Advances in Computational Sciences and Technology
ISSN 0973-6107 Volume 10, Number 7 (2017) pp. 2079-2094

http://www.ripublication.com

# Unstructured Text: A Multi-Dimensional Analysis and Data Cube Formation

[1]Amol Saxena* and [2]Dr. Meenu Dave

[1, 2] *Department of CSE, Jagannath University, Jaipur-303901, Rajasthan, India*

*(*Corresponding Author)*

**Abstract**

Today, unstructured data like texts, documents, or social network service (SNS) messages has been increasingly used in many applications, rather than structured data that consist of simple numbers or characters. Thus it becomes more important to analyze unstructured text data to extract important and useful information for user's decision making. As OLAP (On-Line Analytical Processing) analysis is used over structured data, Multi-dimensional analysis is more commonly used for unstructured data. In this paper, for analysis on unstructured data, a text cube model on multi dimensional text database has been proposed. The existing text cube model has been extended to incorporate TF-IDF and Cosine Similarity as measurements. The proposed text cube model utilizes these new measurements which are popular in information retrieval systems to effectively analyze the unstructured text data. Through experiments, it has been observed that the proposed text cube model can be efficiently utilized for multidimensional analysis of unstructured text data and is very useful for user's decision making.

**Index Terms**—OLAP; Multi-dimensional analysis; Text cube; Data cube; Text databases; Information Retrieval (IR); TF-IDF, cosine similarity, document ranking.

## I. INTRODUCTION

Data cube has been widely used to analyze conventional structured data in various ways to get useful information for decision making [1]. This data cube is an essential structure for multi-dimensional analysis in OLAP (On-Line Analytical Processing). In recent days, however, unstructured data such as texts, web pages, or messages have been exponentially increasing more and more, rather than formatted structured data [2]. Various online-sites or shopping malls, e-mail systems, and SNS (Social Network Service) systems produce a bunch of such unstructured data every day [3].

Thus, there is an increasing demand for the analysis on these unstructured data [4]. To support this analysis requirement, we need an effective method for multi-dimensional analysis over unstructured text data. Text Cube [5, 6] was proposed as an efficient model for multi-dimensional analysis on text databases. While conventional data cube has the aggregate values (e.g., sum, average, count, etc.) for numeric data as its measurements, the text cube maintains the measurements by aggregating text documents, such as TF (Term Frequency) and IV (Inverted Index), which are popularly used to represent the features for the text documents.

To facilitate various analyses for the features of text databases, the text cube provides rollup and drill-down operations over multiple dimensions. The text cube can efficiently provide keyword searches from a large set of texts under diverse conditions on multiple dimensions.

The text cube [7,8,9], has a weak point that the used IV values within its measurements are complex to calculate and need more storage spaces.

In this work, we have extended the existing text cube model i. e. TF-IV (Term Frequency – Inverted Index) by using new measures such as TF (Term Frequency)-IDF (Invert Document Frequency) and cosine similarity model which are more effective in capturing the features of texts. The cosine similarity measure which is vector space model is used to find best matched documents against the user query in question. Vectors deals only with numbers and we are dealing with text documents here, so we use TF and IDF to convert text into numbers so that it can be represented by a vector.

## II. TEXT CUBE MODEL

### A. Text Database

Text databases consist of tables containing text-type columns in addition to conventional attributes (i.e., columns) of basic data types such as integer, real, or characters. Table 1 shows an example of a text database table. This example table has 3 attributes such as S, Q, H, and D. Among them, S, Q, and H are dimension attributes and D is a text-type attribute containing several words or statements. The text attribute D is defined as D = {d1, d2, · · · , dn}, where di represents a text document in D. The set of words which appear within the whole set of documents, W, is defined as W = {w1, w2, · · · , wm}. Here, a document $d_i$ is a multiset of words in W.

A set of documents D is stored in an n-dimensional database DB = ($A_1$;$A_2$; . . . ;$A_n$; D). Each row of DB is in the form of r = ($a_1$; $a_2$; . . . ; $a_n$; d): let r[$A_i$] = $a_i$ ∈ $A_i$ be the value of attribute (or dimension) Ai, and r[D] = d be the document in this row.

