LSI vs. Wordnet Ontology in Dimension Reduction for Information Retrieval*

Pavel Moravec, Michal Kolovrat, and Václav Snášel

Department of Computer Science, FEI, VŠB - Technical University of Ostrava,
17. listopadu 15, 708 33, Ostrava-Poruba, Czech Republic
{pavel.moravec, michal.kolovrat}@vsb.cz

Abstract. In the area of information retrieval, the dimension of document vectors plays an important role. Firstly, with higher dimensions index structures suffer the "curse of dimensionality" and their efficiency rapidly decreases. Secondly, we may not use exact words when looking for a document, thus we miss some relevant documents. LSI (Latent Semantic Indexing) is a numerical method, which discovers latent semantic in documents by creating concepts from existing terms. However, it is hard to compute LSI. In this article, we offer a replacement of LSI with a projection matrix created from WordNet hierarchy and compare it with LSI.

Keywords: vector model, latent semantic indexing, LSI, information retrieval, dimension reduction, WordNet

1 Introduction

The *information retrieval* [13,3] deals among other things with storage and retrieval of multimedia data, which can be usually represented as vectors in multidimensional space. This is especially suitable for *text retrieval*, where we store a *collection* (or *corpus*) of texts. There are several models used in text retrieval, from which we will use the *vector model* [10,12] providing qualitatively better results than the *boolean model* [13], which combines word matching with boolean operators.

In vector model, we have to solve several problems. The ones addressed in this article are the size of resulting index, search efficiency and precision and search for documents similar to the query.

To measure the improvement of a new indexing method, we can use several measures, both quantitative and qualitative. The quantitative measures show us the performance of an indexing structure, they can be for example number of disc accesses – *disc access cost* (DAC) – or total time of performed indexing and search – *wall clock time*. The qualitative measures tell us how good does this new indexing structure reflect the reality when obtaining an *answer set* A for a

* This paper was partially supported by GAČR 201/03/1318 grant.

given *query* Q. The most commonly used qualitative measures are *precision* (P) and *recall* (R) [3], where precision is a fraction of relevant documents in answer set and recall is a fraction of retrieved relevant documents in all relevant ones.

In second chapter, we will describe classic vector model and above mentioned problems. In third, we will describe *latent semantic indexing* (*LSI*). In fourth chapter a basic description of English *WordNet* ontology will be given. In fifth chapter we will offer a way how to use WordNet for concept creation instead of LSI and in sixth we will present comparison of our approach with LSI and random projection on real documents from TREC collection.

2 Vector Model

In vector model, a document D_j is represented as a vector d_j of term weights, which record the extent of importance of the term for the document.

To portrait the vector model, we usually use an $n \times m$ *term-by-document matrix* A, having n rows – term vectors $t_1 \dots t_n$ – where n is the total number of terms in collection and m columns – document vectors $d_1, \dots d_m$, where m is the size of collection C.

Term weights can be calculated in many different ways – $t_i \in \{0, 1\}$, as a membership grade to a fuzzy set, or as a product of functions of term frequency both in a document and in the whole collection [11] (usually $tf * idf$ – count of term occurrences in the document multiplied by a logarithm of the inverse portion of documents containing the term). Sometimes is the normalisation of document vectors applied during index generation phase to make the calculation in retrieval phase faster.

A query Q is represented as an n-dimensional vector q in the same vector space as the document vectors. There are several ways how to search for relevant documents. Generally, we can compute some L_n metrics to represent similarity of query and document vectors. However, in text retrieval can be better results obtained by computing *cosine measure*:

$$sim_{cos}(d_j, q) = \frac{d_j q}{||d_j|| \times ||q||} = \frac{\sum_{i=1}^{n} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{n} w_{i,q}^2}}$$

As one can see, we do not only obtain documents which are considered relevant, but according to their distance (or similarity) to the query vector, we can order them and obtain rank for every document in answer set. We can define a *threshold* t, too, meaning that all documents closer than this threshold will be considered relevant, whilst the rest will be irrelevant. However, the choice of the threshold is not exact and its value is usually determined experimentally.

