Leveraging Category-based LSI for Patent Retrieval

Masaki Aono
Toyohashi University of Technology
1-1 Hibarigaoka, Tempaku-cho, Toyohashi, Aichi, 441-8580 JAPAN
aono@ics.tut.ac.jp

Abstract

Latent Semantic Indexing (LSI) has been employed to reduce dimension of indices of documents for similarity search. In this paper, we will describe a method for retrieving conceptually similar patents first by categorizing patent collection and then by applying LSI algorithm multiple times to each category. The main strategy is keeping the algorithm as simple as possible, while achieving the scalability for massive dataset. During the categorization phase, we allow any patent to be classified into multiple categories, which allows patent document overlaps among different categories. Then, for each category, we applied dimensional reduction using LSI to each category into a much lower dimension. Finally, once a query as a collection of claim sentences for a patent is given, we select the most similar category, and return top fifty ranked patent documents as candidates to invalidate the query document.

Keywords: Latent Semantic Indexing, Dimensional Reduction

1. Introduction

We have participated in the patent retrieval task since NTCIR-5 [17]. In the previous NTCIR-5 patent retrieval task , we developed a prototype patent retrieval system based on a hierarchy of clusters by randomly choosing patent collections from a massive patent dataset [1]. Random choice of patent collections was started with randomly selecting one tenth of original entire patent collections (i.e. some 300,000 patents), and then with applying co-clustering algorithms [2] by varying granularity of the number of generated clusters by 16, 32, 64, and so on until 2048, where these numbers represent the number of clusters to be generated by the co-clustering algorithm. Then, we constructed a hierarchical data structure by comparing inter-cluster similarity between two arbitrary clusters with adjacent clusters in the hierarchy. For instance, we compared an arbitrary pair between a cluster generated by 16-cluster granularity and a cluster generated by 32-cluster granularity. There were two big problems in the above approach. The first problem was due to our strategy of the "sampling" because we sampled only one tenth of the whole dataset, thereby losing information in patent data that were not in the sampled subset. The second problem was due to the "clustering", because clustering generally favors major groups, regardless of algorithmic differences.

To overcome these problems, in NTCIR-6 patent retrieval task [16], we have taken a different approach, which we will describe in this paper. Briefly, it is based on a method for retrieving conceptually similar patents by first categorizing the entire patent collection into categories based on IPC (International Patent Classification) [3], followed by applying LSI to each category repeatedly. Since we have not relied on random sampling and clustering, which might have overlooked "minor" clusters for given massive patent data collection, we have improved performance compared to the previous approach.

2. Related Work

The seminal paper by Deerwester et al [4] introduced the techniques "Latent Semantic Indexing" (LSI) to reduce dimension, given a document-by-keyword matrix of large dimension. LSI has been applied to information retrieval, using singular value decomposition (SVD), featuring data compression, keeping major characteristics, and yielding the natural resolution to synonym and polysemy to some extent. Dimensionality reduction per se has been applied not only to information retrieval, but also to many other research areas where there is a need to keep major characteristics with much smaller size than the original object. For instance, in statistical and machine learning applications, dimensional reduction has been used to extract dominant features (e.g. [5]).

Variants of LSI include Probabilistic LSI (PLSI) by Hoffman [6], Differential LSI (DLSI) by Chen et al [7,8], and Locality Preserving Indexing (LPI) by He et al [9]. Even with these sophisticated extensions to LSI, the original LSI still exhibits several advantages over these methods, including mathematical beauty [10], computational straightforwardness implemented by Lanczos method taking advantage of the sparseness of the original problem space [11], and the ability to provide a globally optimal approximation in reduced dimensional space [9].

2. Problem Statement

We briefly looked through latent semantic indexing (LSI) and its typical extensions in the previous section. Although we are given a significant amount of software tools on the Internet for LSI computation based on the Lanczos method, we have to overcome the biggest problem of LSI, i.e. scalability problem. The most typical datasets that LSI has been successfully used include Reuters news dataset at UCI archive [12], which consists of 21,578 documents (news). If we apply an appropriate morphological analyzer or similar language processing tool to extract keywords for each Reuters news data, the amount of (English) keywords is perhaps ranging from 5,000 to 20,000. This is fine because we only need to take care of, say, 21,578 by 20,000 document-by-keyword (*M-by-N*) matrix. The dimension to be reduced by LSI for information retrieval application is usually the keyword dimension. It should be noted, however, that there has been a long debate as to how many dimensions *k* (where $k \ll N$) to reduce to when applying dimensionality reduction [13]. To our best knowledge, only empirically optimal dimension is determined as case by case. Suffice to say that PLSI proposed by Hoffman [6] mentioned a probabilistic approach to determine *k.*

Apart from the discussion on the best dimension *k*, it should be noted that with Reuters dataset, a naïve vector model would requires 21,578-by-20,000 storage for keeping the entire vectors representing each document (news). This is unrealistic because if we keep each element of the matrix in the storage with a double-sized array, we require 3GB or more memory, only with the array itself. The Lanczos method [11] gets around this problem by only keeping the non-zero elements. In other words, it takes advantage of the sparseness of the matrix. Ordinarily only a few percentage of the original matrix has non-zero elements. Suppose, 5 % of the Reuters matrix has non-zero elements (or 95% sparseness), then we need to have 160MB rather than 3GB.

