©2010 International Journal of Computer Applications (0975 – 8887)
Volume 1 – No. 9

Cube Index: A Text Index Model for Retrieval and Mining

B. Janet
Research Scholar and Lecturer
Department of Computer Applications
National Institute of Technology, Trichy - 620015

A. V. Reddy
Professor
Department of Computer Applications,
National Institute of Technology, Trichy – 620015

ABSTRACT
Text retrieval, Analysis, Mining and Knowledge management have gained a lot of importance in a time when we drown in information but are starved for knowledge. In this paper, we propose a novel Index that uses a Text Cube model to store the text information similar to a data cube in Data Mining. This model creates a direct index, next word index and inverted index in a single Cube Index which is three dimensional in nature. The Dimensions considered are first word, next word and document. The measure of the cube is the frequency of occurrence of the word next-word pair. The cube index has been tested by modifying the open source of terrier 2.1.

Categories and Subject Descriptors
I.7.3 [Index Generation]: Index creation H.3.1 {Content Analysis and Indexing}: - *Indexing methods*

General Terms
Algorithms, Management, Design.

Keywords
Cube Index, Information Retrieval, Inverted index, Next-word index, Associative index, Direct Index, Text Data cube.

1.INTRODUCTION
There is an explosion in the amount of Text data that has been generated and stored in the recent years. The need to manage and convert the text into knowledge has become the problem of the hour. For efficient retrieval of text a number of index structures have been proposed. An index is a structure that has to be created and maintained and searched according to the query of the user. An index categorizes which document contains which word and speeds up the process of information retrieval considerably. [1] An Index can be created manually or automatically or by a combination of both. There are different types of index structures.

A Direct Index helps in search for a word within a document. An inverted index helps in searching for a word in a query, through a collection of documents [5]. Inverted Index has become a standard for indexing though it requires considerable storage space [1]. A Next-Word Index helps to store phrase information.[1] It reduces the time to retrieve the text by 50% [6].

There is a need for an index structure that can retrieve and also analyze the text information. With the advance in technology, mining of information from unstructured text data is possible with a modification to the index structure already used. A Cube index model has been proposed, which can be used to enrich the query and provide intelligent retrieval.

1.1Motivation
To understand a document, we read the words that occur in them and associate the words to form the meaning that describes the content of the document. When we group similar documents together we group them by the content they share. The content can become meaningful when we store the order in which the words occur. This is possible with a next-word index. To analyze the data we use a data cube with various dimensions in which the data is to be studied. This paper is a combination of both the next-word index and the Text cube formation as a direct and inverted index.

2.CUBE INDEX
2.1Data Cube
In Data Mining, a data cube allows data to be modeled and viewed in multiple dimensions. It is defined by dimensions and facts. Dimensions are perspectives or entities with respect to which one wants to keep records. Facts are numerical measures. Facts are numerical functions computed by aggregating the data corresponding to respective dimension-value pairs defining the given point [2]. Data cube has not been used so far as the textual content could not be quantized. A Text cube was suggested for computing IR Measures for text database analysis on Dell customer review datasets [4].

2.2Cube Index
A Cube Index allows text to be modeled and viewed in multiple dimensions. Dimensions are word associations in the documents represented as Word, Next-word, and Document in which the phrase occurs as in figure. Facts are Term Frequency or any term weighting measure.

[1] A Phrase is two words occurring one after the other. They are also called word – next-word pair

Consider a sample document D1 with the text as given below.

Magnetic Lines of Force acting on earth.

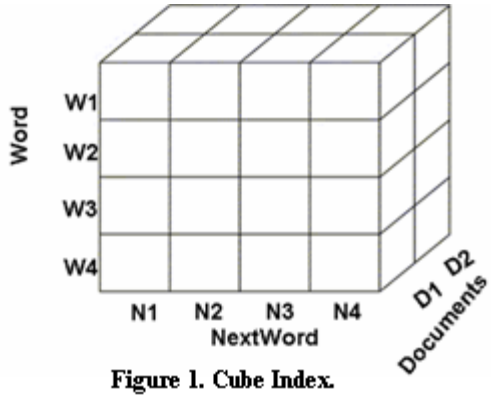

Figure 1. Cube Index.

