Incremental On-line Semantic Indexing for Image Retrieval in Dynamic Databases

Suman Karthik, Chandrika Pulla and C.V.Jawahar
Center for Visual Information Technology International Institute of Information Technology Hyderabad
Hyderabad-500032, INDIA
sumankarthik@research.iiit.net, chandrika@research.iiit.net, jawahar@iiit.net

Abstract

Contemporary approaches to semantic indexing for bag of words image retrieval do not adapt well when the image or video collections dynamically get modified. In this paper, We propose an on-line incremental semantic indexing scheme for image retrieval in dynamic image collections.

Our main contributions are in the form of a method and a datastructure that tackle representation of the term document matrix and on-line semantic indexing where the database changes. We introduce a bipartite graph model (BGM) which is a scalable datastructure that aids in on-line semantic indexing. It can also be incrementally updated. BGM uses tf-idf values for building a semantic bipartite graph. We also introduce a cash flow Algorithm that works on the BGM to retrieve semantically relevant images from the database. We examine the properties of both BGM and Cash Flow algorithm through a series of experiments. Finally, we demonstrate how they can be effectively implemented to build large scale image retrieval systems in an incremental manner.

1. Introduction

We are interested in building scalable semantic indexing schemes for largescale, dynamic image collections. That is, given a query, we want to retrieve the relevant images from a constantly changing database that could range in size from millions to billions of images. This implies the presence of millions of concepts and subconcepts, over which the system is required to perform efficient retrieval of relevant images without any *apriori* knowledge of the concepts present in the data. Any solution to this problem must be computationally viable without sacrificing the quality of the retrieval.

Semantic learning has been extensively used in tandem with image retrieval to associate low level features to higher level concepts[15]. Semantic learning involves knowledge drawn from other sources like text annotations, ontologies, relevance feedback or browsing logs. However, semantic indexing tries to cluster the raw data in an unsupervised manner so as to enable the best retrieval performance. Semantic indexing like semantic learning is used for clustering similar documents together even though they do not cluster together in low level featurespace. Semantic indexing methods have found great success in text retrieval and classification domain [1, 10], this has lead to their application in the image retrieval domain using the bag of words model.

In recent years, the bag of visual words model has been adapted to vision problems[6, 25, 27, 16, 12, 17, 23, 8, 9, 2, 3] with great success. These approaches are shown to be well suited for tasks such as object categorization, object recognition, object retrieval and scene classification. The power of bag of words model to create efficient image and video retrieval systems has been explored by Sivic and Zisserman[27]. The problem of building large scale image retrieval systems has also been looked into by Torralba *et al.*[28]. Though not utilizing the bag of words model, they were able to build a highly scalable system with good performance characteristics. State of the art retrieval systems describe the images by sparse or dense descriptors and index them in an offline phase to build highly scalable retrieval systems. The success of bag of words model lies in its ability to quantize a very high dimensional feature space (using an algorithm like K-means)[5, 19] to build a compact codebook that encodes the similarity between descriptors and paves the way for efficient retrieval systems. The quality of the retrieval is further enhanced with the help of semantic indexing techniques like Probabilistic Latent Semantic Analysis(pLSA)[10] and Latent Dirichlet Allocation(LDA)[1]

Retrieval from a dynamic image collection poses a considerable challenge for the bag of words model. As new images are constantly added to an image collection the semantic index is unable to accurately represent the changing database. This necessitates re-computation of the semantic index at regular intervals which is time consuming and not

978-1-4244-3993-5/09/$25.00 ©2009 IEEE

scalable for large databases. As the number of images and associated concepts increases, these computations become prohibitively expensive. Semantic analysis of a document corpus can be viewed as unsupervised clustering of constituent words and documents around hidden or latent concepts in the corpus. Adaptation of PLSA and LDA to visual bag of words has provided promising results for static image databases[13, 21, 22, 24]. More recently semantic analysis is also being used in conjunction with spatial constraints for object segmentation [4, 22, 29], scene classification [3] and model learning [26, 14, 20].

