Latent Semantic Indexing
Group 7

Boyan Georgiev
University of Mannheim

Chung Chi Leung
University of Mannheim

Kuan Min Chen
University of Mannheim

Abstract

This paper discusses the implementation of a LSI-based (latent semantic indexing) information retrieval model, and evaluates its performance against the traditional vector space model (VSM) on a collection with 18,828 documents.

1 Introduction

1.1 Application Area

Many classic information retrieval models essentially try to match words of queries with words of documents. Examples of such techniques include the boolean retrieval model, vector space model (Salton et al., 1975) and probabilistic models. The amount and importance of term overlap are estimated, and documents are retrieved based on the estimates. However, a potential problem arises when users want to retrieve documents based on a conceptual topic.

Concepts may be expressed in many ways, beyond the limits of terms used in a query. Synonymy describes the state whereby many words may have the same meaning, and users in different contexts may describe the same information need using various terms. The prevalence of synonyms is one of the factors for lower recall performance of retrieval systems. On the other hand, polysemy describes the state whereby a word (e.g. chip) may have multiple meanings, and a document containing such a word is not necessarily of interest. This tends to decrease precision performance.

Synonymy, and to a smaller degree, polysemy, can be addressed by a method known as latent semantic analysis (LSA). LSA analyses the relationships between a set of documents and the terms they contain, to expose their underlying semantic structure (Deerwester et al., 1990). A mathematical technique called singular value decomposition (SVD) is applied on a term-document matrix, in which rows correspond to terms and columns correspond to documents. In doing so, the high-dimensional, sparse matrix is reduced to a lower-dimensional, dense latent space.

Through dimension reduction, it is possible for documents with different term-usage profiles to be mapped into the same vector space of latent factors; terms and documents of similar latent structure are placed near one another, thus accomplishing improvements on traditional methods.

In the context of information retrieval, LSA is often called latent semantic indexing (LSI). This paper focuses on the retrieval performance of LSI on a high-dimensional dataset. It details the implementation of such a model, including but not limited to, the construction of a term-document matrix, the singular value decomposition, the query-document similarity metrics, and the evaluation of LSI in comparison to a traditional term-matching model.

1.2 Data

In this paper, we will evaluate the models on the 20 Newsgroups dataset, a popular collection used for experiments in text applications of machine learning techniques such as text classification and clustering. It consists of 18,828 news articles (hereinafter referred to as *documents* for simplicity and commonness) of varying lengths. They are almost evenly partitioned across 20 different newsgroups, as its name suggests, into topics ranging from religion to sports to technology.

The dataset was chosen for its considerable document size and coverage across newsgroups, allowing for an appropriate high-dimensional term-document matrix, while the provided categories aid in understanding of the context of each document, for query-generation and evaluation purposes that follow.

1.3 Approach

Documents were represented in a bag-of-words model, facilitating the construction of a term-document matrix. which will then both be used in LSI and VSM. The SVD technique was then applied on the matrix, with the subsequent matrices allowing for query-document comparison within the richer latent space. The vector space model (VSM) was chosen amongst the traditional retrieval models, for evaluation against the LSI model. Documents were retrieved based on similarities which best estimate their relevance.

In order to collect relevance assessments of documents retrieved by the two models, the pooling approach was utilized. For each query, evaluation metrics such as average precision could then be trivially calculated. What was also of interest was the dimension k, which controls the representational power of LSI. Different values of k were also experimented with, to explore its effects on the model performance.

2 Implementation

The information retrieval system was implemented in Python3.5. Substantial part of the functionality was carried out by using the language default data structures and basic modules. External libraries were used for more demanding tasks, like `nltk` for stemming of terms and `scikit-learn` for singular value decomposition and efficient implementations of sparse matrices.

2.1 Bag of Words

Obtaining bag-of-words (BoW) representations for the documents is a common step in most information retrieval and also text mining applications. The BoW for a certain document d simply contains all unique terms present in d and their raw frequencies. Note that extracting the terms requires a couple of preprocessing steps. First, we replaced all non-alphabetic characters by spaces. This was necessary because the collection, D, contains a lot of emails which have a lot of tokens without any semantic information, like special characters, tags and signatures. This step also considerably reduced the size of the vocabulary, which decreased the computational complexity later on. Next, we tokenized the documents by splitting them on whitespaces. Then, the tokens were stemmed using the Porter stemming algorithm, thus obtaining the terms in d. Finally for each document we constructed the BoW representation - a dictionary with the unique terms as keys and term frequencies as values.

