Using LSI for Text Classification in the Presence of Background Text

Submitted by: Vishad Patel (7277)
Bhavin Mehta (1325)
Dhaval Patel (3678)

Summary

Text classification is the process to classify the unlabeled data with the use of labeled data. Different methodologies exist for this text classification that uses labeled data to classify unlabeled data. This paper presents use of Latent Semantic Indexing (LSI) for text classification. The key of this paper is to improve classification accuracy by incorporating unlabeled data and other forms of background knowledge to collection of existing labeled data. Classification process that uses background knowledge and LSI as a classification tool achieves wider semantic space that was unreachable with the labeled data alone. Results of methodology that uses LSI with and without the inclusion of background text are plotted and compared. Results are compared with other classification process that also incorporates unlabeled data and background text.

Why classification is needed?
Classification process is to classify any unlabeled data with the use of available labeled data. Need of using classification process is because of large availability of inexpensive unlabeled data compare to expensive and scarcity of labeled data. On the other hand it is tedious and expensive to label unlabeled data with existing classification methodologies. LSI is the methodology that can classify large amount of unlabeled data with the help of small-labeled data and background knowledge.

Why background knowledge is incorporated?
By incorporating background knowledge, first they (background knowledge) are classified using existing labeled data with LSI tool. This background knowledge is now added to the original labeled data (training data-documents) and new classifier is formed. This new classifier is then used as the basis for labeling new unlabeled data. This is iterative process that helps to improve the new class formed that contains the words that even did not occur in the training examples (documents). **Iterative process also provides richer and more reliable patterns for data in the given domain.**

Why LSI as a classification tool?
LSI is the classification tool that neither classifies the background knowledge nor it **directly** compares to any training examples or test examples. LSI is an automatic method that re-describes the textual data in a new smaller semantic space. LSI assumes there exists an underlying semantic structure among the training examples (documents) and terms in a corpus. LSI places documents that appear related in close proximity in the new space. Now to classify any unlabeled data, unlabeled data (query) is run using LSI. LSI places this test example in newly created semantic space close to those documents which are semantically related to it. Compute distance between unlabeled data in new semantic space with other classes and highest score class is used to classify this unlabeled data.

Explanation of LSI:

LSI is the retrieval strategy uses Matrix Computation as a basis. All other strategy directly matches key words. Since the same concept can be described using many different keywords, this type of matching is prone to failure. People used same query for same concept only 20% of time. Goal is to represent the underlying semantics of documents rather than matching keywords. LSI uses Singular Value Decomposition to capture this semantic structure. This filters noise found in a document, such that two documents that have the same semantic concepts are located close to one-another in a multi-dimensional space.

Three major advantages of using the LSI representation with the following labels: synonymy, polysemy and term dependence.

Synonymy: Synonymy refers to the fact that the same underlying concept can be described using different terms. Traditional retrieval strategies have trouble discovering documents on the same topic that use a different vocabulary. In LSI, the concept in question as well as all documents that are related to it are all likely to be represented by a similar weighted combination of indexing variables.

Polysemy: Polysemy describes words that have more than one meaning, which is common property of language. Large numbers of polysemous words in the query can reduce the precision of a search significantly. By using a reduced representation in LSI, one hopes to remove some "noise" from the data, which could be described as rare and less important usages of certain terms. (however that this would work only when the real meaning is close to the average meaning. Since the LSI term vector is just a weighted average of the different meanings of the term, when the real meaning differs from the average meaning, LSI may actually reduce the quality of the search).

Term Dependence: The traditional vector space model assumes term independence and terms serve as the orthogonal basis vectors of the vector space. Since there are strong associations between terms in language, this assumption is never satisfied. While term independence represents the most reasonable first-order approximation, it should be possible to obtain improved performance by using term associations in the retrieval process. Adding common phrases as search items is a simple application of this approach. On the other hand, the LSI factors are orthogonal by definition, and terms are positioned in the reduced space in a way that reflects the correlations in their use across documents. It is very difficult to take advantage of term associations without dramatically increasing the computational requirements of the retrieval problem. While the LSI solution is difficult to compute for large collections, it need only be constructed once for the entire collection and performance at retrieval time is not affected.