The main problem of vector model is, that the document vectors have a big dimension (e.g. 150,000) and are quite sparse (i.e. most coordinates are zero). If we store them as classical vectors, the storage volume is huge – consider size of a term-by-document matrix consisting of 100,000 terms and 200,000 documents.

We can use existing compression schemes for the term-by-document matrix representation like the *compressed column storage* (*CCS*) to conserve memory, but the co-ordinate access time is much longer and we are limited by the fact, that we cannot access the term vectors quickly. Or we can use combined storage with both CCS and *compressed row storage* (*CRS*). Anyway, updating would still be a problem.

The second problem is the so-called *"curse of dimensionality"*, which causes classical indexing structures like M-trees, R-trees [6], A-trees, iDistance, etc. to perform same or even worse than sequential scan in higher dimension. Moreover, the vectors are placed almost equidistantly from each other, which makes clustering ineffective.

Third, even there is a better chance that we can find relevant documents when using some terms which are not contained in them, the synonyms and other semantically related words are not taken in account.

The first two problems can be addressed for queries containing only a few words by *inverted list*, which is in fact compressed storage of term vectors. Only term vectors for terms contained in a query Q are loaded and processed, computing rank for all documents containing at least one of the terms at once. However, the inverted list is not efficient when searching for similar documents, because significant part of index must be processed.

3 Latent Semantic Indexing

Latent semantic indexing (*LSI*) [3,4] is an algebraic extension of classical vector model. First, we decompose the term-by-document matrix A by either *principal component analysis* (*PCA*), which computes eigenvalues and eigenvectors of covariance matrix or *singular value decomposition* (*SVD*), calculating singular values and singular vectors of A. SVD is especially suitable in its variant for sparse matrices (Lanczos [7]).

Theorem 1. (Singular value decomposition): *Let A is an $n \times m$ rank-r matrix. Be $\sigma_1 \geq \cdots \geq \sigma_r$ eigenvalues of a matrix $\sqrt{AA^T}$. Then there exist orthogonal matrices $U = (u_1, \ldots, u_r)$ and $V = (v_1, \ldots, v_r)$, whose column vectors are orthonormal, and a diagonal matrix $\Sigma = diag(\sigma_1, \ldots, \sigma_r)$. The decomposition $A = U\Sigma V^T$ is called* singular decomposition *of matrix A and numbers $\sigma_1, \ldots, \sigma_r$ are* singular values *of the matrix A. Columns of U (or V) are called* left (or right) singular vectors *of matrix A.*

Now we have a decomposition of original term-by-document matrix A. Needless to say, the left and right singular vectors are not sparse. We have at most r nonzero singular numbers, where rank r is smaller of the two matrix dimensions. However, we would not conserve much memory by storing the term-by-document matrix this way. Luckily, because the singular values usually fall quickly, we can take only k greatest singular values and corresponding singular vector coordinates and create a *k-reduced singular decomposition* of A.

Definition 1.: *Let us have* $k, 0 < k < r$ *and singular value decomposition of* A

$$A = U\Sigma V^T = (U_k U_0) \begin{pmatrix} \Sigma_k & 0 \\ 0 & \Sigma_0 \end{pmatrix} \begin{pmatrix} V_k^T \\ V_0^T \end{pmatrix}$$

We call $A_k = U_k \Sigma_k V_k^T$ *a* k*-reduced singular value decomposition* (*rank-*k *SVD*).

We would not conserve any space with the matrix A_k. So instead of the A_k matrix, a concept-by-document matrix $D_k = V_k \Sigma_k$ with k concepts is used. To convert a query Q to the concept-space, we create $q_k = U_k^T q$ [1]. The similarity of terms in concept space can be calculated from $U_k \Sigma_k$ [2].