**Table 1:** A 3-dimensional text database

| **Dimensions** | | | **Text Data** |
|---|---|---|---|
| S | Q | H | Document - D |
| s1 | q1 | h1 | d1={w1, w2, w3,w5, w7} |
| s1 | q1 | h2 | d2={w2, w2, w5, w7, w8} |
| s1 | q2 | h1 | d3={w3, w4, w5, w6, w8} |
| s1 | q2 | h2 | d4={w3, w5, w6, w6,w7,w8,w9} |
| s2 | q1 | h3 | d5={w4, w6, w7, w8, w9} |
| s2 | q2 | h1 | d6={w1, w3, w4,w7, w8, w9} |
| s2 | q2 | h2 | d7={w4, w5, w7, w8, w9} |

The data cube model extended to the above multidimensional text database is called text cube [6]. Table 2 shows some cells in the text cube generated from DB. In the text cube built on DB, a cell is in the form of C =($v_1$, $v_2$, . . . , $v_n$ : D), where either $v_i$ ∈ $A_i$ (a value of dimension Ai) or $v_i$ = * (the dimension $A_i$ is aggregated in C), and D (the aggregated document in C) is the set of documents in the rows of DB having the same dimension values as C on all the non-* dimensions. Formally, for a cell C = ($v_1$, $v_2$, . . . , $v_n$ : D),

D={r[D] | for r ∈ DB, r[$A_i$]= $v_i$ if $v_i \neq *$}

**Table 2:** Some cells in the text cube

| **Dimensions** | | | | **Text Data** |
|---|---|---|---|---|
| Cell | S | Q | H | Document (D) |
| C0 | * | * | * | {d1, d2, d3, d4, d5, d6, d7} |
| C1 | s1 | * | * | { d1, d2, d3, d4} |
| C2 | s2 | * | * | {d5, d6, d7} |
| C3 | s1 | q1 | * | {d1, d2} |
| C4 | s1 | * | h1 | {d1, d3} |
| C5 | s1 | q1 | h1 | {d1} |
| C6 | s2 | * | h1 | {d6} |
| C7 | s2 | q2 | * | {d6, d7} |
| C8 | s2 | q2 | h1 | {d6} |
| C9 | s2 | q1 | h2 | null (empty cell) |
| C10 | * | q2 | h3 | null (empty cell) |

Here, C[Ai] is used to denote the value vi of dimension Ai in the cell C, and C[D] to denote the aggregated document D of the cell C. All the cells with the same set of non-* dimensions form a cuboid. A cuboid with m non-* dimensions is an m-dimensional cuboid. The n-dimensional cuboid (with no aggregated dimension) is called the base cuboid.

## B. Measures supported for Information Retrieval (IR)

a. **Term Frequency:** Term Frequency $\mathbf{tf_{t,d}}$ of term t in document d is defined as the number of times that t occurs in d [10].

b. **Normalized tf: tf** count is normalized to prevent a bias towards longer documents (which may have a higher term count regardless of the actual importance of that term in the document) to give a measure of the importance of the term $t_i$ within the particular document $d_j$ [10].

$$\mathrm{tf}_{\mathrm{i,j}} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

where $n_{i,j}$ is the number of occurrences of the considered term ($t_i$) in document $d_j$ , and the denominator is the sum of number of occurrences of all terms in document $d_j$ , that is, the size of the document $|d_j|$.

**c. Inverse Document Frequency:** Estimates the rarity of a term in the whole document collection. (If a term occurs in all the documents of the collection, its IDF is zero.)

The total number of documents in a collection by N, we define the inverse document frequency (idf) of a term t as follows:

$$\mathrm{idf}_t = \log \frac{N}{\mathrm{df}_t}.$$

Thus the idf of a rare term is high, whereas the idf of a frequent term is likely to be low [10].

**d. TF-IDF (Term Frequency Inverse Document Frequency)**

The tf-idf weight of a term is the product of its tf weight and its idf weight [10].

$$\text{tf-idf}_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t.$$

**e. Vector Space Model – Cosine Similarity**

The tf-idf values are used to create vector representations of documents. Each component of a vector corresponds to the tf-idf value of a particular term in the corpus dictionary. Those dictionary terms that do not occur in a document are weighted zero. Using this kind of representation in a common vector space is called vector space model [12, 13], which is essential to a host of information retrieval operations ranging from scoring documents on a query, document classification and document clustering.