2.2 Term-document Matrix

The term-document matrix, A, is a convenient way to represent the relations between terms and documents in the collection. The number of rows is equal to the size of the vocabulary and the number of columns is the number of documents in the collection. For instance, the (i, j) element from A, w_{ij}, captures the importance of term i, t_i, for document j, d_j. If t_i is not present in d_j, then $w_{ij} = 0$.

Before creating the term-document matrix A, it was necessary to obtain a vocabulary for the whole collection. For each unique term in the collection, we had to compute three components: the positional index i, the inverse document frequency, idf, and the inverted index (a list of documents which contain the term). i determines the row of some term in the term-document matrix, the idf contributes to the weight of this term and the inverted index is used to improve the efficiency of VSM-based retrieval. So the vocabulary is a dictionary with the unique terms from the collection as keys and triples (i, idf, list of documents) as values. It also important to note that terms which appear only in a single document were removed, to reduce the total vocabulary size. Then, by iteration over the terms in the bag-of-words representations, the term-document matrix was constructed. For each term, a triple of information was extracted: row index (contained in the vocabulary), column index (simply the index of the document) and the weight w_{ij}, calculated according to equation 1, where f_{t_i,d_j} is the frequency of t_i in d_j. The resulting matrix has 45,827 rows, 18,828 columns and 1,981,506 non-zero elements. Due to its sparsity, the term-document matrix was stored as a `scipy.sparse.csr_matrix`. At this point, we also computed the L_2 norms of the document vectors (columns of A) and stored them for subsequent calculation of similarities between documents and queries.

$$w_{ij} = \frac{1 + log(f_{t_i,d_j})}{1 + log(max\{f_{t',d_j} : t' \in d_j\})} \times log\left(\frac{|D|}{|d' \in D : t_i \in d'|}\right) \quad (1)$$

2.3 Singular Value Decomposition

The best rank k approximation of A in terms of Frobenius norm is $A_k = U_k \Sigma_k V_k^T$, where the columns of U_k are the first k left singular vectors, Σ_k is a diagonal matrix with the singular values, and the rows of V_k^T are the first k right singular vectors. We used `scipy.sparse.linalg` to the obtain the $k = 1000$ approximation of A. It is important to note that this step was quite expensive and also required a lot of memory, which is why using LSI for rapidly-changing collections imposes a computational challenge. Another crucial observation is that U_k and V_k^T are dense matrices and take a lot of space. For instance in our application the size of U_k was roughly 366 MB. For the LSI retrieval we stored U_k^T, $\Sigma_k V_k^T$ and also the L_2 norms of the columns of $\Sigma_k V_k^T$.

2.4 Retrieval

As already mentioned, we implemented two information retrieval systems: LSI and VSM. For both of them, the first necessary step was to obtain a vector representation of the query q. The terms from the query were extracted using the same preprocessing methods used to obtain the initial BoW representations for the documents, in subsection 2.1. Then, for the query terms which were present in the vocabulary, we extracted the corresponding row indices of A (also U_k) and the idf scores. Let q_v by the vector representation of q. q_v's image in the latent concept space given by the SVD, is simply $q_l = U_k^T q_v$. Of course, we do not need all rows from U_k because q_v generally has only few elements which are larger than zero and the indices of those elements are precisely the rows from U_k which are necessary for the computation. Then the similarities between q_l and the latent vectors of the documents (columns of $\Sigma_k V_k^T$) were calculated according to $\frac{q_l^T [\Sigma_k V_k^T]_{*j}}{||[\Sigma_k V_k^T]_{*j}||}$, where $[\Sigma_k V_k^T]_{*j}$ is the j-th column of $\Sigma_k V_k^T$. Note that the resulting value is not the actual cosine between q_l and $[\Sigma_k V_k^T]_{*j}$ but the ranking of the documents will be the same as the one with cosines. Unlike the VSM, where we don't have to consider any of the documents which do not contain any query terms, for LSI we have to compute all similarities because it is very likely that all documents and all query terms contribute to the latent topics. This is another aspect which makes the LSI a more expensive IR system.