LSI process is really straight forward.

A term-document matrix is constructed such that location (i, j) indicates the number of times the term i appears in document j.
A SVD of this matrix is computed.

Steps to the SVD of the matrix $A \in R^{mxn}$:

1. Find the eigen values of the matrix A^TA and arrange them in descending order.
2. Find the number of nonzero eigen values of the matrix A^TA.
3. Find the orthogonal eigenvectors of the matrix A^TA corresponding to the obtained eigenvalues and arrange them in the same order to form the column-vectors of the matrix $V \in R^{nxn}$
4. Form a diagonal matrix $\sum \in R^{mxn}$ placing on the leading diagonal of it the square roots $\sigma i=\sqrt{\lambda} i$ of eigenvalues found of the matrix $A\mathit{T}A$ in descending order
5. Find the column vectors of the matrix $U \in R^{mxm}$:
   $ui = \sigma i\text{-}1\ A\ Vi\ (i = 1\text{:}r)$
6. Add to the matrix *U* the rest of the column vectors using the Gram-Schmidt orthogonalization process
7. $A = U\sum V^T$

In $U\sum V^T$, $\sum$ is a diagonal matrix and values in $\sum$ are referred to as singular values. Select top k sorted singular values and set all other values to zero and corresponding columns of U and rows of V^T are deleted. Top k values are used as a means of developing a "latent semantic" representation. Prior work shows that smaller values of k (100-300) achieve effective results. New matrix after limiting value of k is the product of simplified three matrices $A` = U_2\sum_2 V_2^T$
New matrix represents original relationship as a set of k orthogonal factors. Each factor is a linearly independent concept and can be considered as a means of combining meanings of different terms and documents. Training examples (documents) are re-expressed in same semantic space (using these factors). Query (here unlabeled data) is represented in same space by treating query as document vector. It is done by multiplying the transpose of term vector of query (q^T) with $U_2\sum_2^{-1}$, ($q^TU_2\sum_2^{-1}$).
Find distances between query and documents those are represented in same semantic space using cosine metric, which are similarity measures between documents and query.
Relevant documents to query are those documents, which are having higher cosine distance value than some cutoff point. This is how LSI works.

LSI for Text Classification:

Single example to be labeled (classified) is judged by LSI methodology similar to a range of different training examples. And labeling is done according to how different documents vote for the label of new example. Result of LSI can be looked at as a table of different tuples **(train-example, train-class, cosine-distance)** with one line for each training example of training (document) collection. Combine score of all training examples of each class to get score of each class for new example.
Cohen and Hirsh used noisy-or operation to get one similarity value per class for given new example to be labeled.

Final score for single class is given by

$\mathbf{1 - {}^{n}\prod_{i=1} (1-s_i)}$ where, $\{ s_1, s_2, \ldots, s_n\}$ are scores of all documents of one class.

Final scores of all classes is sorted and the class with highest score for given unlabeled data is used to classify it.

This way LSI is used in text – classification.

Incorporating background knowledge:

LSI is used in text classification as explained above. But incorporation of background knowledge is more helpful in classification. **Fist** this **background knowledge** that is easily available and inexpensive **is classified using available labeled data** and classification tool **LSI**. This **classified background knowledge** is **incorporated** to the **original training examples**. Now new unlabeled data could be classified with the help of these **new classifier** formed by incorporating background text.

For this first term-by-document matrix is formed for all training examples and classified background knowledge. LSI is run on this term-by-document matrix and SVD of this matrix is computed and original relationship is expressed as a set of top k orthogonal factors. This **new term-by-document matrix is a model** of the space that was unobtainable with the training examples alone. The larger matrix contains words those even did not occur in the training examples at all. It also **provides richer and more reliable patterns for data in the given domain**.