Because of the difference in lengths of documents, simply computing the difference between two vectors has the disadvantage that documents of similar content but different length are not regarded as similar in the vector space. To avoid the bias caused by different document lengths, a common way to compute the similarity of two documents is using the cosine similarity measure.

Cosine similarity is a measure of similarity between two non zero vectors of an inner product space that measures the cosine of the angle between them.

The cosine of 0° is 1, and it is less than 1 for any other angle. The cosine is thus a judgment of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors at 90° have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude.

How vectors are normalized to unit lengths is explained next. The inner product of the two vectors that means sum of the pairwise multiplied elements is divided by the product of their vector lengths. This has the effect that the vectors are normalized to unit length and only the angle, more precisely the cosine of the angle, between the vectors accounts for their similarity.

Documents not sharing a single word get assigned a similarity value of zero because of the orthogonality of their vectors while documents sharing a similar vocabulary get higher values (up to one in the case of identical documents). Using the formula given below we can find out the similarity between any two documents.

**Cosine Similarity (d1, d2) = Dot product(d1, d2) / ||d1|| * ||d2||**

Dot product (d1,d2) = d1[0] * d2[0] + d1[1] * d2[1] * … * d1[n] * d2[n]

||d1|| = square root(d1[0]2 + d1[1]2 + ... + d1[n]2)

||d2|| = square root(d2[0]2 + d2[1]2 + ... + d2[n]2)

## III. PROPOSED TEXT CUBE MODEL

Users expect the result matching their search intent. OLAP based multi-dimensional text data analysis with IR model applies to the search using particular values of the text data which are calculated by the aggregate values of the text data according to the combination of dimensions. While the existing text cube model was used to measure two kinds of text measurements, TF and IV (Inverted Index), the proposed text cube model extends the existing text cube model to incorporate TF-IDF (Term Frequency Inverse Document Frequency) and Vector Space -Cosine Similarity model as measurements to find similarity between two documents. Figure 1 shows the overall architecture of the proposed text cube model. In the figure, the new model is stored in the text database through the extraction and parsing of text data. It also performs the computation of text cubes from text databases. With the text cubes, users can process multi-dimensional keyword searches. For example, let’s suppose that a user gives a phrase or keywords k2, k4 and dimension conditions d2, d4 as an input query. S/he will get the cuboids corresponding to the dimensions d2, d4 which contain the keywords k2, k4 as the query result.

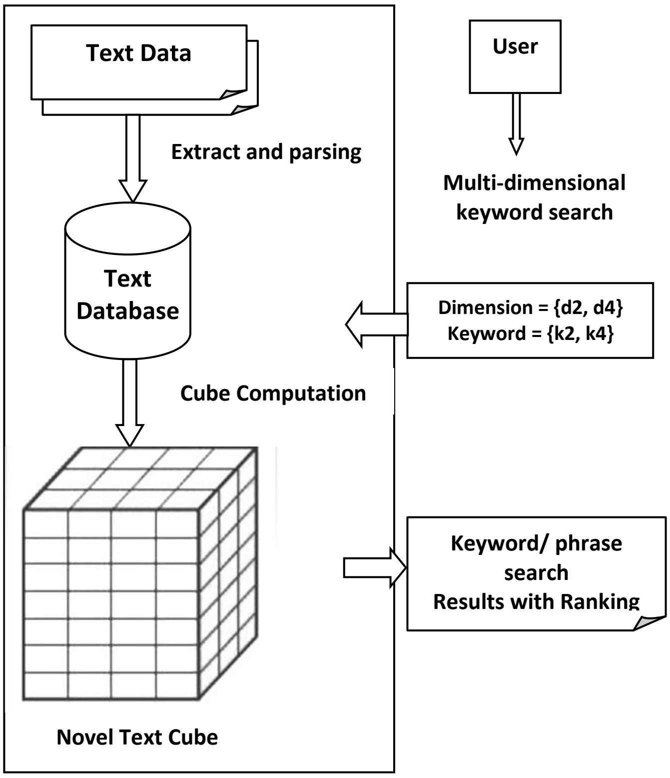


**Figure 1:** Overview of proposed Text Cube model

## IV. OVERALL PROCESS/ PHASES TO CREATE NEW TEXT CUBE MODEL

### 1. Extracting and Parsing of text data

In this step the text data (such as user reviews about some products or services) is organized in text database. The example of text database is given above in Table-1.