If every document contains only one topic (for more details see [9]), we obtain a *latent semantics* – semantically related terms will be close in concept space and will result in similar answer set when querying. This addresses the third of above mentioned problems. And since the first co-ordinates of D_k have the greatest influence on similarity, the clustering results are better.

Experimentally was k determined to several tens or hundreds (e.g. 50–250), exact value of k is however a mystery; it is dependent on the number of topics in collection. For a illustration of rank-k SVD see Figure 1.

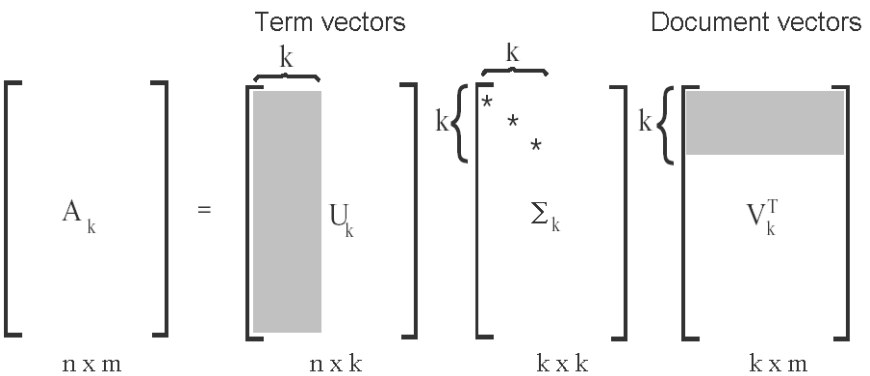

Fig. 1. k-reduced singular value decomposition

Rank-k SVD is the best rank-k approximation of original matrix A. This means, that any other decomposition will increase the sum of squares of matrix $A - A_k$. However, this does not mean, that we could not obtain better precision and recall values with a different approximation.

The LSI is hard to compute and once computed, it reflects only the decomposition of original term-by-document matrix. If several hundreds of documents

[1] The second approach is to use a matrix $D'_k = V_k$ instead of D_k and $q'_k = U_k^T \Sigma_k^{-1}$

[2] or U_k in second approach

or terms have to be added to existing decomposition (*folding-in*), the decomposition may become inaccurate. The recalculation of LSI is expensive, so it is impossible to recalculate LSI every time documents and terms are inserted. The *SVD-Updating* [8] is a partial solution, but since the error slightly increases with inserted documents and terms, the recalculation of SVD may be needed soon or later.

4 WordNet Ontology

WordNet is an online lexical reference system whose design is inspired by current psycholinguistic theories of human lexical memory. English nouns, verbs, adjectives and adverbs are organised into synonym sets, each representing one underlying lexical concept.

The goal of WordNet project is the creation of dictionary and thesaurus, which could be used intuitively. The next purpose of WordNet is the support for automatic text analysis and artificial intelligence. WordNet is also useful for determining semantic connections between sets of synonyms, for tracing morphological connections between words.

The ontology is organised not only by the "is-the-synonym-of" relation; the verbs and nouns are hierarchically organised via the *hypernym/hyponym* relation (superior/inferior concepts), too. An example of hypernyms for "ontology" is given in figure 2.

```
psychological feature
  cognition, knowledge, noesis
    content, cognitive content, mental object
      knowledge domain, knowledge base
        discipline, subject, field, study, bailiwick, ...
          humanistic discipline, humanities, liberal arts, arts
            philosophy
              metaphysics
                ontology
```

Fig. 2. Example of hypernyms of the term "ontology"

EuroWordNet is a multilingual database with WordNets for several European languages (Dutch, Italian, Spanish, German, French, Czech and Estonian). The WordNets are structured in the same way as the American WordNet for English (Princeton WordNet) in terms of synsets (sets of synonymous words) with basic semantic relations between them. Each WordNet represents a unique language-internal system of lexicalizations.

In addition, the WordNets are linked to an *Inter-Lingual-Index*, based on the Princeton WordNet. Via this index, the languages are interconnected so that it is possible to go from the words in one language to similar words in any other language.