Now new test example (**unlabeled data**) to be labeled is **re-described** in the **new space** created above using LSI. Voting of each training example to label unlabeled data is computed as distance (score) in the space. Using operation (Cohen Hirsh noisy-or operation) that is described above **score of each class to label unlabeled data is computed**. **Class with highest score** is **used** to **classify this unlabeled data.**

It is important to for the background knowledge to be similar in content to the original training set that it is combined with. If the background knowledge is totally unrelated to the training corpus then features would be unrelated to the actual classification task though LSI might successfully model the background knowledge. Because in that case documents (training examples) might stay far away though they are semantically similar in the multi dimensional space.

Example:

Consider collection originally contains training examples on car and related. Now background knowledge (unlabeled data) those contains data on automobile is incorporated. First data on automobile is classified using existing collection of training examples with LSI classification tool. LSI is run on this background knowledge and existing training examples and distances between different classes and background knowledge is found. Class with highest similarity is used to classify this background knowledge. Here, this class will be the class of automobile that is used to classify data on automobile. Now this background knowledge is treated as training examples of automobile class and incorporated into existing collection of documents.

Now when query **car repairing** is run with newly formed collection it could retrieve documents those even don't contain the term car but related to query semantically (may contain automobile repairing place) and thus serve the purpose of adding background text using LSI.

Thus, for text classification incorporation of background knowledge is a good technique to expose to a large semantic collection. And for this classification use of LSI dramatically increases the semantic representation of new collection. That's why use of LSI for incorporation of background knowledge is the best classification tool compared to other tools because its places the background knowledge at proper place (in proper class) in existing collection close to semantically similar documents more efficiently.

What is data set?
They apply LSI tool to classify the input data set . This data set is same as previously used on text classification in the presence of background text. The purpose of using the same data set is that so that they can compare the result that obtain with this LSI tool with background knowledge with the other alternative classification tool.

How this data set is used for experiments?
The graph plotted for different percent of data vs. accuracy for both LSI without background knowledge and with background knowledge. For each run they took one fifth of the total data as test example and one fourth of the total data as training example now they make the test example constant and increase the percent of the training example and plot the graph for 20 ,40, 60, 80, 100 percent of data.

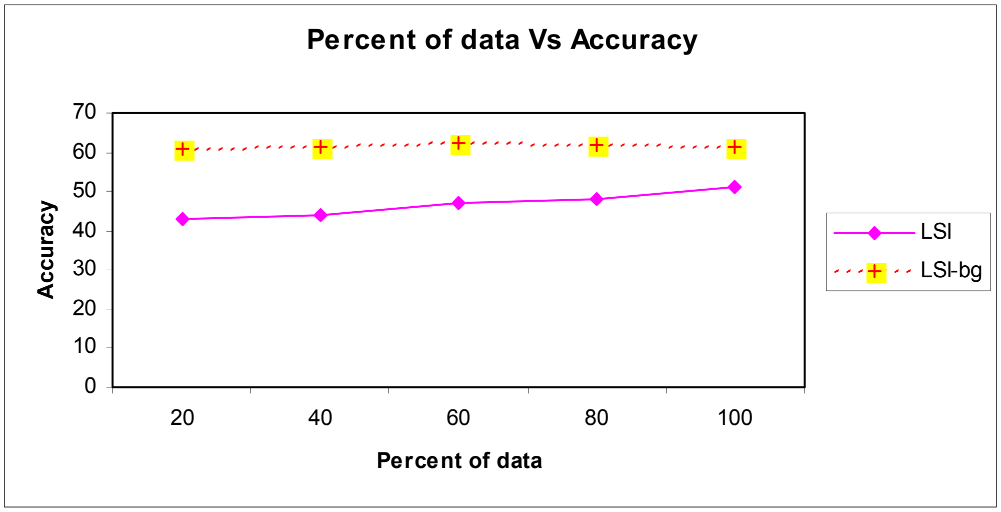
This data set is categorized in to four different categories.

