Retrieval for Text Stream by Random Projection

Hirohito OH'UCHI   Takao MIURA
Dept.of Elect.&Elect. Engr., HOSEI University
3-7-5 Kajino-Cho, Koganei, Tokyo 184–8584 JAPAN
{i03r3208, miurat}@k.hosei.ac.jp

Isamu SHIOYA
Department of Management and Informatics, SANNO University
1573, Kamikasuya, Isehara, Kanagawa 259–1197 JAPAN
shioya@mi.sanno.ac.jp

Abstract: In this investigation we discuss powerful yet efficient retrieval mechanism for text stream such as news stream. Difficulty comes from the fact how to manage incremental information while keeping efficiency. Recently *random projection* has been paid much attention on dynamic dimensionality reduction. Here we show this novel technique is really useful for querying text stream in terms of cost and accuracy. We examine some experimental results and excellent efficiency in computation and memory usage.

1 Motivation

Recently much attention has been paid on *text stream* along with time axis, called *Topic Detection and Tracking* (TDT), in Information Retrieval activities. We obtain a huge amount of documents along with time axis, store and retrieve them. Here we face to new kinds of difficulties that have been never attacked.

When processing text stream, we receive part of the stream in an *incremental* and *unlimited* manner. Thus we couldn't manage all the information without any summarization. Also sometimes we should examine the stream efficiently in a form of queries while receiving and storing them.

Generally we have evaluated queries to a collection of documents by extracting important *words* and specifying vectors to the documents based on Vector Space Model[GF98, Ki02]. In this approach, very often we get high dimensionality of the vectors, say 10,000 to 100,000 words in ten thousands of documents. Clearly it would take much amount of time and space to manage them without any works thus it is hard to examine the stream information. One of the main issues here is how we obtain dimensionality reduction against input-stream.

There have been many techniques proposed and, among others, *Latent Semantic Indexing* (LSI) has been paid much attention[De90, Oh03, Pa98]. This technique comes from Singular Value Decomposition (SVD) in linear approximation, and provides us with dramatic reduction (efficient execution) yet excellent approximation (precise results) to queries.

However, whenever changes arise in data collection, the technique requires recalculation of SVD from scratch. Since it takes much time, it is not easy to apply the technique to text stream. We can apply *folding-in*[BDO95] to avoid the recalculation, but the accuracy decreases because this approach comes from *sampling* and assumes small changes.

In this investigation we propose a framework of information retrieval to text stream based on *Random Projection* (RP)[Pa98]. Basically RP is a technique of dimension reduction, especially RP can be obtained very quickly yet the recalculation is not necessary to any changes. These properties show that RP allows us to apply dimension reduction dynamically and to process queries in a real-time manner. Also we show RP provides us with efficient (in both time and space) and precise results.

There have been *active* investigation about RP so far. Among others, there exists some comparison with RP and LSI to the dimension reduction of text data and image data[BM01] The work says about excellent accuracy result of RP with less amount of computation complexity. However, the evaluation has been made based on data distortion/distribution caused by dimension reduction but not on information retrieval. No consideration has been discussed to data stream.

This paper is organized as follows. In section 2 we review RP and LSI as dimension reduction techniques very quickly and we give some comparison. Section 3 contains our proposal of information retrieval to text stream based on RP, while in section 4 we show some experimental results. Finally we conclude our work in section 5.

2 Dimensionality Reduction of Text Data

In this section we quickly review RP and LSI techniques by which we can reduce dimensionality of text information. In the following let X be a matrix of $d \times N$ where d means a number of words and N a number of documents. Each column corresponds to one document and (i, j) element in X, denoted by X_{ij}, means i-th word appears X_{ij} times in j-th document. X is called a *Term Frequency* (TF) matrix.

2.1 Latent Semantic Indexing

Latent Semantic Indexing (LSI) is a technique of dimension reduction based on *Singular Value Decomposition* (SVD).

It is possible to show that a TF matrix X is decomposed into 3 matrices U, S, V in such a way that:

$$X_{d\times N} = U_{d\times r} S_{r\times r} V^T_{r\times N} \tag{1}$$

where two matrices U, V are orthogonal[1], called left-singular and right-singular respectively. Each row vector in U is called left-singular vector, so is true for V. A matrix S is diagonal[2] where $S_{11} \geq S_{22} \geq \cdots$, and each element is called a singular value. This decomposition is called SVD, and it is well known that SVD takes time of $O(dN^2)$[GV89].

