```
def GP($G, N, labels$)
Update amount of labels that have passed through node $N$
Label[$N$] += $labels$
if Node $N$ is of type Word then
    $labels$ = $labels$ * IDF($N$)
end if
if Amount of labels transferable from N $<$ cutoff then
    exit
end if
for each $node$ in neighbourhood of $N$ do
    GP($G, node, labels * TF(N, node)$)
end for
```

A new document can be inserted in a Bipartite Graph Model by creating a new document node and creating edges to the relevant words based on their term frequency (TF) values and updating the IDF values of the relevant word nodes. The complexity of insertions and deletions of documents is linear to the number of words within a document.

To summarize, most of the existing techniques like pLSA generally categorize the entities in a datasets into multiple groups and interaction between them are stored in a matrix. The values in the matrices represent the strength of interaction between them and elements in

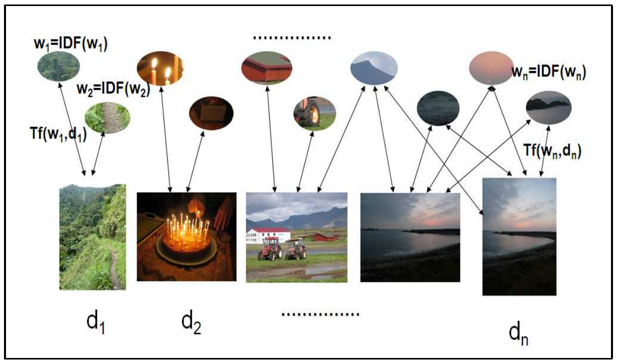

Figure 1. Graphical representation of Bipartite Graph Model. The image in the database is represented as a collection of visual words. The edges connect the visual words to the images in which they are present.

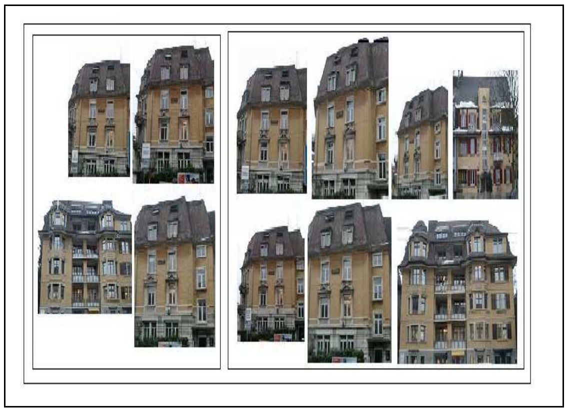

Figure 2. The result of retrieval on Zurich building data for simple indexing and BGM, first image is query image.

the same category are considered independent of each other. As the data size increases and interactions become sparser and we need to retrain the pLSA model when ever new data come, which is computationally expensive and time consuming. A natural progression of the method is to represent the interactions as graphs. The normalized strength of interaction between two entities being the weight of the edge connecting the two.

3. Experiments

We first present the retrieval performance of BGM, and compare it with a direct retrieval without any semantic indexing. For this, we use Zurich Building Image Database [11] consisting of 1005 images of 201 buildings. We extracted SIFT vectors from the images and quantize the feature space using Kmeans with a vocabulary size of 1000. Then we build a simple indexing scheme, where the similarity between documents is compared using cosine metric between the documents (vectors) from the term document matrix. BGM is constructed as explained in section 2. The Mean Average Precision(mAP) retrieval performance for simple retrieval is 0.26, whereas for BGM it is 0.54 mAP. As can be seen from Figure 2, BGM is able to retrieve images that simple retrieval can not.

We now, compare the retrieval performance of pLSA with the retrieval performance of BGM. For this experiment we have used holiday dataset[5], it contains 500 image groups, each representing a different scene or object. The first image of each group is the query image and the correct retrieval is the other images of the same group, in total the dataset contains 1491 images. We made extensive use of local detectors like Laplacian of Gaussian(log) and the SIFT descriptors[3]. Initially all the images from the dataset were downsampled to reduce number of interest points, after which feature detection and SIFT feature extraction was done. Once the features were extracted the cumulative feature space was vector quantized using K-means. With the aid of this quantization the images were converted into documents or collection of visual words.
For pLSA, we first construct a term document matrix A of the order $M \times N$ where M is the vocabulary size and N is the number of documents. Here, each image is represented as a histogram of visual words. An unobservable latent topic Z_k is introduced between the documents and the words. Thus $P(w_i, d_j) = P(d_i) \sum_k P(z_k|d_j) P(w_j|z_k)$. We learn the unobservable probability distribution $P(z_k|d_j)$ and $P(w_i|z_k)$ from the data using the Expectation Maximization Algorithm. For retrieval the Euclidean distance of the documents over topic probabilities was used to retrieve the 10 most similar images.