After it has been preprocessed [1], the content words are considered for cube construction. Let the words considered be Magnetic, Lines, Force, acting and Earth taken as w1, w2, w3, w4 and w5. Now Lines is the next-word n1 for w1 in D1. Force is n2 for w2 in D1. Acting is n3 for w3 in D1. Earth is n4 for w4 in D1. The frequency of occurrence of the phrase is one. So the cube will be as in figure 2.

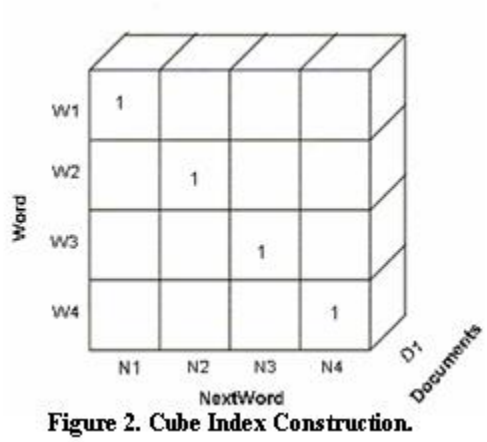

Figure 2. Cube Index Construction.

If we consider all five words to represent a term x term association, then we can construct the cube with five words as one dimension and the same five words as the next dimension. If there is an occurrence of the phrase, then we store as fact, the frequency of the phrase in the cube.

2.3Model Overview

An unstructured text document contains some words that represent the meaning of the document. The Cube model of text is found to be suitable to hold both the term associations and term - document associations for a text document. We consider the three dimensions as the word, next-word and document in the cube. The facts are to store term-frequency or inverse document frequency.

The cube gives the word associations by the word-next-word space and the word-document space gives the term-document associations. The diagonal in text cube index gives the word occurrence frequencies in the document, if the terms are used to represent a term x term matrix.

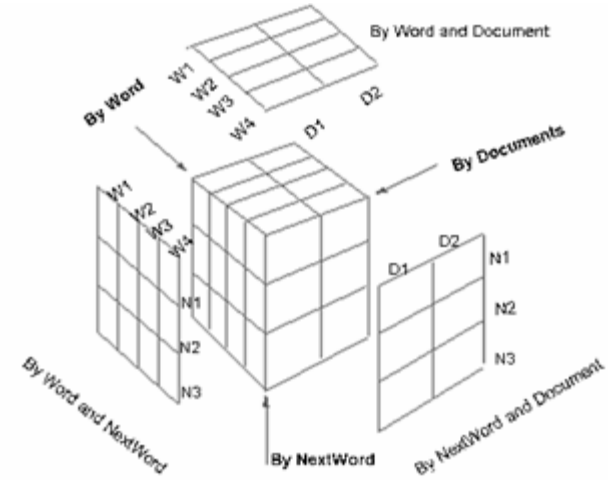

Figure 3. Cube Index Representation

The cube index can also be used to mine association rules between words to enrich the queries by applying the apriori algorithm. The documents are taken as the transactions and the words are taken as the itemsets. The association of the words can generate association rules based on word proximity with some support and confidence for dense documents like abstracts of the document. These association rules can be used to enrich the queries to retrieve results based on user preferences.

2.4Cube Operations

A Concept hierarchy in a data cube defines a sequence of mappings from a set of low level concepts to higher level more general concepts [2]. The concept hierarchy for the dimension word is the phrase, sentence, paragraph or block of text which is considered. This organization provides the user with the flexibility to view the text from different perspectives. The OLAP operations performed on a data cube is applied for a cube index.

2.4.1Roll up

The Roll up operation by aggregation groups the words as a block or paragraph as in figure 4.

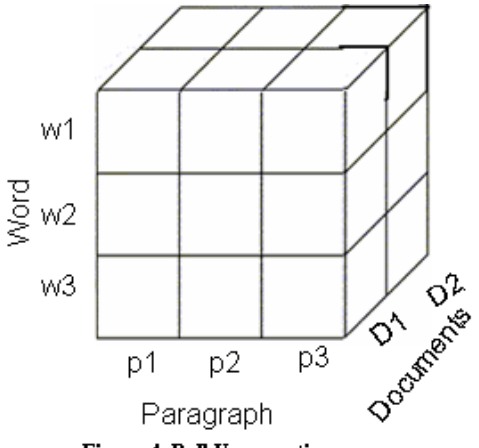

Figure 4. Roll Up operation.