For the VSM-based retrieval, we used the alread-calculated q_v and from the inverted index derived in 2.2, we extracted only the documents which contain some of the query terms (column indices of A). Then, those documents were ranked according to $\frac{q_v^T A_{*j}}{||A_{*j}||}$ where j is the index of a document which contains at least one query term. At this point, we used the L_2 norms of A's columns obtained in 2.2.

At this point we have mention a very important property of the LSI retrieval. If we use the full rank SVD the LSI and the VSM will produce identical rankings of documents. Consider the way we derive the dot products between q_v and the columns of A.

$$q_v^T A = q_v^T U \Sigma V^T = (U^T q_v)^T \Sigma V^T = q_l^T \Sigma V^T$$

We can also show that the A and ΣV^T will have the same L_2 column norms due to the orthogonality of U. Because of this it is clear that the rankings of LSI with full rank SVD and VSM will coincide. This observation is important for our understanding of the LSI concept.

3 Evaluation

3.1 Methodology

Relevance assessment is the process of determining whether a document satisfies the information needs of a user for a given query. As collections are very often large, this is a time-consuming and expensive step involving human participation and judgment. A standard approach in collecting such assessments is *pooling*. In pooling, relevance is assessed over a subset of the collection (as opposed to the entire collection), that is formed from the top-ranked documents by the retrieval engines of interest (usually the ones to be evaluated) (Manning et al., 2009).

The relevance assessments then form the basis for evaluation of ranked retrieval results, which are commonplace for modern-day search engines. One common evaluation metric is *mean average precision* (MAP), which provides a single-figure measure of quality across recall levels and queries. MAP is known for its stability and discriminatory ability (Manning et al., 2009). For a single query q, average precision (AP) is the unweighted average of precision values calculated at ranks of relevant documents $d_1, d_2, ..., d_m$:

$$AP(q) = \frac{1}{m} \sum_{k=1}^{m} P(R_k)$$

document	*query*	*rj*
comp.graphics/38807	povray issues after upgrade	1
comp.graphics/38829	povray issues after upgrade	0
comp.graphics/38890	cheap voltage converters	0
comp.graphics/38629	robotics modem for sale	0
comp.windows.x/67185	dynamic window title bar	1

Table 1: Annotation table

MAP is then simply AP averaged over the set of queries Q:

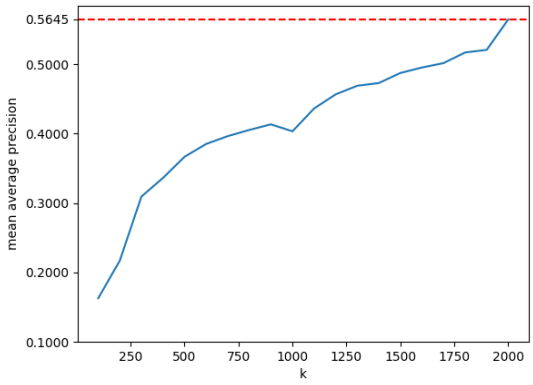
$$MAP = \frac{1}{|Q|} \sum_{j=1}^{|Q|} AP(q_j)$$

First we created 50 queries by examining a portion of the documents. To evaluate the performance of the LSI retrieval system we needed sets of relevant documents for those queries. For *pooling*, we derived the union of the top 10 documents returned by the LSI and by the VSM per query. On average a union contained roughly 15 documents, so we obtained 774 (*document*, *query*) pairs which had to be annotated for relevance. To facilitate the annotation process we created a csv file, Table 1 shows some random rows from it. Essentially the column ***rj*** had to be filled manually, by determining whether or not the query is relevant (1) for the document or not (0). For instance, the query *"povray issues after upgrade"* is relevant for the document *"comp.graphics/38807"*. This process was very time consuming and contained certain amount of ambiguity. We realized that the concept of relevance is very flexible and because of that different people can annotate the same (*document*, *query*) pairs in different ways.