1. **Technical papers**
2. **Web page titles**
3. **WebKB**
4. **Newsgroups**

Technical papers:

This data set contains the categories related to technical papers. They created a data set from the physic paper archives. They took all the title of the paper that contain the information about astrophysics as well as condensed matter by downloading and put the abstract of all these papers as a background text by downloading them without their label. By doing this approach has no more direct access to this background.

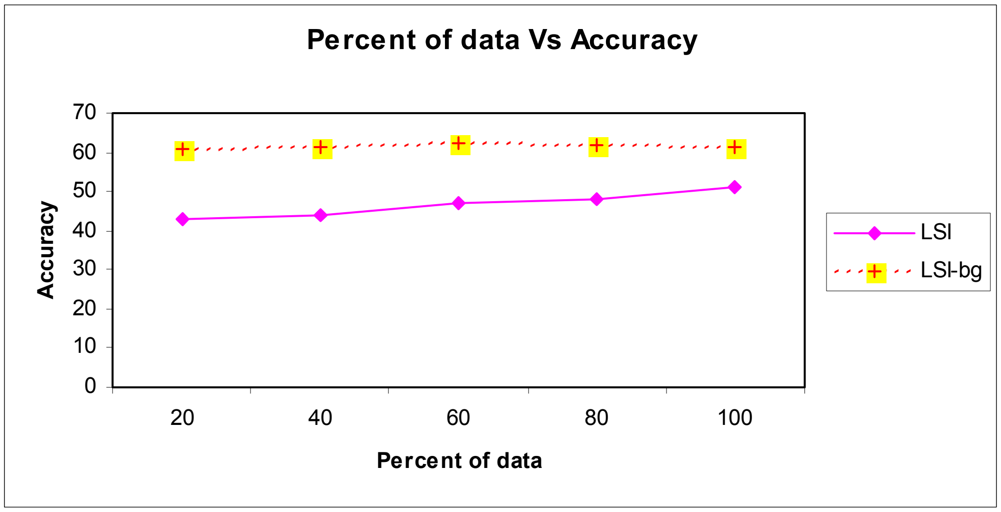
Fig1: LSI and LSI-bg for the two-class paper title problem



Webpage titles:

They took this data set from the NetVet site that contain the web page title of hourse, cow, birds, cats, etc. Now each of this title had a URL link they took this link that pointing to another page as background knowledge by incorporating the first 100 words from that page. Note: The URLs that were not reachable were ignored by program that has created the background knowledge database.

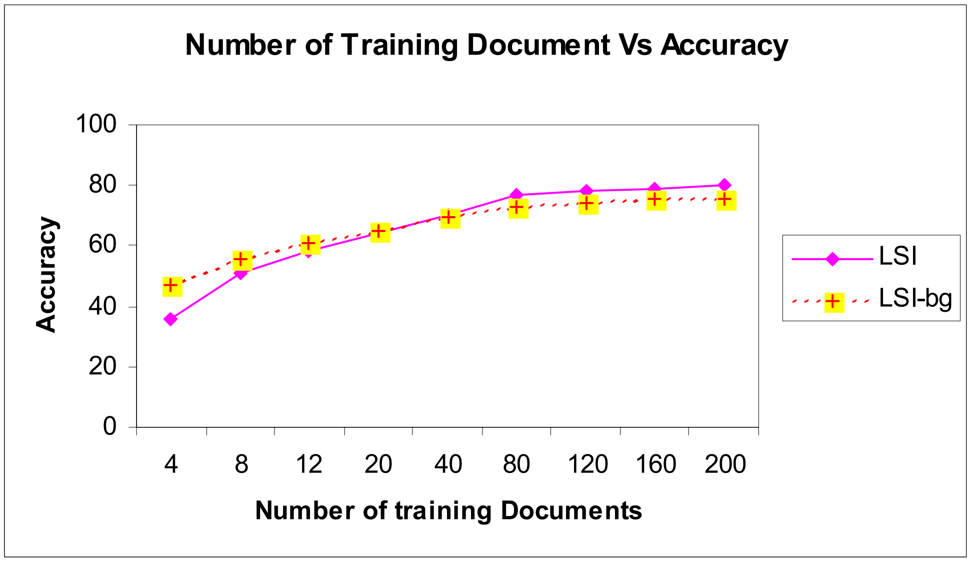
Fig 2: LSI and LSI-bg for the NetVet problem