This index also gives access to a shared top-ontology of 63 semantic distinctions which provides a common semantic framework for all the languages, while language specific properties are maintained in the individual WordNets. The database can be used, among others, for monolingual and cross-lingual information retrieval, which was demonstrated by the users in the project.

5 Using WordNet Hypernyms instead of LSI Concepts

As mentioned above, the calculation of SVD is quite difficult and since the resulting matrices U and V are dense, memory can be exhausted quite quickly. So we face a question, how to create concepts for given document collection.

One possibility is the usage of Papadimitriou's two-step algorithm [9] combining *random projection* (see e.g. [1]) with LSI. Simply said, we first create a pseudoconcept-by-document matrix A' with a suitable number of pseudoconcepts by multiplication of a zero-mean unit-variance projection matrix and the term-by document matrix A. In second step we calculate rank-$2k$ LSI of A', which gives us a very good approximation of rank-k LSI of original matrix A.

We experimentally verified this method recently and showed that the approximation error against LSI is low, however we do not obtain same concepts as with original LSI. Because there is usually not the same number of singular values as for the original matrix A (e.g. 60'000), but only the number for the reduced dimension (e.g. 1000), the concepts are created differently and are almost equal (having similar singular values). This results in poor clustering and worse selection of k.

However, we know the hierarchy of concepts defined by the WordNet synonym/hypernym organisation. We can use all hypernyms of given term from l top levels, applying a fraction of term weight dependent on its level in WordNet to hypernym weight in concept-by-document matrix. This would give us a different linear combination of term vectors than classical LSI, with non-negative concept weights.

This way we can create a term-to-concept projection matrix, applying adequate parts of each term to its hypernyms. The top l levels of hypernyms will give us new concepts. The results may be worse than in case of LSI, but would give us a better starting point than random projection which chooses the pseudoconcepts as a random linear combination of terms.

Is the resulting dimension too high for convenient direct use (but not too high to make the LSI calculation problematic as in case of original matrix A), we can replace random projection in the Papadimitriou's two-step approximate LSI calculation method by the term-to-concept projection matrix and calculate LSI on generated term-by-concept matrix, which will improve the response time.

The problem is, that hypernym hierarchy was created only for nouns and verbs. Adjectives and adverbs can't be handled this way, which brings some complications. We can either use all concepts from this area, or silently ignore them, which will cause either an increase of reduced dimension or worse recall. The same problem is with numbers and names. While a number can be easily identified, we can create a category "Number" to place all numbers in, we cannot do this with names, so we either ignore them, too, or we can create a predefined number of random concepts which would contain terms not found in WordNet with a given weight.

6 Experimental Results

For the comparability with LSI, 5000 Los Angeles Times articles from the TREC 5 document collection were indexed. LSI into a dimension of 100 and 200 was calculated using both classical and two-step algorithm. For comparison, the random projection was calculated, too.

Tests were run with English Wordnet version 2.0; the WordNet concepts were used both directly and as the first step in the two-step algorithm instead of random projection. In first case, two and three top levels were used. In second case, four top levels of WordNet hierarchy were used. The term weight in a concept was inversely proportional to a logarithm of concept level.

Tests were executed on AMD Athlon 2200+ with VIA KT-400 chipset and 1GB DDR-333 RAM. The LSI and random projection routines were written in C/C++.

The average precision and recall for 50 TREC queries was calculated for all mentioned methods and classical vector model. The results are summarised in table 1.

Table 1. Precision and recall of 50 executed TREC queries

Method	Precision	Recall
Original sparse term-by-document matrix	79%	74%
rank-100 LSI	74%	100%
rank-200 LSI	74%	100%
rank-200 LSI after RP to dim. 1000	75%	94%
Random projection to dimension 1000	77%	82%
rank-200 LSI of 5961 WordNet concepts	74%	96%
2747 WordNet concepts calculation	73%	100%
502 WordNet concepts calculation	73%	100%

Unfortunately, there are some problems which reduce usability of these results. First, the TREC queries consist of a small number of words, thus they are not usable as document similarity queries. Second, because of the collection size, there were between 1 and 5 relevant documents for each query and 0 to

10 documents which are surly irrelevant. The rest is supposed to be irrelevant but was not checked manually when the queries were created. When we treat these documents as non-relevant, the precision for both LSI and WordNet-based reduction and cosine measure is poor (around 1%).