To obtain LSI dimensionality reduction, we calculate X^{SVD} as follows:

$$X^{SVD}_{k \times N} = U_k^T X \tag{2}$$

where U_k means a matrix of $d \times k$ generated from U by selecting the first k left-singular vectors.

For querying information, we describe each query as a vector $\mathbf{q}_{d \times 1}$ that should be projected into lower dimensional one:

$$\mathbf{q}^{SVD}_{k \times 1} = U_k^T \mathbf{q}_{d \times 1} \tag{3}$$

Then we evaluate *similarity* between the query vector $\mathbf{q}_{k \times 1}$ and candidate document vectors, and generate ranking as the answers in descending order.

Similarity is defined as *cosine* value between query vector and document vector. According to the definition, similar documents should have the value close to 1.0. To examine i-th document k_i, we obtain the similarity value $\cos \theta_{k_i}$ as follows:

$$\cos \theta_{k_i} = \frac{(\mathbf{q}^{SVD}, \mathbf{X}_i^{SVD})}{|\mathbf{q}^{SVD}||\mathbf{X}_i^{SVD}|}$$

where $\mathbf{X}_i^{SVD}$ means i-th row vector in the LSI matrix.

Because of dimension reduction by LSI, there must be some error. In fact, we have the upper bound in error, called *Frobenius Property*, according to approximation theory[Pa98]D Given a TF matrix X in $d \times N$, *Frobenius norm* $||X||_F$ is defined as follows[GV89]:

$$||X||_F^2 = \sum_{i=1}^{d} \sum_{j=1}^{N} |x_{ij}|^2 \tag{4}$$

This value describes one measure of *recall-factor*.

Once X is decomposed into UCSCV through SVD process, we extract the first k row vectors to generate U_kCS_k and V_k. Then we construct a matrix X_k from X where k means a *rank* of X as below:

$$X_k = U_k S_k V_k^T \tag{5}$$

Then the following property holds between X and X_k:

$$\min_{rank(Y)=k} ||X - Y||_F^2 = ||X - X_k||_F^2 \tag{6}$$

[1]A matrix M is called *orthogonal* if $M^T \times M = I$ holds where I is identity.

[2]A matrix M is called *diagonal* if $M_{ij} = 0$ for every $i, j, i \neq j$.

This property (6) tells us that an error caused by LSI is the *minimum* in a sense of Frobenius norm.

2.2 Random Projection

Random Projection is a matrix where each element is randomly selected. In our case, a TF matrix X in $d \times N$ is projected into a lower dimensional one Y in $k \times N$ where $k << d$. Then we need a matrix R in $k \times d$, called a RP matrix.

Then dimensionality-reduced X^{RP} is obtained as follows:

$$X^{RP}_{k\times N} = R_{k\times d} X_{d\times N} \tag{7}$$

The process takes time complexity $O(dkN)$[Pa98]. Thus smaller dimensionality causes less complexity.

In RP matrix R, every element must satisfy the following two conditions:

(1) each row vector in R must have the length 1.0
(2) R is orthogonal

However, it takes much time to satisfy the second condition.

To overcome the issues above, much simpler approach has been proposed based on naive data distribution[Ac01]. More specifically each element R_{ij} is selected as follows:

$$R_{ij} = \sqrt{3} \cdot \begin{cases} +1 & (probability\ 1/6) \\ 0 & (probability\ 2/3) \\ -1 & (probability\ 1/6) \end{cases} \tag{8}$$

To generate this matrix, it takes complexity of $O(kd)$. Practically this value is really small because of $k << d$.

Query based on RP is defined similarly as LSI. That is, we project a query matrix into the one with lower dimensionality.

$$\mathbf{q}^{RP}_{k\times 1} = R\mathbf{q}_{d\times 1} \tag{9}$$

We calculate similarity values between the query vector and document vectors, and we generate ranking information just same as LSI case.