In this paper, we study the nature of semantic indexing in dynamic image collections and design a semantic indexing technique that is effective in this setting and investigate its effectiveness, explore techniques for building scalable image retrieval systems for dynamic image collections. To this end, we propose a Bipartite Graph Model (BGM) for semantic indexing that converts the vector space model into a bipartite graph which can be incrementally updated with *just in time* semantic indexing. We further propose a Cash-Flow Algorithm that traverses the BGM to retrieve relevant images at runtime. We also explore the use of traditional text retrieval engines for building image retrieval systems in a dynamic setting. We show that the retrieval performance of BGM is comparable to pLSA while being highly scalable and much more efficient. Similarly we show the retrieval performance improvement over naive retrieval (TF-IDF based document retrieval with no semantic indexing). Finally we demonstrate the scalability, efficiency and real world retrieval capability of BGM in a near duplicate image retrieval application.

2. Offline Semantic Indexing

Semantic analysis techniques analyze the data within the term document matrix to uncover the latent relationships within it. The document term matrix or more generally an identifier vector matrix is used to represent data of the vector space model. Semantic analysis techniques use term co-occurrence data within the documents to cluster the data in an unsupervised manner. Both Probabilistic Latent Semantic Analysis(pLSA)[10] and Latent Dirichlet allocation(LDA)[1] use a generative model to explain the the observed data of words and documents, in context of the underlying latent data or concepts. These models have been applied successfully to various problems[13, 21, 22, 24, 2, 3]. However, these methods do not scale well for retrieval in large scale, highly dynamic image databases. LSI, pLSA and LDA are prohibitively costly to scale when dealing with very large datasets due to the resource intensive matrix computations needed. They can process a document corpus offline and cannot be updated in an incremental manner as is desirable in a dynamic environment where new data is constantly being added.

pLSA is a generative model from text retrieval literature[10]. It is used in tandem with the bag of words model for topic discovery in a document corpus. Due to the advent of bag of words model in visual recognition and retrieval pLSA made a transition into the visual bag of words domain as well. Following is a brief description of pLSA model for bag of visual words. Given that a collection of images is represented as documents $D = d_1, d_2, d_3, \ldots, d_n$ and the visual words are represented as $W = w_1, w_2, w_3, \ldots, w_m$ then the $M \times N$ matrix corresponding the visual words and images is called the term document matrix. In this matrix the element e_{ij} represents the term frequency for the term w_i in the image d_j. pLSA models the relation between the term and document entities using a latent variable z that represents a topic. Here the hidden variable that models co-occurrence data is of the form $Z = z_1, z_2, z_3, \ldots, z_o$. In this case a joint probability model $P(w|d)$ over the term document matrix is defined as

$$P(w|d) = \sum_{z \epsilon O} P(w|z)P(z|d)$$

$P(w|z)$ models topic specific distribution of words while $P(z|d)$ models the mixture of topics that comprise a document. Expectation Maximization is used to estimate the latent variables. Similarly LDA is a generative model that allows sets of observations to be explained by unobserved groups which explain why some parts of the data are similar. For example, if observations are words collected into documents, it implies that each document is a mixture of a small number of topics and that each word's creation is attributable to one of the document's topics. LDA has also been used extensively in the vision community for image retrieval and topic modeling.

However, LDA along with pLSA still cannot be used in real world large scale systems that are dynamic in nature. They are either too resource intensive in nature over large scale databases or too inefficient for a dynamic environment. Most large real world image and video collections like on-line user generated photo and video websites are too large for these methods to be employed effectively. One needs an on-line, incremental, efficient and scalable method of semantic indexing to deal with such data.

3. Bipartite Graph Model for Semantic Indexing

We suggest a semantic indexing model called *Bipartite Graph Model* (BGM) (Figure 1), that intuitively models and indexes the term document data in a scalable and incremental manner. BGM is designed to enhance the performance of large scale and highly dynamic image retrieval systems while at the same time providing an incremental concept centric indexing scheme with sublinear insertion and look-up performance.

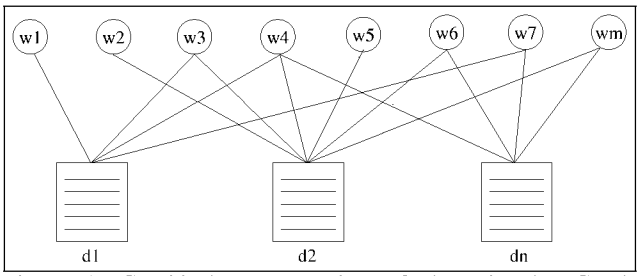

Figure 1. Graphical representation of the Bipartite Graph Model(BGM). d_i are the image or documents present in the corpus and w_j are the quantized feature vectors or words Present in the corpus. The edges connect words to documents in which they are present.