WebKB:

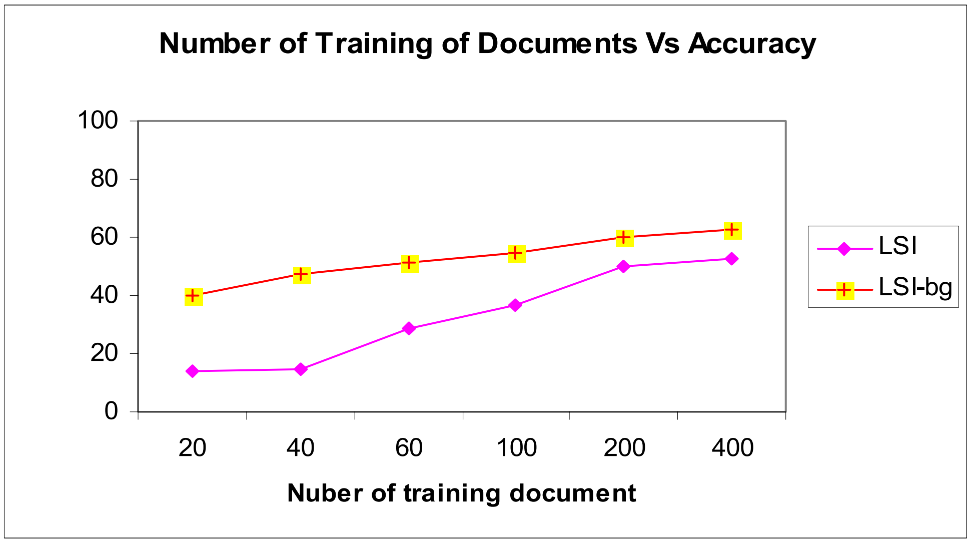
This data set contain the information about the web pages from the computer science department but they have selected only the class named student, project, faculty and project. For this set the background knowledge is totally unlabeled and for optimization for EM technique they took only first 300 words as a back ground knowledge.

Fig 3: Evaluation of results for WebKB four class problem



20 News groups:
This data set contains the articles from the 20 different newsgroups. The latest 4000 articles are added as the training examples and random 10000 are used as a background text.

Fig 4: LSI and LSI-bg for the 20 Newsgroups problem



Comparison of the result that obtained by running LSI tool with above data set and the data set that do not have any in corporate background knowledge in it.

Fig: 1 and Fig: 2 shows Results for Physical data and NetVet Data. Results for both shows accuracy using LSI- bg changed very little as can seen by the flatness of line. This lead to believed that background knowledge compensates for the limited training data. In each case reduction of error increased as training the training set size decreased.

Fig: 3 shows LSI-bg only outperform LSI on small training sets. LSI-bg degrade as training class size grown compare with LSI without background text. It might because of the incorporating optimized background knowledge that is for EM but not clear whether it works for LSI or not.

Fig: 4 shows that increment in training set size does not degrade in accuracy. LSI without background knowledge perform extremely poor.