3.2 Choosing rank for LSI

(Dumais, 2004) argues that choosing an appropriate rank for the latent concept space (k) is a difficult task which can considerably affect the performance of the system. Generally, for small k the quality of the retrieved rankings tends to be low. This result is somehow intuitive because if the collection naturally has more than k identifiable topics, some topics have to be merged together to obtain a latent space with rank k. Dumais further claims that often with increasing k, initially the quality improves then for high values of k it starts declining and converges to the performance of the VSM. Of course, this is not always the case. Let us consider a very extreme example, a collection in which none of the documents shares any terms

Figure 1: Rank vs. MAP



with the rest of the documents. Although, a person might be able to identify certain topics, the LSI will not be able to do that. In Figure 1 we can see that the LSI achieves the highest MAP for the biggest value of k. The not very satisfying result is that for k from 1 to 2000 the LSI does not perform better than the VSM which has a MAP of 0.583 In the next section we will discuss different factors which can explain the result.

4 Discussion

4.1 Structure of the collection

The fundamental assumption for LSI is that there is an underlying relation between the terms which can be used to retrieve documents without overlapping terms with the query. Taking the example of synonym pairs *doctor* (appearing in the query) and *physician* (appearing in some document d), one would expect LSI to retrieve document d. However, on closer inspection of the documents, it becomes evident why this would be difficult on the collection used in this paper.

The nature of the documents in the 20 Newsgroup dataset is such that they take the form of emails or forum posts. In addition to them being shorter than one would expect, the documents are also written in a non-verbose and colloquial manner. As such, synonyms of nouns and verbs are less likely to appear together. This contributes to the documents deviating from an 'average' article, which then have negative implications on LSI. For this reason, the collection used could be the biggest factor in the unsatisfactory results.

Because of the fact that documents are in general brief and not as content-rich as articles coming from, for example, newspapers or scientific

journals, it would have been rather more meaningful to have generated the term-document matrix from a larger corpus, such as Wikipedia pages. Each of the documents and queries can then be projected into the latent space space with matrices generated from its SVD decomposition.

4.2 Relevance judgements

Another factor that could explain the results acquired lies in the reliability of relevance judgements. Humans are not devices that reliably churn out spot-on judgements of relevance between documents for a query. Judgements then tend to be idiosyncratic and variable. For documents of the nature described earlier, the task complexity increases as it requires even more effort to determine the contextual topic. It would not be unreasonable to assume that while assessing relevance, the human eye also subconsciously seeks out terms that match the query, before being certain of relevance.

One way to mitigate this could be to consider the agreement between judges on relevance assessments. The kappa statistic is a popular measure, which corrects the agreement rate for the rate of chance agreement (Cohen, 1960). A high kappa value would indicate that there is good agreement between judges on their relevance annotations, and thus validate their usage. However, this was not carried out in this paper, due to the lack of human resource to annotate the 774 retrieved documents more than once.

4.3 Query generation

One last factor to consider is the query generation process. When generating queries, we first looked into sampled documents, before thinking of appropriate queries that would retrieve such documents. This was done so as to increase the likelihood of retrieving any documents at all. To illustrate this point, take the example of the query "*space travel competitions*". The document referenced to generate this query did not contain any of the three search terms, but instead contained "*moon colony prize race*". The query was generated thinking that simply replacing terms with synonyms would suffice for LSI to extract such documents. However, not only was the particular document not retrieved by LSI, it also performed less adequately than the VSM, with an AP of 0.125 for VSM and only 0.01 for LSI. This shows that purely replacing terms with their synonyms is not an effective way to generate queries.

5 Summary

In this project we implemented a LSI-based retrieval system. We expected it to outperform a traditional model like VSM due to its ability to uncover latent structures in the collection of documents. Unfortunately, our results largely did not agree with our expectations, which prompted a deeper analysis of the collection and the evaluation methodology. We found out that LSI heavily depends on the structure of the collection that it is applied on and it would be interesting to repeat the experiment with a different collection with exisiting relevance judgments.

References

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391.

Susan T. Dumais. 2004. Latent semantic analysis. *Annual review of information science and technology*, 38(1):188–230.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2009. *An Introduction to Information Retrieval*, volume 1. Cambridge University Press, www.cambridge.org.

Gerard M. Salton, Andrew Wong, and Chungshu Yang. 1975. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.