3.1. Semantic Similarity

Here we present a simple method for calculating semantic similarity. A document d_i from the global document collection D is assumed to have set of visual concepts C_i drawn from a global set of visual concepts C. The distribution B_{W_i} formed by the set of words W_i drawn from global set of words W within the document are generated by a mixture of these concepts P_{C_i}. Now the basic retrieval problem is to retrieve documents from the database that resemble the the mixture of concepts P_{C_i} in d_i as closely as possible. The intuition is that a document d_j with a word distribution B_{W_j} that is highly similar to B_{W_i} is likely to have been produced by mixture of concepts P_{C_j} that is highly similar to P_{C_i}. Here if f_{dis} is a function that calculates dissimilarity between two entities then

$$f^d_{dis}(d_i, d_j) \approx f^B_{dis}(d_i, d_j) \tag{1}$$

However a mixture of concepts P_{C_k} could be very similar to P_{C_i}, yet could generate B_{W_k}, where W_k is disjoint from W_i yet has a good amount of overlap with W_j. Then from Equation 1 it follows that

$$f^d_{dis}(d_i, d_k) \approx f^{\beta}_{dis}(d_i, d_j) + f^{\beta}_{dis}(d_j, d_k) \tag{2}$$

If there are m documents in the corpus the general form of Equation 2 could be written as

$$\sum_{x=1}^{m} f^{\beta}_{dis}(d_i, d_x) + \sum_{x=1}^{m} f^{\beta}_{dis}(d_x, d_k) + f^{\beta}_{dis}(d_i, d_k) \tag{3}$$

However this kind of approximation would require a transitive closure on the term document matrix, which would be prohibitively costly.

3.2. Bipartite Graph

The central idea behind the bipartite graph model is that the vector space model is encoded as a bipartite graph of words and documents. Here the edges are weighted with term frequencies of words in the documents as is relevant between each term and document. Each term is also associated with an inverse document frequency value.G is the bipartite graph such that

$$G = (W, D, E)$$
$$W = \{w_1, w_2 \ldots, w_n\}$$
$$D = \{d_1, d_2 \ldots, d_m\}$$
$$E = \{e^{d_1}_{w_1}, e^{d_2}_{w_7} \ldots, e^{d_m}_{w_n}\}$$
$$w_1 = IDF(w_1)$$
$$e^{d_1}_{w_1} = TF(w_1, d_1)$$

TF and IDF: In this model, the term and inverse document frequencies represent the word distribution within the document and in the corpus as a whole. These values together help in determining the importance of a word to a particular document. The term frequency representation of a document can be seen as a generative model of a document or histogram representation of a document and can be used to compute *KL-divergence* like dissimilarity. The IDF can be treated as a discriminative model of the document where the most discriminative words within a given document are given greater importance. The bipartite graph model combines both TF and IDF to be used in tandem like a hybrid generative-discriminative model.

3.3. Cash Flow Algorithm

We propose a *Cash Flow Algorithm* to find the semantically relevant documents in a document corpus in sublinear time using the *Bipartite Graph Model*. The main idea behind the cash flow algorithm is that, a query document(node) in the index is given cash to distribute among nodes that are relevant to it and they in turn propogate this cash distribution until the cash runs out. The higher the amount of cash flowing through a node the higher the relevance of the document(node) to the query. The cash flow algorithm is designed such that, at the time of querying, a single node or a set of nodes in the bipartite graph are infused with cash. If a node is a document node the cash is distributed among its edges in a quantity that is proportional to their flow capacity that is calculated by the normalized Term Frequency (TF) value. If the node is a word node it takes a portion of the cash it receives(Table 1) as a *service fee* and distributes the rest like the document node based on the flow capacity of its edges. The service fee at each word node is calculated by using the Inverse Document Frequency (IDF) value of the word. Hence the cash is propagated through the system until a point when the cash flowing through a node is considered too little to justify the overhead, this is judged at each node with a *cutoff* value that is the least amount of

cash needed for a node to forward the cash. At the start of every initialization each node that receives cash maintains a record of its cashflow. The end of a session is when there is no more cash flowing through the system due to the residual cashflow falling below the cutoff value. At this point the nodes that received cash are sorted based on the amount of cash that flowed through them. Total cashflow $Cash_{total}$for node N is

$$cash^N_{total} = cash^N_{previous} + cash^N_{current}$$

The two sorted node lists generated are the semantically most relevant documents and words to the given query according to the bipartite graph model.

```
def cashFlow(G, N, cash)
    cashFlow[N] += cash
    if N.type == WORD
        cash = cash * N.idf
    end
    if cash < cutoff
        exit
    end
    foreach node in G.connectedNodes (N)
        cashFlow(G, node, cash * G.tf(N, node))
    end
end
```