### 2. Computation of text cubes from text databases

With the text cubes, users can process multi-dimensional keyword searches. For example, let us suppose that a user gives keywords k1, k2 and dimension conditions d3, d5 as an input query. The user will get the cuboids corresponding to the dimensions d3, d5 which contain the keywords k1, k2 as the query result. The next step explains how we apply the measures for IR to find out relevant documents (such as user reviews about some products or services) from a set of documents in the searched cuboid.

3. **Measure for IR (Information Retrieval)**

a. **TF-IDF (Term Frequency-Inverted Document Frequency)**

TF-IDF (Term Frequency-Inverted Document Frequency) is the popular weighting model in information retrieval and text mining areas. TF-IDF is used as a statistical value or a measure to indicate how important any word in a particular document when there is a set of documents is to be analyzed. The proposed text cube uses the aggregate values of TF, IDF, and TF-IDF corresponding to the combination of dimensions as per the user query. A document consists of multiple sentences and each sentence contains several words. The text cube can be computed by measuring the frequency of the words in the documents when users aggregate the documents depending on the combination of dimensions. TF represents the frequency of the keywords which are capable of determining the priority value for the keywords of users query. IDF represents the inverse of DF (Document Frequency). TF-IDF weighting produce a composite weight for each term in each document.

b. **Cosine similarity measure**

Because a query can be considered a short document, it is of course possible to create a vector for the query, which can then be used to calculate the cosine similarities between the query vector and those of the matching documents in the cuboid. Finally, the similarity values between query and the retrieved documents are used to rank the results.

4. **Query search result with ranking**

Basically the purpose of information retrieval is to return the resulting documents well-matched to user's query. The user will get the cuboids corresponding to the dimensions d2, d4 which contain the keywords k2, k4 as the query result. The result will contain the similarity values between query and the retrieved documents are used to rank the results.

## V. IMPLEMENTATION OF NOVEL TEXT CUBE MODEL

In this work, our experiments used a computer with Intel Core i3-2350 CPU 2.30 GHz, 2GB memory and the operating system is Windows 7, 64 bits. We used Microsoft SQL Server 2008 SP1 database server. The Query processing module of the proposed text cube model was implemented using the C# language on the .NET framework.

The following figure 2 shows an example of a text cube model with the concept of dimension hierarchies.

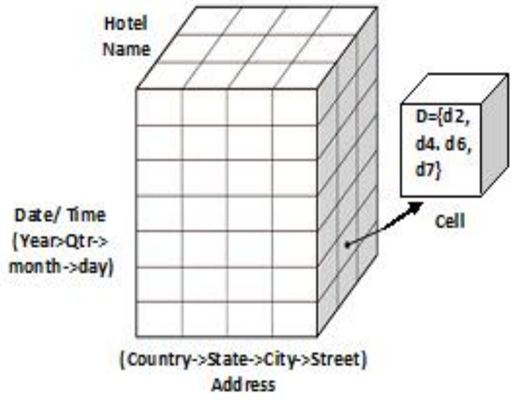


**Figure 2:** Example of the implemented Text Cube Model

Data cube which consists of multiple dimensions determines the analysis point of view according to the choice of dimensions. In other words, measurements are calculated according to the values of dimensions. Also, each dimension can have a concept hierarchy. In this paper, we used three dimensions such as Hotel_Name, Date/Time, and Address as an example. The Hotel_Name dimension does not have a hierarchy, but Date/Time and Address dimensions have hierarchies. The hierarchy of the Date/Time dimension is year → quarter→ month → day and the hierarchy of the Address dimension is country → state → city→street.

The above text cube is created with the help of the following star schema diagram.