The above observation tells us that given a three and half million patent documents, each having keywords ranging from 100 to 2,000 (on the average, say, 300), even 95% sparseness requires 8.4 GB (3.5M x 300 x 8 (double)) or more storage to keep the data, which is unrealistic for most computers. Even worse, since our keyword extraction program tells us that the total amount of different keywords (mostly general and proper nouns) is close to one million, a simple and naive LSI cannot handle the whole patent collection in a unified matrix, and fails to keep minority information from a group of documents coming from low-frequency keywords when plain dimensional reduction is applied. These observations have motivated us to develop a method to handle the scalability problem of this patent retrieval task from the different point of view.

3. Divide and Conquer

Recall that the bottleneck of LSI is its difficulty in applicability to a large scale problem such as patent search and Web search tasks. Although we cannot predict what kind of query is to be given to a retrieval system, we can pre-compute features of each (patent) document out of three and half million patent collections.

The document model we have adopted is a vector space model. With our vector space model, we can list up all the keywords that can be extracted from patent collections in advance, which amounts to close to one million keywords. From the previous NTCIR task, we learned that a straightforward sampling cannot keep minority information. Yet we also learned that by making the problem size smaller, we can handle the problem with ordinary computer environment. Unlike sampling, we adopted a “divide-and-conquer” approach in this NTCIR-6 patent task to handle a large scale patent collection. Specifically, we used a set of IPC (International Patent Classification) categories attached to each patent, although we realized that 115 patents have no IPC category numbers at all, out of almost three and half million patents. As described in the IPC handbook [3], we have eight big categories (sections) starting alphabetical letters from “A” to “H”. However, simple eight subdivisions wouldn’t solve the problem of LSI. Thus, we further subdivided each category (section) into sub-categories (sub-sections). For instance, “A” section (life necessaries) was subdivided into nineteen subsections. This way we subdivided the patent collection into 200 sub-categories. It should be noted that a single patent usually has one or more IPC numbers, sometimes striding over two or more different big categories. Considering this, we allowed document overlaps when we subdivided the entire patent collection into 200 categories. In the following, we use “categories” instead of sub-categories as long as no confusion may occur. The total amount of patent documents summing up the documents from every category exceeds almost twice of the original patent collection (i.e. almost equal to 7 million). We expect that this overlapping strategy makes it possible to increase “recall” of conceptually similar patent retrieval.

After patent classification based on IPC was made, we applied LSI repeatedly to each category, i.e. 200 times. The number of patent documents in each category ranges from about 2,000 to 300,000. The reduced dimension *k* is empirically set to $k = k_1 = N / 10$ if *N* (number of keywords) is equal to or less than 5,000, $k = k_2 = N / 200$ if *N* is equal to or more than 50,000 and less than 100,000, $k = \alpha k_1 + (1-\alpha) k_2$ if *N* is in-between 5,000 and 50,000, and $k = 500$ if *N* is more than 100,000.

Given a query patent (claim), we first vectorize the claim and compute the similarity between the query vector and 200 category vectors. Once the most similar category is identified, we reduce the query vector by using the *k* singular vectors belonging to the category. Finally, we compute the similarity between the reduced query vector and the document vectors in the category, followed by sorting them in ranked order. As a minor detail, we also keep the PDATE (Patent issue date) and FDATE (Patent file date) for invalidity search task, in order to exclude similar patents which were issued later than the query document. Note that since we have not participated in passage retrieval task, we have never used <PASSAGE> and <PNUM> tags included in each patent. Note also that we keep the original keyword vector for each patent document, to expand the query with the most similar patent document if the number of keywords extracted from <CLAIM> tag is less than a pre-defined threshold.

4. System Overview

In the previous section, we described the overall algorithm of our approach to NTCIR-6 patent retrieval task. Figure 1 illustrates the overall flow of our system based on multiple LSI applications after categorization of patent collection into 200 classes. In Figure 1, we labeled major category from "A" to "H", but in reality, there were sub-categories that straddle between two or more major categories. Examples include "G06F+B41J" category, which is not clearly grasped from Figure 1.