Roll-up by dimension reduction removes one dimension say, next-word and aggregates by document rather than by document and next-word.

2.4.2Drill down

Drill down operation is the reverse of roll-up. It adds more dimension to the data.

2.4.3Slice

Slicing operation selects one dimension of the cube. Slicing by a document gives the word-next-word association for the given document as in figure 5. This is a direct index with next-word information in it.

Slicing by a word gives the term – document relationship. This is an inverted index with next-word information in it.

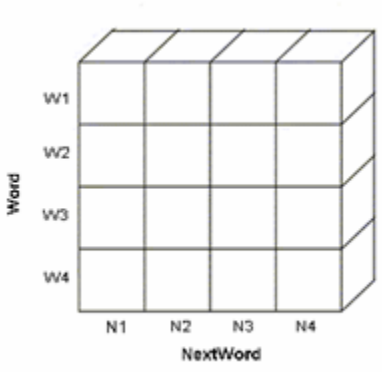

Figure 5. Slicing a cube

2.4.4Dice

The dice operation defines a subcube by performing a selection on two or more dimensions. To evaluate the query, we will have

Figure 6. Dicing a Cube

to dice the cube to get only those words and next-words that form a part of the query as in figure 6. We can do a roll up operation on the cube to include paragraph details and drill it down to retrieve the results.

2.4.5Pivot

Pivot or rotate is a visualization operation that rotates the data axes in view in order to provide an alternative presentation of the text. The Document-word-next-word axes give the direct index of the document collection stored as in figure 7.

This direct index can be pivoted to form an inverted index by rotating the axes as word - next-word -Document axes as in figure 8.

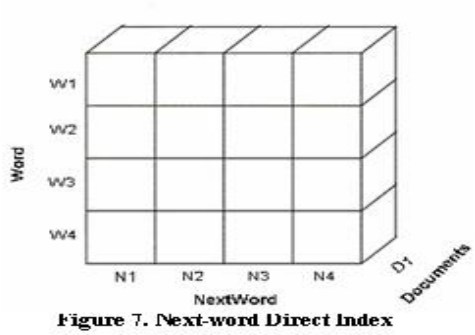

Figure 7. Next-word Direct Index

Various operations can be performed on the index depending on the type of analysis and the need of the application.

The Latent semantic Index structure is obtained in the word – document space that can be used to find the word – document associations.

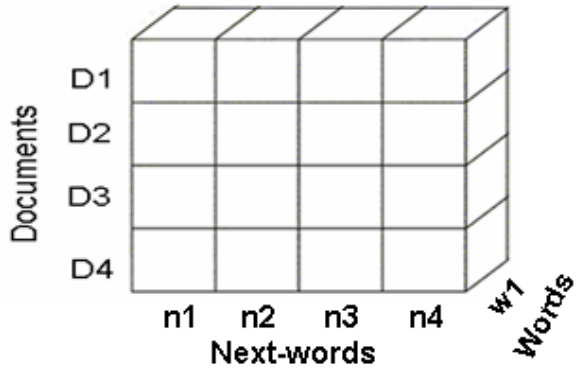

Figure 8. Next-word Inverted Index

2.5Measures for IR and Mining

Various measures for IR have been used [1]. Let N be total number of documents represented,

F_{11} be the number of times the word w_1 appears in document D_1,

n_1 be number of documents in which the word w_1 appears then

- Term Frequency = F_{12}
- Inverse Document Frequency = $\log (N/n_1)$
- Term Weight = $F_{12} * \log (N/n_1)$

In our experiments, the term frequency has been considered as a measure.

Complete materialization of the cube index may not be possible due to memory constraints. So a partial materialization of the cube can be done to suite the user requirement and the complete cube stored in the disk memory.