Table 1. *Cash Flow Algorithm* for *Bipartite Graph Model*

The cutoff value along with service fee ensures that the cash flowing through the system decays over time and especially distance from point of initialization and that the algorithm eventually converges. Documents are inserted into the 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. Insertions and deletions are linear in complexity to the number of words within a document. The system can be parallelized easily. The graph is thread safe allowing simultaneous reading and only requires conflict resolution when more than one thread is trying to update the IDF value of a word node. The cash flow algorithm essentially is a graph cut algorithm that divides the nodes in the bipartite graph into relevant and nonrelevant sets.

4. Experiments

4.1. Naive Retrieval vs BGM

First we study the retrieval performance of BGM and its variants when compared to simple retrieval without any semantic indexing involved. We make use a Flickr sports dataset with 9 categories and a Flickr animal dataset with 5 categories both of which combined have more than nine thousand images. We extracted SIFT vectors from the images and quantize the feature space using Kmeans quantization with a vocabulary size of 10,000 and 5,000 respectively for sports and animals datasets and build a BGM as well as a simple inverted index for comparison of retrieval performance. We used four different variants of the cashflow algorithm to traverse the BGM. We measure the retrieval performance of an algorithm by calculating its F-score.

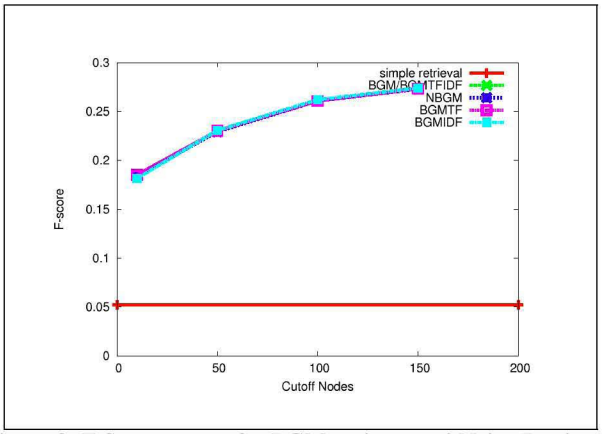

Figure 3. F-Score curves for BGM variants and Naive Retrieval, BGM clearly outperforms naive retrieval.

From Figure 3 we see that the performance of BGM algorithm compared to naive retrieval. BGM performs significantly better than simple retrieval which forms the baseline with an F-score of 0.05. As the number of cutoff nodes increases the performance increase of BGM begins to taper, this is due to the fall in recall as more and more noise from non relevant image enters the system in successive iterations. Figure 3 and Figure 2 show how BGM is able to retrieve images that cannot be retrieved by simple retrieval.

Tweaking the edge flow capacities and node service fee leads to different variants of BGM. Naive BGM or NBGM does not have edge flow capacities. BGMTF has edge flow capacities and no service fee. BGMIDF has service fee and no edge flow capacities. BGMTFIDF or BGM has both edge flow capacities and service fee. Since the number of nodes traversed by the different cashflow algorithms for the same cutoff varies drastically as seen in Figure 2, we used number of nodes traversed as the cutoff condition to compare the different algorithms.

4.2. pLSA vs BGM

The objective of this experiment is to compare the offline retrieval performance of pLSA with that of the on-line retrieval performance of BGM. For this experiment we have used holiday dataset[11] ,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