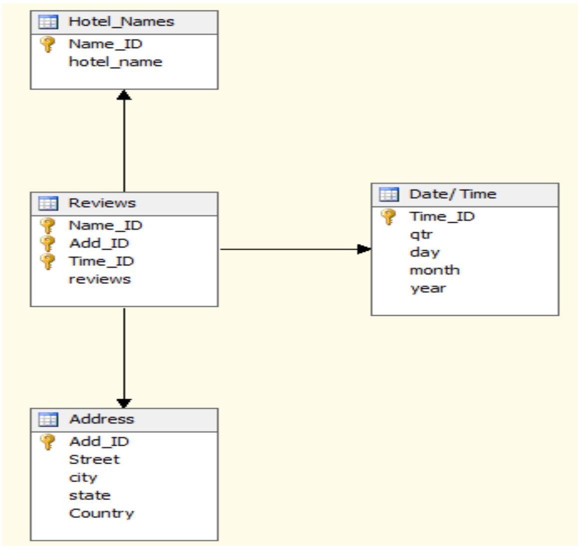


**Figure 3:** Star Schema for the proposed Text Cube Model

The following example illustrates the process of rollup operation on the multi-dimensional data cube. Here aggregation is performed on documents (i. e. hotel reviews) and as a result, multiple documents are combined corresponding to the dimensions in the query.

In the below given table three dimensions are shown – State, Quarter and Hotel_Name with reviews as fact or measures. As can be seen in the star schema diagram of Figure 3 the fact table stores the actual reviews on different hotels as text documents. For the sake of simplicity only the count of reviews corresponding to different dimensions are shown below in the tables instead of textual reviews.

**Table 3:** Hotel reviews data according to the three dimensions State, Quarter and Hotel_Name

| State | Quarter | Hotel Name | Number of Records (Reviews) |
|---|---|---|---|
| Delhi | q2 | Casablanca | 1 |
| Himachal Pradesh | q1 | Holiday Inn | 1 |
| Karnataka | q1 | Hotel Giraffe | 2 |
| Karnataka | q2 | Hotel Giraffe | 1 |
| Madhya Pradesh | q1 | The Beekman, A Thompson Hotel | 1 |
| Madhya Pradesh | q2 | Holiday Inn | 1 |
| Madhya Pradesh | q2 | The Beekman, A Thompson Hotel | 2 |
| Maharashtra | q1 | Wellington Hotel | 1 |
| Maharashtra | q2 | Casablanca | 1 |
| Rajasthan | q1 | Casablanca | 1 |
| Rajasthan | q1 | Wellington Hotel | 1 |

The Table 3 shows that there are two reviews corresponding to the State - Karnataka, Quarter - q1 and Hotel_Name – ‘Hotel Giraffe’. Similarly there are two reviews for the Sate - Madhya Pradesh, Quarter – q2 and Hotel_Name – ‘The Beekman, A Thompson Hotel’ as a result of aggregation.

Now we drop the hotel name dimension to perform the rollup operation. The above data will be aggregated on State and Quarter fields.

**Table 4:** Aggregated data of hotel reviews as per the dimensions State and Quarter

| State | Quarter | Number of Records |
|---|---|---|
| Himachal Pradesh | q1 | 1 |
| Karnataka | q1 | 2 |
| Madhya Pradesh | q1 | 1 |
| Maharashtra | q1 | 1 |
| Rajasthan | q1 | 2 |
| Delhi | q2 | 1 |
| Karnataka | q2 | 1 |
| Madhya Pradesh | q2 | 3 |
| Maharashtra | q2 | 1 |

As a result of rollup operation, it can be seen in the Table 4 that there are three records of hotel reviews in the State Madhya Pradesh and in Quarter – q2.

Now again we drop the quarter dimension to perform the rollup operation further on the above table. The above data will be aggregated on State field only.

**Table 5:** Aggregated data of hotel reviews as per the dimensions State only (rollup operation).

| Sr. No | State | Number of Record |
|---|---|---|
| 1 | Delhi | 1 |
| 2 | Himachal Pradesh | 1 |
| 3 | Karnataka | 3 |
| 4 | Madhya Pradesh | 4 |
| 5 | Maharashtra | 2 |
| 6 | Rajasthan | 2 |

It can be observed from Table 5 that as a result of aggregation (rollup operation) there are four reviews on hotels in the state of Madhya Pradesh which include all hotels and all four quarters. Here the dimensions Quarter and Hotel_Name have been dropped to perform the rollup operation. Therefore, the corresponding cell in the text cube (multidimensional structure) contains four review documents which belong to any hotel of the state of Madhya Pradesh and the reviews can be of any of the four quarters of time.

## VI. EXPERIMENTAL RESULTS

The purpose of this new text cube model is to return the resulting documents well-matched to user's query (information retrieval). With the text cubes, users can process multi-dimensional keyword searches. For example, let's suppose that a user gives keywords k2, k4 and dimension conditions d2, d4 as an input query. The user will get the cuboids corresponding to the dimensions d2, d4 which contain the keywords k2, k4 as the query result. The result will contain the similarity values between query and the retrieved documents are used to rank the results.