Error caused by RP is defined as Euclidean distance between vectors.

$$|\mathbf{x} - \mathbf{y}|^2 = \sum (x_i - y_i)^2$$

where x_i, y_i means i-th values of $\mathbf{x}, \mathbf{y}$. From a matrix X in $d \times N$, we extract two row vectors $\mathbf{x}_1$ and $\mathbf{x}_2$. Then we define the Euclidean distance (in d dimension) as as $|\mathbf{x}_1 - \mathbf{x}_2|$.

To our surprise, this distance can be obtained in a vector space (in k dimension) reduced by RP as follows[BM01]:

$$\sqrt{d/k}\,|R\mathbf{x}_1 - R\mathbf{x}_2| \tag{10}$$

To obtain (10), R should be orthogonal. When R^TR is close to identity I, R is almost orthogonal at the same time. We define a matrix ϵ of $d \times d$ as the measure that says to what extent R is *almost* orthogonal:

$$\epsilon = R^TR - I \tag{11}$$

Then we see elements in ϵ have the average 0.0 and the distribution $1.0/k$. This means the bigger k becomes, the smaller we have error in Euclidean distance[Ka98].

2.3 Comparison of LSI and RP

Although both of LSI and RP get dimensionality reduction by using matrix projection, there exist important difference between them. LSI takes SVD approach while RP takes random choice of values. The latter requires less amount of time complexity.

In text stream, we receive incremental information continuously. In LSI approach, we generate LSI matrices from TF matrices and, whenever changes happen in TF matrices, we have to modify LSI matrices according to the changes. Then we project incremental information into lower dimensionality using new SVD matrices. On the other hand, the RP matrices R are independent of TF matrices X, and we don't modify R at all when changes happens in X. What we need is that we project incremental information into lower dimensionality (using same R).

Errors in LSI and RP can be calculated: in LSI, the error is defined by Frobenius norm to matrices, which means LSI matrices try to preserve relationship among words and documents. On the other hands, in RP, the error is defined by Euclidean distance between two vectors, which means RP matrices try to preserve relative relationship among TF vectors. This is the *essential* reason why RP matrices are independent of data and suitable for incremental updates. Also this RP property allows us to expect good accuracy similar to LSI in cosine-based queries.

All these discussions show that LSI is not suitable for querying text stream but RP is in terms of time and space complexity: no recalculation of the matrices and no additional memory for incremental data. Note that RP matrices are orthogonal in a sense of approximation and that accuracy of RP-based queries varies widely. It is known[BM01] that the variance is inversely proportional to reduced dimensionality k, thus we expect bigger k causes better accuracy.

In the next section we describe how to process text stream by using RP technique. And, in section 4, we show some experimental results about the comparison of LSI and RP, the accuracy and the variances of RP queries, and we discuss RP technique can be applied to querying text stream.

3 Querying Text Stream

Now we are ready to develop our theory. We discuss two points, (1) how to manage and query text stream by means of RP matrices, and (2) how to manage temporal aspects of text stream.

Let R be a RP matrix. We consider text stream as a stream of document vectors. Then, whenever a new document $\mathbf{d}_{d\times 1}$ comes in a stream, we project it into $\mathbf{d}^{RP}_{k\times 1}$ through R as follows:

$$\mathbf{d}^{RP}_{k\times 1} = R\mathbf{d}_{d\times 1} \tag{12}$$

Since each document vector has temporal aspect such as timestamp, we have a collection of temporal vectors. Especially we have fresh new vector and ancient vectors at the same time. Generally it seems better to put priority on new vectors than ancient ones, because users asks of recent topics very often. When a query comes in, we expect new vectors as answers even if others satisfy the query conditions. That's why we discuss *weight* on temporal text data.

In this work, we take a weight w_a that decreases exponential along time axis. More specifically, we take a lifetime t of data (how long it lives) as a variable of w_a :

$$w_a(t) = \exp(-t/a) \tag{13}$$

A value a means a parameter by which we can manage a *drop rate*. The bigger a becomes, the more gentle w_a we have. When $a = \infty$, we have $w_a(t) = 1$ and we have no distinction. When $t = 0$ (fresh new), we have $w_a(t) = 1$.

Assume two vectors $\mathbf{d_1}^{RP}$ with lifetime t_1 and $\mathbf{d_2}^{RP}$ with lifetime t_2 in reduced dimensionality. By multiplying the cosine similarity between the two vectors by $w_a(t_1)\times w_a(t_2)$, we can capture temporal aspects thus it is possible to distinguish new vectors from ancient ones.