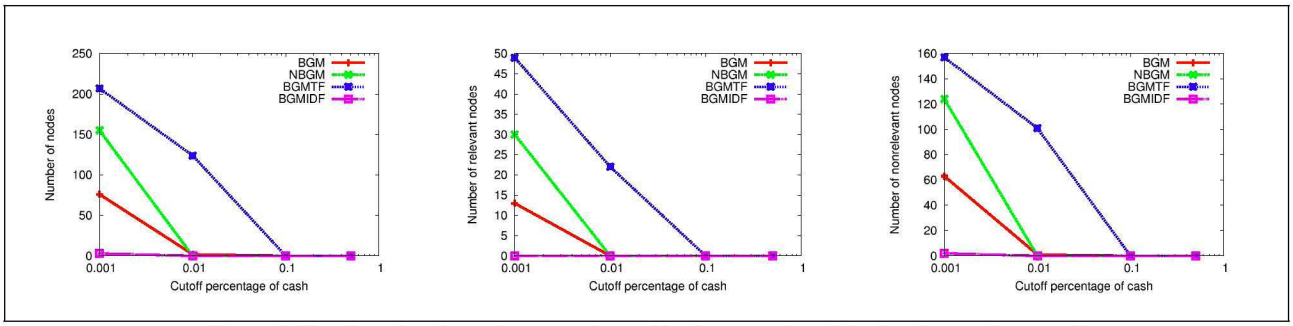

Figure 2. Number of nodes, relevant nodes and irrelevant nodes visited under varying cutoff

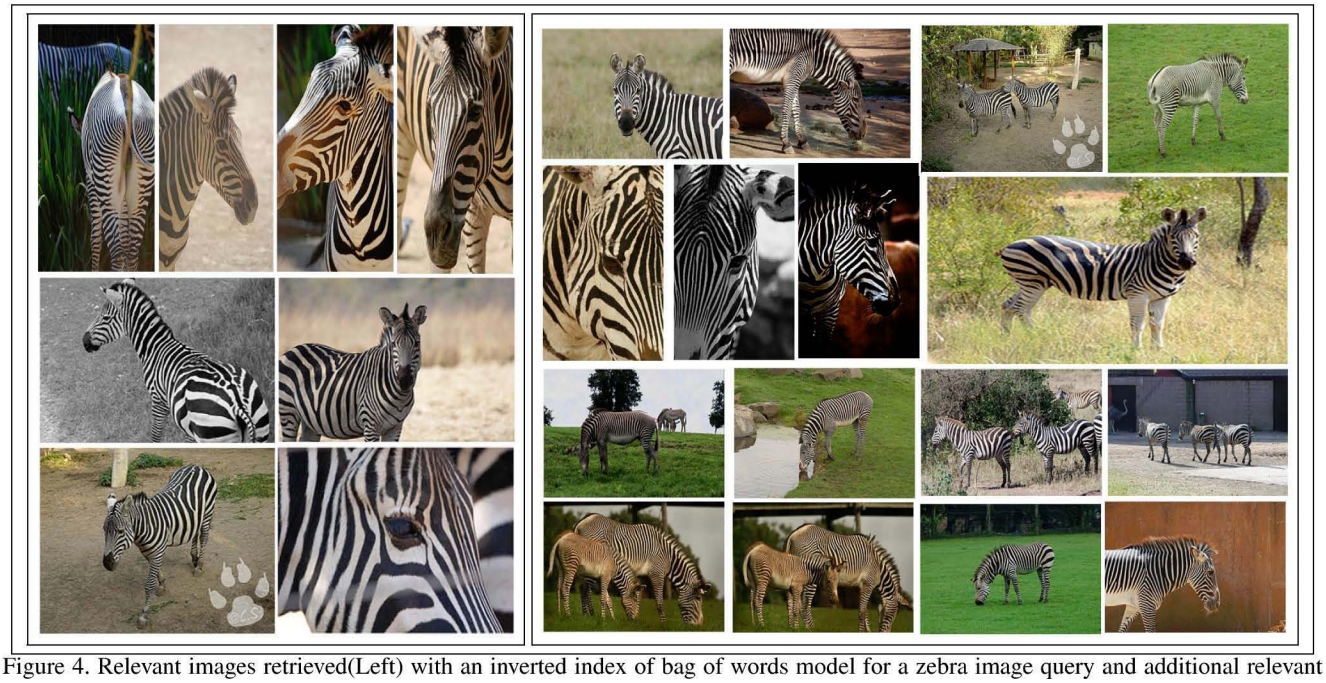

Figure 4. Relevant images retrieved(Left) with an inverted index of bag of words model for a zebra image query and additional relevant images(Right) retrieved by BGM for the same query. BGM significantly outperforms Naive retrieval

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[7]. 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 each image was represented as a histogram of visual words. Aggregating these histograms the term document matrix was represented by A of the order $M \times N$ where M is the vocabulary size and N is the document corpus size. Here $A(w_i, d_j)$ is the term frequency of the term w_i pertaining to the document d_j. This term document matrix is used for pLSA where a hidden aspect variable Z_k is associated with each occurrence of a visual word w_i in an image d_j. The conditional probability $P(w|d)$ is

$$P(w_i|d_j) = \sum_{k=1}^{K} P(z_k|d_j)P(w_i|z_k)$$

where $P(z_k|d_j)$ is the probability of the topic z_k occurring in the document d_j and $P(w_i|z_k)$ is the probability of the word w_i occurring in a particular topic z_k. The pLSA(EM) model generates $P(z)$,$P(z|d)$,$P(w|z)$. The EM model was initialized with latent 500 topics which is similar to the number of categories in the dataset. Once the model converges all the topic probabilities for all the documents in the corpus are generated. For retrieval the Euclidean distance of the documents over topic probabilities was used to

retrieve the 10 most similar images.