In this paper, we are using data of customer reviews about different hotels. We have used the data of traveling in Trip Advisor [14] to evaluate the proposed text cube as a large-scale real text data. The data of customer reviews is being stored in the form of star schema as shown in figure 3.

Now, we present the experimental results on the basis of cosine similarity measures computed on query and aggregated hotel reviews data as per the multi-dimensional analysis of OLAP queries. Here we are analyzing unstructured data using a text cube model on multi dimensional text database.

Different examples have been taken to explain the processing of multidimensional searches. The experimental results shows the best matched ranked documents (customer reviews) as per the user queries.

**Query 1**

Suppose a user wants to see the best matched customer reviews which have the phrase – "**Great for business travelers**".

**Other multidimensional constraints required are as follows –**

State = 'Karnataka' or 'Rajasthan' and

Time = q1 (Quarter-1)

Hotel names= 'Hotel Giraffe' or 'Casablanca'

The above query will give the following tuples as a result.

**Table 6:** Result of Query 1

| **Hotel_Name** | **Review** |
|---|---|
| Hotel Giraffe | Amazing property. First-class experience. Good for travelers. |
| Casablanca | Great Hotel- very friendly crew. The Chinese restaurant is really awesome and highly recommendable. It has a spa, pool and gym in the basement. It's located closely to the airport and to the exhibition center-which makes it great for business travelers. |
| Hotel Giraffe | It was a very busy hotel. Delightful Experience |

The Table 6 shows that there are three reviews corresponding to the query 1 which says that hotel reviews can belong to any of hotel 'Hotel Giraffe' or 'Casablanca' that are located in any of the state either 'Karnataka' or 'Rajasthan' and time quarter must be q1.

Now we have to find those documents/ reviews which best matches the keywords – **"Great for business travelers"**. To find out those reviews we will use the cosine similarity factor.

The result of cosine similarity is shown in the following table.

**Table 7:** Best matched result (reviews) corresponding to query 1

| **Document Number** | **Cosine Similarity** |
|---|---|
| 1 | .556 |
| 2 | 1 |

The Table 7 depicts that document (or review) 2 has the highest score of 1, it has all the keywords of the query.

**Query 2**

Suppose a user wants to see the best matched customer reviews which have phrase – "**perfect room**".

**Other multidimensional constraints required are as follows –**

State = 'Madhya Pradesh' and

Time = q2 (Quarter-2)

The above query will give the following tuples as a result.

**Table 8:** Result of Query 2

| **Hotel Name** | **Reviews** |
|---|---|
| Holiday Inn | Close to everything. Amazing location! Close to everything: Penn station, Macy's, many shopping places. |
| The Beekman, A Thompson Hotel | Fabulous staff!! |
| The Beekman, A Thompson Hotel | We loved our stay at the Casablanca. The room was perfect. The breakfast in the morning was very nice. |

Now we have to find those documents/ reviews which best matches the keywords – "**Perfect room"**. To find out those reviews we will use the cosine similarity factor.

The result of cosine similarity is shown in the following table.

**Table 9:** Best matched result (reviews) corresponding to query 2

| **Document Number** | **Cosine Similarity** |
|---|---|
| 3 | 1 |

We can see as shown in Table 9 that from above three review documents the third review best matches with the query.

**Query 3:**

Suppose a user wants to see the best matched customer reviews which have phrase – "**Amazing location"**.

**Other multidimensional constraints required are as follows –**

State = 'Madhya Pradesh'

The above query will give the following tuples as a result.

**Table 10:** Result of Query 3

| **Hotel_Name** | **Review** |
|---|---|
| The Beekman, A Thompson Hotel | Great Hotel Property. I tried this downtown jewel because of their reasonable price and location during my last trip to NYC. I didn't realize it was a recently renovated historical building and was very pleasantly surprised by their exceptional service and appointments. |
| Holiday Inn | I stayed here during our vacations for 2 days. Great welcome and hospitality i got from staff. Room was very nice with very nice deco. Food was also well cooked and great. This hotel also had a nice location, it is situated central of the city. |

| | |
|---|---|
| The Beekman, A Thompson Hotel | We loved our stay at the Casablanca. The room was perfect. The breakfast in the morning was very nice. |
| The Beekman, A Thompson Hotel | Fabulous staff. Amazing location |

Now we have to find those documents/ reviews which best matches the keywords – "**Amazing location"**. To find out those reviews we will use the cosine similarity factor.