The computational time to compute LSI for all the 200 categories was nearly five days (almost 100 hours). Since we have 200 sub-categories in total, the average LSI computation time was 30 minutes using Pentium 4 of 3 GHz CPU processor with 3GB memory. Here the computational time of LSI includes not only the core LSI algorithm (dimensional reduction using Lanczos method) but also the I/O time to produce and deploy reduced singular vectors for later processing when a query is given, as well as the time to make index for PDATE and FDATE as mentioned in the previous section. All these computations were done in advance.

Once a query for invalidity search task is given, the system follows the bottom flow in Figure 1. The query expansion mentioned in the previous section is omitted in Figure 1 for brevity. It took 30 seconds on the average for each query to compute top 50 similar patents, filtering out similar patents but appeared later than the patent in concern. Since the number of queries was about 3,260, it took almost one day for completing the query processing. There is no specific reason we chose top 50 similar patents, and the number of similar documents to return can be varied easily.

5. Evaluation

The original program we made only produced top 50 relevant documents that satisfy the condition that "issue" dates are predated by "application" date so that they are invalidated the patent to be applied. NTCIR-6 allows participants to output up to top 1,000 relevant documents produced by each system. After deadline, we have added experiments of producing outputs up to 1,000, which slightly increases "recall-precision". In terms of MAP (Mean Average Precision) [18], our results throughout all the tasks range from below 0.001 to 0.6. On the average, for the invalidation tasks where there are multiple relevant documents, they are approximately 0.04 for top 50 (original submitted result) and 0.05 for top 1,000 with our system. It is far best among all the participants. However, we have learned several lessons to be discussed in the next section.

6. Conclusion and Discussion

We described a divide-and-conquer approach to retrieving similar patents from a large-scale patent collection. We used IPC as the subdivision criteria, and classified all the patents into 200 categories, allowing overlaps. For each category, we applied LSI repeatedly for reducing dimension and extracting features.

In NTCIR-5, we employed a clustering approach by constructing a hierarchy of trees (clusters) using so-called "co-clustering" (based on mutual entropy between documents and keywords), which we believe had an academic meaning to challenging the seemingly formidable task for a novel participant in patent retrieval task. On the other hand, in this NTCIR-6, we attempted to conduct a little more practical approach than before. In both approaches, our basic principle has been to confirm whether a vector space model can be used for a practical problem by constructing some data structures in advance.

There are several lessons that we learned from two consecutive patent retrieval tasks, including the necessity to have as complete dictionary as possible when making vectors from each patent document, and the necessity to exploiting the domain knowledge as much as possible. In particular, patent documents are full of special terminologies or keywords not seen in daily life. In this aspect, we begin to consider automatic keyword extraction (e.g. chemical substance names) such as SEQUITUR algorithm [14, 15] to augment user dictionary.

References

[1] Hironori Doi, Yohei Seki, and Masaki Aono, A Patent Retrieval Method Using a Hierarchy of Clusters at TUT, *Proc. Of the Fifth NTCIR Workshop Meeting on Evaluation of Information Access Technologies*, December, 2005.

[2] I. S. Dhillon and Y. Guan, Information Theoretic Clustering of Space Co-Occurrence Data, *Proc. Of the Third IEEE International Conference on Data Mining (ICDM)*, pp. 517-520, November, 2003.

[3] Japan Patent Office, *Handbook on International Patent Classification*, 7th Edition, (in Japanese) 2000.

[4] Scott Deerwester et al. "Indexing by Latent Semantic Analysis", Journal of the American Society for Information Sciences, Vol.41, No.16, pp. 391-407, 1990.

[5] Amir Globerson, Sufficient Dimensionality Reduction, *Journal of Machine Learning Research*, Vol. 3, pp. 1307-1331, March, 2003.

[6] T. Hoffman, Probabilistic Latent Semantic Indexing", *Proc. ACM SIGIR'99*, pp. 50-57, 1999.

[7] L. Chen, N. Tokuda and A. Nagai, Probabilistic Information Retrieval Method Based on Differential Latent Semantic Index Space , *IEICE Trans. on Information and Systems*, E84-D, No. 7, pp. 910-914, 2001.

[8] L. Chen, N. Tokuda and H. Adachi, "A Patent Document Retrieval System Addresses Both Semantic and Syntactic Properties", Proceedings of ACL2003 Workshop on Patent Corpus Processing, pp.1-6, Sapporo, Japan, July 12, 2003

[9] Xiaofei He, Deng Cai, Haifeng Liu, and Wei-Ying Ma, Locality Preserving Indexing for Document Representation, *Proc. ACM SIGIR'04*, pp. 96-103, 2004.