3.IMPLEMENTATION

Terrier [3] is modified to implement a cube index in Pentium 4 processor with 3.2 GHz and 1 GB RAM.

3.1Terrier 2.1

The cube index has been implemented by modifying the Terrier 2.1 to construct a cube index. Terrier is the Terabyte Retriever version 2.1 developed by University of Glasgow - Department of Computing Science, Information Retrieval Group. The open source version of Terrier is written in Java.

Terrier is modified to implement a cube index. The direct and inverted index is modified to hold the next-word information and stored using the unary compression used in terrier.

3.1.1Lexicon Index

The document collection is preprocessed by stemming and stop listing using the Term Pipeline used in Terrier. Then the words of the document are stored as lexicons in the Lexicon Index, used in Terrier with words arranged in the sorted order. The lexicon index stores the lexicon identifier, Lexicon word, frequency of occurrence of the lexicon in the collection and the word frequency. The lexicon hash is used for fast binary search through the lexicon file.

3.1.2Document Index

The Document Index, used in Terrier stores the document information of the document identifier, Document name, Path and start and end offset bytes in the direct index file.

3.1.3Next-word Direct Index

The word pairs are considered as they occur together in the text document, as a phrase with the word and the next-word forming a phrase. The Next-word Direct Index is constructed by storing the word id, next-word id and the frequency of occurrence of the phrase in one document and the offsets are written into the document index to enable easy retrieval.

3.1.4Cube Index

The word pairs are used to construct the cube index with the dimensions as word id which is the lexicon id in the lexicon index, the next-word id and document id. The cube index stores the word identifier, document identifier, next-word identifier and the frequency information for a phrase. A Cube index is constructed using the next-word direct index and it is pivoted to be stored as the next inverted index structure.

The cube index is represented as a 3-dimensional array. The resulting cube is very sparse. So sparse matrix storage method is used to store in memory and retrieved. The Cube index is stored in a file as a sparse matrix representation that uses unary encoding to store the integer identifiers.

3.2Advantages

3.2.1Index Size Vs Retrieval efficiency

As the next-word information is stored, the size of the index file is much larger than the direct file as shown in table 1 and 2. In a time when memory is becoming cheaper, index size will not be a constraint for creation of the cube index considering the applicability of the index for both analysis and retrieval.

The index is for a phrase. So resolving a query [7] needs 50% lesser processing time as it directly locates the phrase. It is just dicing the cube to suit the query. Thus retrieval time is reduced.

If the size of the index has to be reduced then this cube can be constructed for a limited set of words in the text corpora.

3.2.2Expandable Index

The dimensions of the index can be increased to suite the user needs if it is for a specific query base [4]. A single structure can be used for representing both the direct index, next-word index, and inverted index.

3.2.3Semantic Index

Since it stores the information of the word and the next-word occurring together in the document, it has the semantic information stored in the word next-word space. The Latent Semantic indexing methods can be used on this space to understand the meaning or context of the document or paragraph.

3.2.4Applications

As the index stores the next-word information, it can be used for Phrase completion, Phrase suggestion and Word completion.

It can be used to cluster groups of documents based on their word proximity analysis.

It can help when the user need is browsing documents in a collection rather than information retrieval.

It can also be used to generate association rules from the word associations in the index to enrich user specification. Related documents that do not share the same words but convey the same meaning can be grouped together to enrich the user query.

3.3Limitations

Complete materialization of the cube index is not possible due to memory constraints. Partial materialization of the index to represent a section of the text is attempted. It can be merged to form the full cube.

The number of words can be reduced to include the block information. The cube can be expanded according to the application.

It is feasible for small collections of text like citations and abstracts. The creation of the index can be focused on the usage of the index.

To reduce the size of the index, we can use univocal descriptors to reduce the number of words to represent their common meaning.

3.4Experimental Results

Given below is a comparison between the index sizes of the terrier direct index and the cube index created.