**Table 11:** Best matched result (reviews) corresponding to query 3

| Document Number | Cosine Similarity |
|---|---|
| 1 | 0.474 |
| 2 | 0.474 |
| 4 | 1 |

We can see from the Table 11 that documents (or reviews) 1 and 2 have the highest score of 1 because they contain all the keywords of the query.

## VII. CONCLUSION

In this work, we have explored a new text cube model which can be used efficiently for multi-dimensional analysis and search over unstructured text data. The text cubes maintain the aggregates of text features (e.g., term frequency) for a set of documents according to several dimensions. With the help of the text cubes, users can capture dynamically the overall features of the documents according to dimensions' conditions.

This model uses TF-IDF rather than IV (Inverted Index) which is the most popular measures for texts being used by information retrieval systems. We have then used vector space model and the cosine similarity measure for relevance score calculation between the user query and documents (in the cuboids) being searched. Finally we get the ranked list of relevant documents as an output of the query. Thus, with the text cubes, users can process multi-dimensional keyword searches efficiently.

## REFERENCES:

[1.] J. Gray, A. Bosworth, A. Layman, H. Pirahesh, "Datacube: A relational aggregation operator generalizing group by, cross-tab, and sub-total", In ICDE, 1996.

[2.] Scaffidi, C., Bierhoff, K., Chang, E., Felker,M., Ng, H. and Jin, C., "Red Opal

:Product Feature Scoring from Reviews", In Proceedings of the 8th ACM conference on Electronic Commerce, 2007.

[3.] Adrien Guille, Hakim Hacid, Ccile Favre, Djamel A. Zighed, "Information Diffusion in Online Social Networks: A Survey", in SIGMOD Record, June 2013 (Vol. 42, No. 2).

[4.] Qingliang Miao, Shu Zhang, Yao Meng, Hao Yu Fujitsu R& D Center Co., LTD, "Domain-sensitive Opinion Leader Mining from Online Review Communities", In WWW 2013 Companion, May 1317, 2013.

[5.] Nenad Jukic, Boris Jukic, Mary Malliaris, "Online Analytical Processing (OLAP) for Decision Support", International Handbooks Information System, pp 259-276, 2008.

[6.] X. Lin, B. Ding, J. Han, F. Zhu, and B. Zhao, "Text cube: Computing IR measures for multidimensional text database analysis", in ICDM, 2008.

[7.] Yintao Yu, Cindy X. Lin, Yuzhou Sun, Chen Chen, Jianwei Han, Binbin Liao, Tianyi Wu, ChengXiang Zhai, Duo Zhang, Bo Zhao, "iNextCube: Information Network-Enhanced Text Cube", In VLDB, 2009.

[8.] Bolin Ding, Bo Zhao, Cindy Xide Lin, Jiawei Han, Chengxiang Zhai, "TopCells: Keyword-Based Search of Top-k Aggregated Documents in Text Cube", In ICDE, 2010.

[9.] S. E. Robertson, S. Walker, and M. Hancock-Beaulieu, Okapi at trec-7: Automatic ad hoc, filtering, vlc and interactive, in TREC, 1998.

[10.] Christopher D.Manning, Prabhakar Raghavan, and Hinrich Schutze, "An Introduction to Information Retrieval", Cambridge University Press, pp 6-10, pp 117-123 2009.

[11.] Suan Lee, Namsoo Kim and Jinho Kim, "A Multi-Dimensional Analysis and Data Cube for Unstructured Text and Social Media", IEEE Fourth International Conference on Big Data and Cloud Computing, DOI 10.1109/BDCloud.2014.117.

[12.] G. Salton, A. Wong, and C. S. Yang (1975), "A Vector Space Model for Automatic Indexing", Communications of the ACM, vol. 18, nr. 11, pp. 613-620.

[13.] C. D. Manning, P. Raghavan and H. Schütze (2008), "Introduction to Information Retrieval", Cambridge University Press.

[14.] TripAdvisor, http://www.tripadvisor.com