4 Experimental Results

In this section, we describe several preliminary about text stream data and evaluation methodologies to query them. After that we show some experimental results of the comparison of LSI and RP, and several querying text stream. Finally we discuss these results.

4.1 Preliminaries

We discuss our experiments under the environments of FreeBSD 4.6.2 on Pentium4 2.8 GHz with 1 GB real memory. Here we assign all the information to dynamically allocated array. To generate random values that are utilized for RP matrices, we use *Mersenne Twister* software[3].

To examine our approach, we assume *Reuter-21578* [4] where Reuter-21578 consists of 21578 news articles for the purpose of test collection to document classification. However, there exist some news articles that have no contents nor categories, and we utilize 19042 articles finally. We discuss Reuter-21578 mainly because each article has timestamp correctly and contineously. You might imagine other test sets whenever they keep the same properties.

We examine all the words appeared in the category names and the article contents, and we take them as terms (in a sense of our TF matrix). Then we remove *stop words* from them and make *stemming*[GF98, Ki02]. Eventually we obtain 26870 terms.

Looking at the results into detail, we see 9610 terms appear only once. These terms are not really useful to retrieve documents since they don't play comprehensive role. Thus we'd better remove non-frequent words. Here in this work we assume *Zipf's law*[Zi49] which says about experimental relationship between "frequency in use of words"and "ranking of the frequency". Applying the law to the articles, we get 2662 terms in total. [5]

We consider the news articles as text stream since each article contains *timestamp*. After sorting them in timestamp and dividing the duration into time-intervals of 6 hours, we have 199 non-empty chunks each of which contains 95.7 articles in average and 422 articles at most.

[3]http://www.math.keio.ac.jp/ matumoto/emt.html

[4]http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html

[5]There are two kinds of Zipf's laws, one for frequent terms and another for non-frequent terms. The first law says that frequency f in use of a term and the order r of the ranking satisfy the equation

$$f \times r = C \tag{14}$$

In the second Zipf's law, the number F_f of terms of frequency f and the number F_1 of terms that appear only once satisfy the relationship

$$\frac{F_1}{F_f} = \frac{f(f+1)}{2} \tag{15}$$

Then we obtain appropriate frequency suitable for useful terms, thus we examine frequency f_k that satisfies both laws. To do that, in the equation (15), we put $F_{f_k} = 1$. Then we obtain f_k value as follows:

$$f_k = \frac{\sqrt{8F_1 + 1} - 1}{2} \tag{16}$$

Accoding to the value for f_k, we select the number of terms:

1. We select all the words of the frequency f_k as terms.
2. Also we select all the words of the maximum frequency to frequency $f_k - 1$ as terms. Assume there are K terms.
3. We select the top K words which have the frequency less than or equal $f_k + 1$ as terms.

In our experiment, we have $F_1 = 9610$ and we get $f_k = 138$ by putting it into the equation (16). According the procedure above, we have $K = 1339$ and finally 2660 terms in total. By examining Reuter-21578, we select 2662 terms with frequency 48 or higher.

4.2 How to Evaluate Results

We take *average precision factor of 11 points* which means average precision factors in the cases of recall-factors 0.0,0.1,0.2,...,1.0 in queries. This plays an overall measure to describe relationship between precision and recall factors.

Recall factor has been devised as a measure how comprehensive query result has, defined as follows:

$$Recall = \frac{NumberOfCorrectDocumentsRetrieved}{NumberOfCorrectDocuments}$$

Precision factor is a measure how precise query results have, defined as below:

$$Precision = \frac{NumberOfCorrectDocumentsRetrieved}{RetrievedDocuments}$$

There exists trade-off between recall and precision factors. Clearly we'd better have query systems with recall 1.0 and precision 1.0. However, it is common that higher recall causes lower precision and that higher precision causes lower recall. In our case, we evaluate query without dimensionality reduction and then we select documents with similarity value 0.5 or higher, by which we can examine query results under the influence of dimensionality reduction.

We also apply the average precision of 11 points to text stream. In the following experiments, we evaluate queries each time a new chunk of documents (in 6 hours) comes in as incremental information and we obtain a series of the average precisions, the average and so on. To evaluate queries in text sream, we select documents with similarity value 0.5 or higher without dimensionality reduction.