Table 1. Index size and time for Direct file

No .of documents	Number of words	Size in KB	Time is Sec
5	18013	4	20
10	31511	7	38

15	47633	10	59
20	62785	14	84
30	97857	21	128
40	142806	31	180
50	174421	38	208

The time is the time taken to construct the Index file stored in the memory in KB size. The cube index has been constructed as a direct index and pivoted to form the inverted index and the cube materialized for a limited set of words. The direct index structure has been stored in memory for a larger set of words for which the

No .of documents	Number of words	Size in KB	Time is Sec
5	18206	3165	20
10	32280	7575	37
15	48609	11983	59
20	64104	17594	81
30	99943	30510	122
40	145640	49776	181
50	177762	61074	215

Table 2. Index size and time for Cube file cube has not been materialized due to memory constraint.

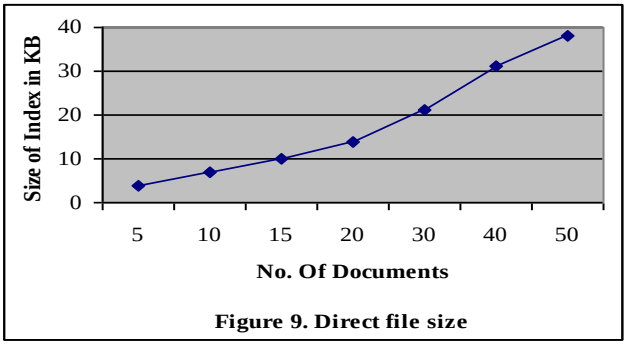

Figure 9. Direct file size

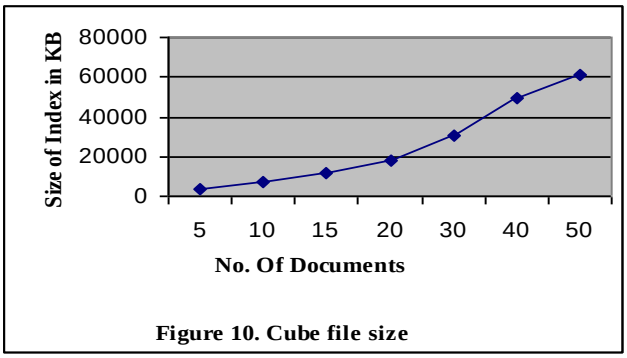

Figure 10. Cube file size

The time taken to create a direct index and a cube index is the same as the same structure is used. When the number of words increases, the size of the file also increases.

The size of the cube index as shown in figure 10 is larger than the direct index due to the next-word information that is stored in the index as shown in figure 9. This size can be reduced by using other compression algorithms.

4.CONCLUSION AND FUTURE WORK

A Cube Index model has been proposed for text retrieval and text mining. The model has been developed using terrier and found to be feasible for an experimental set of text corpora. Experimentally, it is shown that phrase queries can be resolved faster using the cube index by implementing it on the terrier 2.1. Construction of the index is also fast.

The complete model has to be built and tested for various text corpora. The index has to be tested with the standard text database and compared with the existing models for various relevance models. Various applications suggested have to be developed using the cube index.

5.REFERENCES

[1] Frakes W. B. and Yates, R. B. 2008, Information Retrieval Data Structures and Algorithms, Pearson.

[2] Han Jaiwei, Kamber M.,2006 Data Mining Concepts and Techniques, Elsvier, Morgan Kaufmann Publishers.

[3] http://ir.dcs.gla.ac.uk/terrier/

[4] Lin C. X., Ding B., Han J., Zhu F., Zhao B., Text Cube: Computing IR Measures for multidimensional Text Database Analysis, Proceedings of the 8th IEEE International Conference on Data Mining, 2008, http://doi.acm.org/10.1109/ ICDM.2008.135

[5] Salton G. and McGill, M. J., 1983, Introduction to Modern Information Retrieval, McGraw Hill Company.

[6] Williams, H. E., Zobel, J., and Anderson, P. What's next? Index structures for efficient phrase querying. In Proceedings Australasian Database Conference, M. Orlowska, Ed. Springer-Verlag, Auckland, New Zealand, 1999, p141-152.

[7] Bahle D., Williams H.E., and Zobel J., Efficient Phrase Querying with an Auxiliary Index, In Proc. ACM-SIGIR Conf. on Research and Development in Information Retrieval, Tampere, Finland, August 2002, p215-221.