For BGM each image was represented as a document comprising of visual words. Then a term document matrix was created where each row representing m_i representing the term frequencies of the relevant document was 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 cash flow 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. The mAP results show that BGM performs very comparably to pLSA. However, when one looks at the memory usage and time taken for creating the semantic indexes(training) in both cases one can clearly notice the difference. Here, BGM outperforms pLSA by the order of 100. However, the real advantage of BGM is noticed when adding another image to the index only takes a few milliseconds while for pLSA the computation of the entire semantic index needs to be done again incurring high time and memory costs.

Model	mAP	time	space
Probabilistic LSA	0.642	5473s	3267Mb
BGM + CashFlow	0.594	42s	57Mb

Table 2. Mean Average Precision for both BGM and pLSA for the holiday dataset, along with time taken to perform semantic indexing and memory space used during indexing

4.3. Retrieval Performance

Text retrieval systems and search engines have become a commodity with large numbers of *off the shelf* and open-source systems available. These can be easily scaled to handle billions of documents and millions of queries with ease. This kind of scalability has always been a challenge for image retrieval systems, but bag of words model enables one to build such image retrieval systems[18]. Can this gap be eliminated by using text retrieval systems for image retrieval. We explore this possibility by building a full scale image retrieval system by using a text search engine. Accomplishing this means access to proven technology from basic text indexing schemes to advanced crawling, index sharding, index optimization and ranking algorithms and implementations. In order to accomplish this we must first convert image documents to text documents. We achieve this using a simple hash function that converts codebook bins into text strings and subsequently images into text documents. We build our image retrieval engine using the Ferret search library which is a ruby port of the *Apache Lucene* project.

BGM was used in conjunction with the Ferret index to achieve semantic indexing. The space complexity of PLSA 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. Thus 10 million nonzero elements in the document term matrix would necessitate a memory requirement of no less than 10GB. At this scale pLSA takes a few hours to compute. Both space and time complexity of PLSA make it an impractical choice in a dynamic environment. BGM, on the other hand is a data structure that is resident on disk, which makes updating BGM highly efficient due to absence of any significant computation. In order to put BGM and the Ferret index through their paces we adjusted vector quantization parameters to create a large and descriptive vocabulary of more than 6 million words. Each image in the dataset on average has 110 visual words across 100,000 images. The average time taken to insert an image into BGM is of the order 0.0134 seconds the same as the time it takes for an image to be inserted into the ferret index. The average response time for a query for Ferret is 0.29 seconds while the average response time for a BGM query is 2.42 seconds. The discrepancy in response times can be attributed to the multiple levels of graph traversal by the Cashflow algorithm in case of BGM. Even though BGM improves retrieval performance (Figure 2 and Figure 3) by a large margin, the discrepancy in retrieval time is very low as clearly seen in Figure 5. The response time of BGM and Ferret can be improved by sharding the index across multiple machines while at the same time providing high scalability.

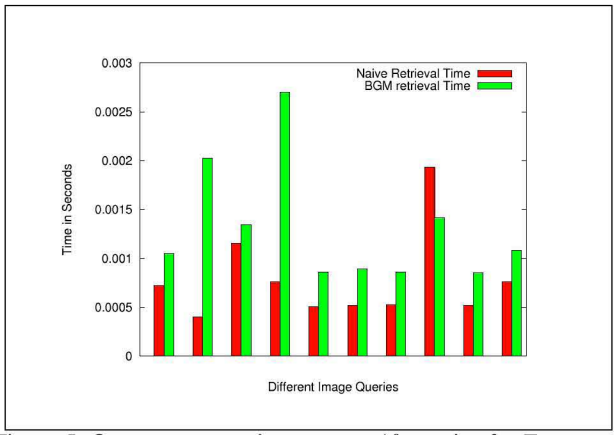

Figure 5. Query response times across 10 queries for Ferret and BGM, One can clearly notice that the retrieval times are very comparable to one another

5. Near Duplicate Detection

Near duplicate detection in videos and images involves finding images that are almost similar to the query image with only slight changes, like successive frames in a video. It is a challenging problem for bag of words based image retrieval methods. Some of the interesting problems that need to be tackled involve scalable and efficient vector quantization and semantic learning. Here we discuss the application of BGM over a large dataset.