[10] Michael W. Berry and Susan T. Dumais, and Gavin W. O'Brien, Using Linear Algebra for Intelligent Information Retrieval, *Numerical Linear Algebra with Applications,* Vol.3, No.4, pp. 301-328, 1996.

[11] Michael Berry, Theresa Do, Gavin O'Brien, Vijay Krishna, and Sowmini Varadhan, SVDPACKC (Version 1.0) User's Guide*, available at* www.netlib.org/svdpack/*,* April, 1993.

[12] UCI KDD Archive, Reuters-21578 Text Categorization Collection, University of California, Irvine, *available at kdd.ics.uci.edu/*.

[13] Richard K. Belew, *Finding Out About*, Cambridge University Press, 2000.

[14] Satoshi Shirai, Osamu Torii, and Tatsunori Kanai, Keyword Extraction from Documents Based on the String Repetition Acyclic Graph, *DBSJ Letters*, Vol. 4, No.1, pp.1-4, (in Japanese) , 2005.

[15] C.G. Nevill-Manning and I.H. Witten, Identifying Hierarchical Structure in Sequences : A linear-time algorithm, *Journal of Artificial Intelligence Research*, Vol.7, pp.67-82, 1997.

[16] Atsushi Fujii, Makoto Iwayama, and Noriko Kando, "Overview of the Patent Retrieval Task at NTCIR-6 Workshop", *Proceedings of the Sixth NTCIR Workshop Meeting*, 2007.

[17] Atsushi Fujii, Makoto Iwayama, and Noriko Kando, "Overview of Patent Retrieval Task at NTCIR-5", *Proceedings of the Fifth NTCIR Workshop Meeting*, pp.269-277, 2005

[18] Kazuaki Kishida, "Property of Average Precision as Performance Measure for Retrieval Experiment", *IPSJ Transaction of Database*, Vol. 43, No. SIG (TOD 13) 2, March, pp. 11-24, 2002.

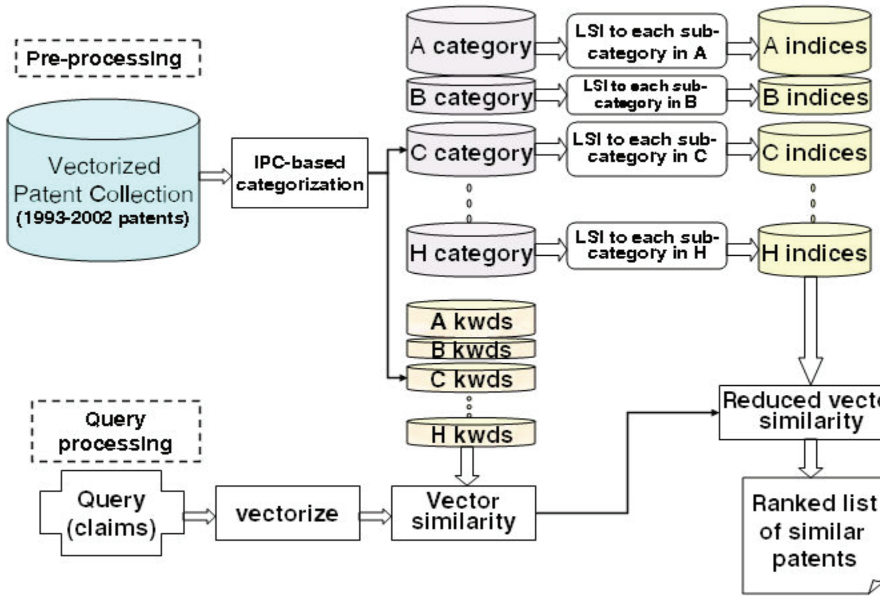
Pre-processing
Vectorized Patent Collection (1993-2002 patents)
IPC-based categorization
A category
B category
C category
H category
LSI to each sub-category in A
LSI to each sub-category in B
LSI to each sub-category in C
LSI to each sub-category in H
A indices
B indices
C indices
H indices
A kwds
B kwds
C kwds
H kwds
Query processing
Query (claims)
vectorize
Vector similarity
Reduced vector similarity
Ranked list of similar patents

Figure 1. System outlook: As "Pre-processing", we first categorize each patent into 200 subcategories, based on IPC. Then LSI is applied repeatedly to each subcategory to produce indices and singular vectors in reduced dimensional space. "Query processing" starts off with making a vector for the query (claim), and compares the vector with pre-computed keywords in 200 subcategories. Once the most similar category is found, the dimension of query vector is reduced to match the dimension of the most similar category. Finally by considering PDATE and FDATE (not delineated in this picture), the system produces the ranked result.