For BGM term document matrix was constructed and normalized. Then all the terms in the matrix were updated with their inverse document frequency values. This term-document matrix was then converted into a bipartite graph between the set of terms and documents as described by the BGM model. For each of the 500 query images the graph partitioning algorithm was used over this graph to retrieve the 10 most similar images.

Retrieval results for the both BGM and pLSA were aggregated and the evaluation code provided for the holiday dataset was used to calculate the Mean Average Precision(mAP) in both cases in Table 1.

We now demonstrate the retrieval performance of

Model	mAP	time	space
Probabilistic LSA	0.642	547s	3267Mb
Incremental PLSA	0.567	56s	3356Mb
BGM	0.594	42s	57Mb

Table 1. Mean Average Precision for both BGM, pLSA and IpLSA for the holiday dataset, along with time taken to perform semantic indexing and memory space used during indexing.

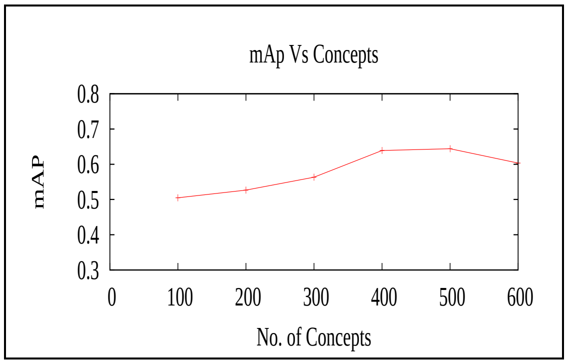

Figure 3. The retrieval performance of PLSA varying the number of Concepts.

pLSA with respect to number of concepts. For this we used Holiday database [5]. As we can see from the figure 3 the retrieval performance is minimal if there is mismatch between the concepts assumed for training and the actual concepts in the database.

Typical image retrieval systems are generally built on static databases whereas, in real world the data keep changing i.e., the images are added or removed frequently. pLSA cannot handle streaming/constantly changing data as the model has to be retrained on both new and old data which is computationally expensive. To handle this Incremental pLSA [14] was proposed in which when ever a new image is added, the probability of a latent topic given the document $P(z|d)$ and the probability of words given topic $P(w|z)$ are updated based on Generalized Expectation Maximization [8, 14]. The Table 1 shows the comparison of BGM with IPLSA using the evaluation code provided for the holiday dataset for calculating the Mean Average Precision(mAP) in both cases. The mAP results show that BGM performs better than IpLSA. As well as the memory usage of pLSA and IpLSA for creating the semantic indexes(training) much higher than BGM as their space complexity is of the order $O(TN_z)$ where N_z is the number of nonzero elements in the document term matrix and T is the number of topics.

4. Conclusion

In this paper, we propose a Bipartite Graph Model for semantic indexing. Our model is effective, and computationally efficient. Experimental results on many standard data sets demonstrate the utility of the method. Since the method does a just in time semantic analysis, it is scalable and efficient. It is also robust to parameters associated with the model.

-

References

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[2] A. Bosch, A. Zisserman, and X. Muoz. Scene classification via plsa. In *CIVR*, pages 307–312, 2003.

[3] G. Dorkó and C. Schmid. Object class recognition using discriminative local features. Technical report, INRIA - Rhone-Alpes, 2005.

[4] T. Hofmann. Probabilistic Latent Semantic Indexing. In *SIGIR*, pages 50–57, 1999.

[5] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, pages 304–317, 2008.

[6] A. N. Langville and C. D. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. 2006.

[7] C. Liangliang and L. Fei-Fei. Spatially coherent latent topic model for concurrent segmentation and classification of objects and scenes. In *ICCV*, pages 1–8, 2007.

[8] R. M. Neal and G. E. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. pages 355–368, 1999.

[9] U. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76:36106, 2007.

[10] B. C. Russell, A. A. Efros, J. Sivic, W. T. Freeman, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *CVPR*, 2006.

[11] H. Shao, T. Svobodal, T. Tuytelaars, and L. V. Gool. Hpat indexing for fast object/scene recognition based on local appearance. In *CIVR*, pages 307–312, 2003.

[12] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering object categories in image collections. In *ICCV*, 2005.

[13] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, pages 1470–1477, 2003.

[14] H. Wu, Y. Wang, and X. Cheng. Incremental probabilistic latent semantic analysis for automatic question recommendation. In *RecSys*, pages 99–106, 2008.

[15] H. Zha, X. He, C. Ding, H. Simon, and M. Gu. Bipartite graph partitioning and data clustering. In *CIKM*, pages 25–32, 2001.