Comparison of result of the LSI tool with the alternative tool EM both having same data sets.
EM is Expectation maximization techniques that use probabilistic method and iterative approach. Disadvantages of EM are
1. It requires much more computation than more standard techniques.
2. The chief disadvantage of EM is slow convergence.
3. EM algorithm might converge to local optima of the cost function

Fig 5: Comparison LSI with EM for Physical Data

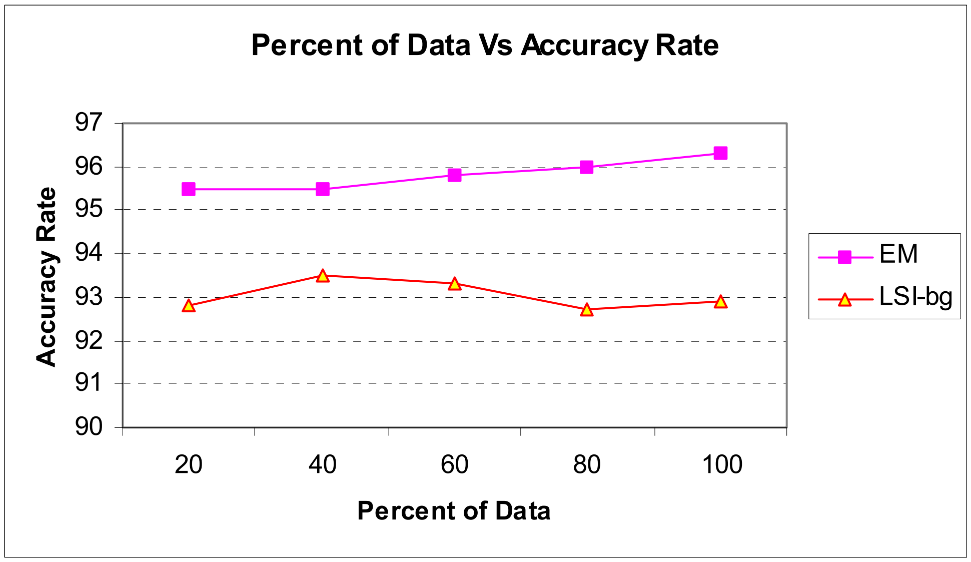
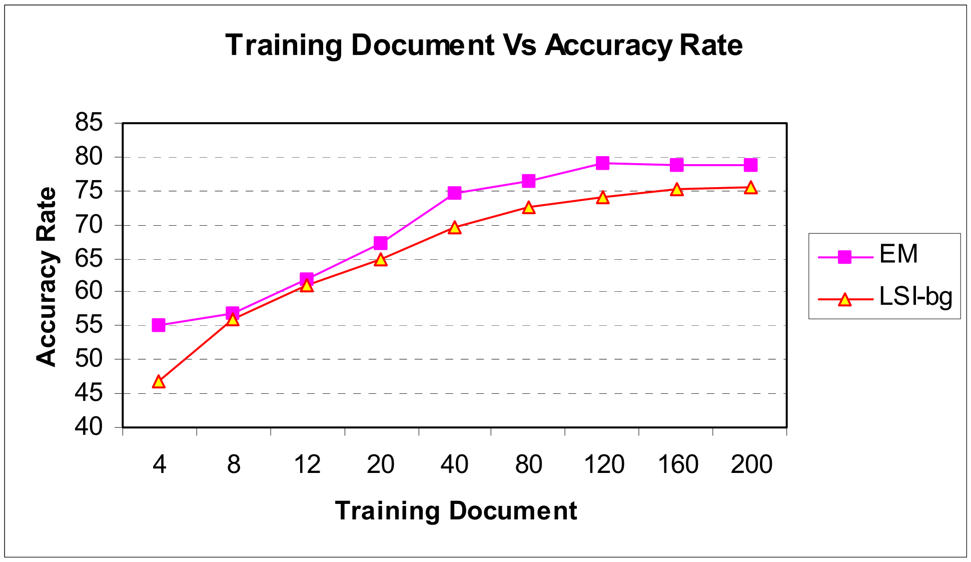


Fig 6: Comparison LSI with EM for WebKB



.