Because the response times for TREC queries were too short and they did not represent the similarity queries we are mainly focused on, we created a set of 1000 document similarity queries for following tests. The query times for these query set are shown in table 2 together with index size and dimension reduction time. The query times represent an average over 5 test runs.

Table 2. Dimension reduction and query times in seconds; document matrix size

Method	Reduction time	Query time	Index size
Original sparse term-by-document matrix	N/A	33	8,3 MB
rank-100 LSI	2730	6	1,9 MB
rank-200 LSI	3784	11	3,8 MB
rank-200 LSI after RP to dim. 1000	2026	10	3,8 MB
Random projection to dimension 1000	20	44	19,0 MB
rank-200 LSI of 5961 WordNet concepts	3262	10	3,8 MB
2747 WordNet concepts calculation	267	50	16 MB
502 WordNet concepts calculation	227	23	5,15 MB

7 Conclusion

We have shown, that using WordNet ontology instead of random projections offers better results and is even comparable with classical LSI. This could make the dimension reduction feasible even for huge document collections.

The selection of concepts was very rough, we can employ a filtration step and use concepts whose document frequency lies within given borders, which will probably lead to improved precision. We can also filter out several inappropriate meanings of given terms and use other lexical categories like antonyms. Both weighting approach and similarity function may be further modified to provide better precision.

With the use of EuroWordnet's Inter-Lingual-Index for mapping of given concepts, we may be even able to index texts written in different languages.

We are currently studying an interesting application offered by a reversed approach – we try to verify an existing ontology with LSI. If a suitable mapping could be found and formally described, it may become possible to create an ontology on a collection of multimedia documents, e.g. images by calculation of LSI of the collection or its randomly-selected sample [5].

References

1. D. Achlioptas. Database-friendly random projections. In *Symposium on Principles of Database Systems*, 2001.
2. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, New York, 1999.
3. M. Berry and M. Browne. *Understanding Search Engines, Mathematical Modeling and Text Retrieval*. Siam, 1999.
4. M. Berry, S. Dumais, and T. Letsche. Computation Methods for Intelligent Information Access. In *Proceedings of the 1995 ACM/IEEE Supercomputing Conference*, 1995.
5. A. Frieze, R. Kannan, and S. Vempala. Fast Monte-Carlo Algorithms for Finding Low Rank Approximations. In *Proceedings of 1998 FOCS*, pages 370–378, 1998.
6. A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In *Proceedings of ACM SIGMOD 1984, Annual Meeting, Boston, USA*, pages 47–57. ACM Press, June 1984.
7. R. M. Larsen. Lanczos bidiagonalization with partial reorthogonalization. Technical report, University of Aarhus, 1998.
8. G. W. O'Brien. Information Management Tools for Updating an SVD-Encoded Indexing Scheme. Technical Report ut-cs-94-258, The University of Tennessee, Knoxville, USA, December, 1994.
9. C. H. Papadimitriou, H. Tamaki, P. Raghavan, and S. Vempala. Latent semantic indexing: A probabilistic analysis. In *Proocedings of the ACM Conference on Principles of Database Systems (PODS)*, pages 159–168, 1998.
10. G. Salton. *The SMART Retrieval System – Experiments in Automatic Document Processing*. Prentice Hall Inc., Englewood Clifs, 1971.
11. G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
12. G. Salton and M. Lesk. Computer evaluation of indexing and text processing. *Journal of the ACM*, 15(1):8–39, January 1968.
13. G. Salton and G. McGill. *Introduction to Modern Information Retrieval*. McGrawill, 1983.