The data for the application comes from frames of various motion pictures. Each frame is processed by interest point detectors and subsequently these points are represented using the SIFT descriptor. Once each frame is processed the frame is inserted into the content server while the descriptors are processed by an incremental on-line vector quantization scheme that converts the descriptors into text based visual words. These collections of words or documents each having a relevant image in the content server are then indexed using a text search library Ferret. When a new query frame is submitted to the system first the new document is treated as the node initiating the flow in BGM and the TF-IDF index is used to boost the terms and submitted as a query to the Ferret index. This same process continues at every subsequent document node until sufficient number of duplicates with the relevant scores are retrieved. Even with such massive amounts of data like movie frames the system is able to scale very effectively. The indexing time for 1000 images after nearly a million images are already present in the index is of the order of 100s of seconds. Similarly the retrieval time is on average less than 2 seconds over the entire index. BGM significantly outperforms naive retrieval by discovering and retrieving a varied range of near duplicate frames than Naive retrieval.

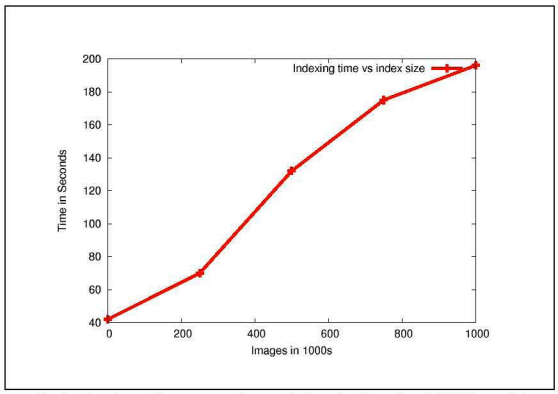

Figure 6. indexing time vs size of the index in 1000's of images, even at a million images the time taken for inserting a batch into the semantic index is under 200 seconds

6. Conclusion

We have proposed a new Online, incremental semantic indexing method for large scale dynamic image collections. We studied its properties and compared it with the current state of the art in semantic indexing. We have also demonstrated its utility through an application for near duplication image and video detection.

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. Munoz. Scene classification via pLSA. In *ECCV*, 2006.

[3] A. Bosch, A. Zisserman, and X. Munoz. Scene classification using a hybrid generative/discriminative approach. *PAMI*, 30(4), 2008.

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

[5] O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *BMVC*, 2008.

[6] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision*, pages 59–74, 2004.

[7] G. Dorkó and C. Schmid. Object class recognition using discriminative local features. Rapport de recherche RR-5497, INRIA - Rhone-Alpes, February 2005.

[8] R. Fergus. *Visual Object Category Recognition*. PhD thesis, University of Oxford, Dec. 2005.

[9] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from Google's image search. In *ICCV*, 2005.

[10] T. Hofmann. Probabilistic Latent Semantic Indexing. In *Proceedings of the 22nd Annual ACM Conference on Research and Development in Information Retrieval*, pages 50–57, Berkeley, California, August 1999.

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

[12] D. Larlus and F. Jurie. Latent mixture vocabularies for object categorization and segmentation. *Journal of Image and Vision Comput.*, 27(5):523–534, 2009.

[13] F.-F. Li and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR*, pages 524–531, 2005.

[14] L.-J. Li, G. Wang, and null Li Fei-Fei. Optimol: automatic online picture collection via incremental model learning. *CVPR*, 0:1–8, 2007.

[15] Y. Liu, D. Zhang, G. Lu, and W.-Y. Ma. A survey of content-based image retrieval with high-level semantics. *Pattern Recogn.*, 40(1):262–282, 2007.

[16] E. Nowak, F. Jurie, and B. Triggs. Sampling strategies for bag-of-features image classification. In *ECCV*, pages IV: 490–503, 2006.