On the Physics data and WebKB data, EM is far superior in all cases.
For Physics Data Zelikovitz and Hirsh showed that in this domain the simple process of labeling the background text using the training data and then adding the resulting data to the training data without any further processing, gets extremely high accuracy as well**.**

Fig 7: Comparison LSI with EM for NetVet Data

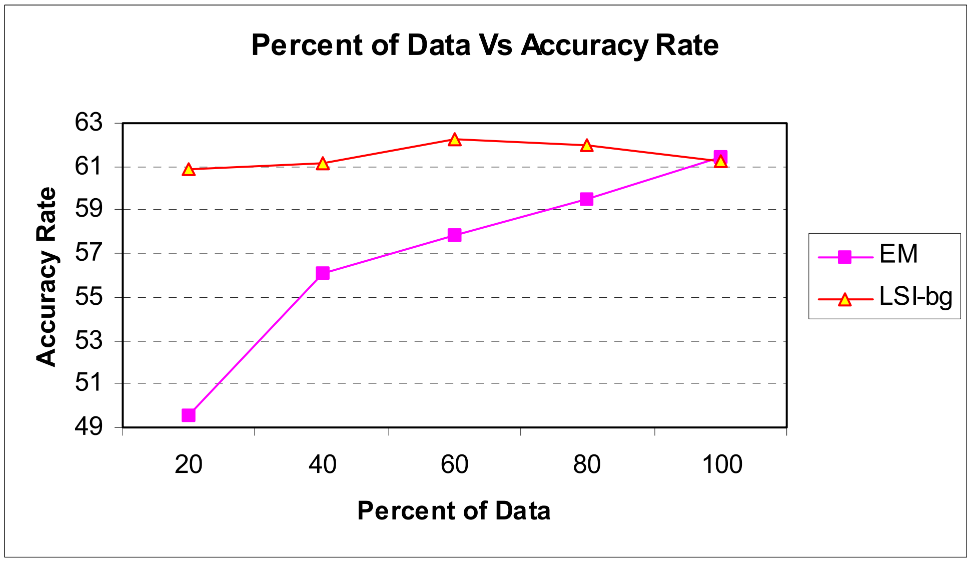
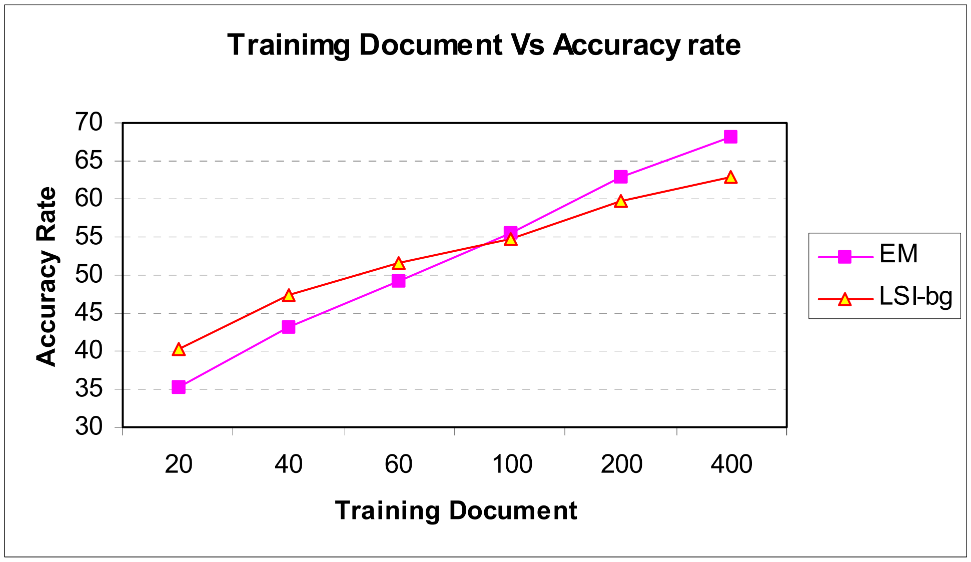


Fig 8: Comparison LSI with EM for 20 Newsgroups



For small data set in both the NetVet domain and 20 Newsgroups domain, LSI-bg outperforms EM. As more training examples are added to these problems LSI-bg does not perform as well. This phenomenon occurs mostly in all data set.