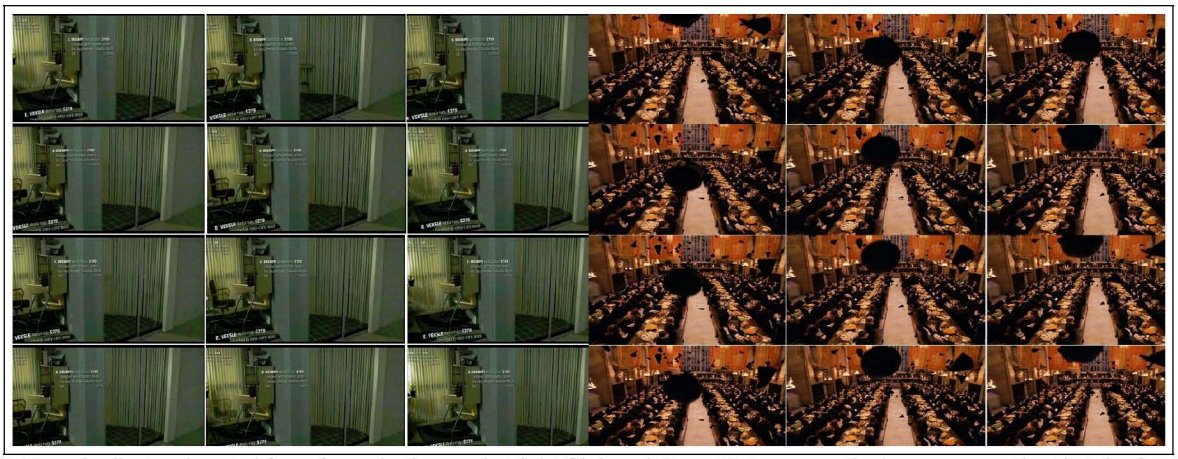

Figure 7. Near duplicates detected for a frame in the movie Fight Club and Harry Potter respectively, one can notice that the frames only differ slightly from each other.

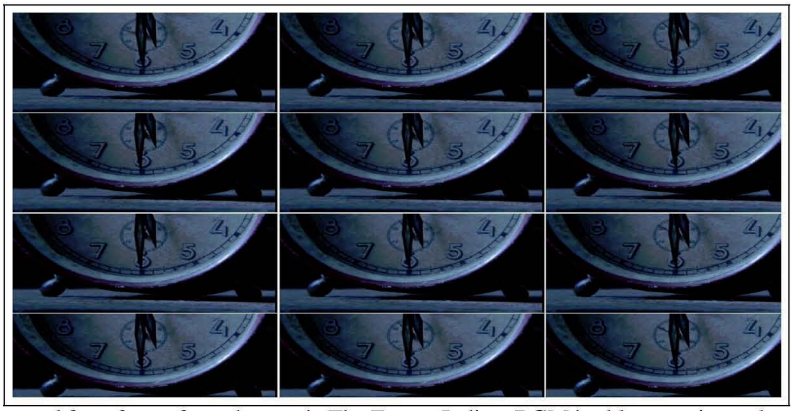

Figure 8. Near duplicates detected for a frame from the movie The Fastest Indian, BGM is able to retrieve a larger number of near duplicate frames than Naive retrieval.

[17] F. Perronnin, C. Dance, G. Csurka, and M. Bressan. Adapted vocabularies for generic visual categorization. In *ECCV*, pages 464–475, 2006.

[18] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.

[19] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*, 2008.

[20] J. Philbin, J. Sivic, and A. Zisserman. Geometric lda: A generative model for particular object discovery. In *BMVC*, 2008.

[21] P. Quelhas, F. Monay, J.-M. Odobez, D. Gatica-Perez, T. Tuytelaars, and L. V. Gool. Modeling scenes with local descriptors and latent aspects. In *ICCV*, pages 883–890, 2005.

[22] 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.

[23] F. Schroff, A. Criminisi, and A. Zisserman. Single-Histogram Class Models for Image Segmentation. In *ICVGIP*, 2006.

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

[25] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering objects and their location in images. volume 1, pages 370–377, 2005.

[26] J. Sivic, B. C. Russell, A. Zisserman, W. T. Freeman, and A. A. Efros. Unsupervised discovery of visual object class hierarchies. In *CVPR*, 2008.

[27] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, volume 2, pages 1470–1477, Oct. 2003.

[28] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. pages 1–8, 2008.

[29] X. Wang and E. Grimson. Spatial latent dirichlet allocation. In *NIPS*, volume 20, 2007.