4.3 Comparison of LSI and RP

First of all, let us describe experimental results of LSI and RP. The main purpose of this experiment is that we show how well RP technique works compared to LSI in terms of time and accuracy. Then we discuss these comparison in this section. After next section, we only discuss Text Stream using RP.

To discuss LSI queries, we select first 10,000 articles from text stream, which means we need SVD processing to a matrix of $2662 \times 10,000$. And this takes time of 21469 seconds (about 6 hours).

In RP queries, we evaluate all of the 19042 articles at once, and we show the computation results in a table 1. Clearly readers see computation time is in proportion to the number of dimensionality.

Now let us describe query accuracy of LSI and RP along with 10 kinds of dimensionalities (5,..., 250). There are 635 correct documents in RP and 407 in LSI respectively. We examine RP queries 3 times on each dimensionality. We generate RP matrix each time for

Dimension	Time (seconds)
100	74
200	150
300	231
400	308
500	387
(LSI:2662)	21469

Table 1: Computation Time in RP/LSI

dimensionality reduction and w obtain the average of the average precision factors of 11 points the minimum, the maximum and the variance.

We show the LSI result in a figure 1 and the RP result in a figure 2.

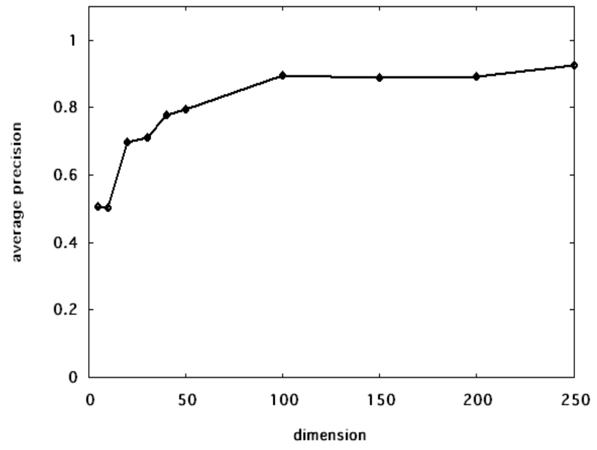

Figure 1: Accuracy in LSI

All the experimental results tell us that computation time of queries by RP is always superior to LSI case. This is mainly because SVD calculation takes time a lot and the matrix depends on every TF matrix. Also smaller dimensionality of TF matrices doesn't relieve SVD calculation. On the other hand, RP matrices are easily and efficiently obtained. The matrices become smaller dramatically according to smaller dimensionality of TF matrices.

Compared to query accuracy, RP is superior to LSI except 5 dimension, although RP shows big variances. The higher dimension we have, the smaller the variances become. In 100 dimension, we have very small difference (1% to 2 %) between the maximum and the minimum. Thus we can say RP technique is superior to LSI in more than 100 dimension.

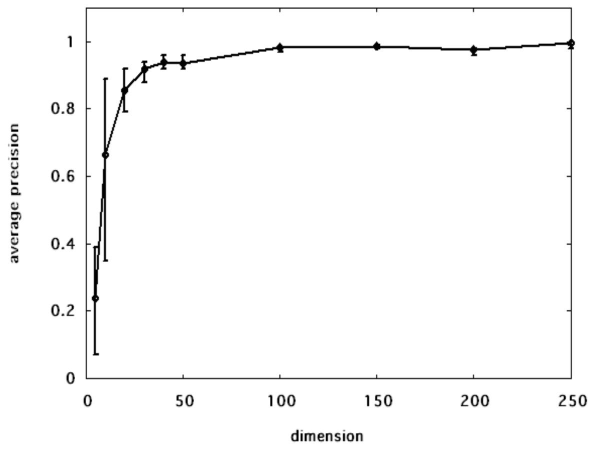

Figure 2: Accuracy in RP

4.4 Text Stream Using RP

As we said, we consider our news articles as text stream by sorting transaction time of articles. Then we take weight scheme for the stream according to the equation (13). Here we take *day* as a unit (for example, 0.25 means 6 hours).

Here in the equation (13), we give 3 kinds of weight parameters, *radical* ($a = 10$), *gentle* ($a = 45$) and *transparent* ($a = \infty$). With the gentle parameter, the weight value decreases Down to 0.5 in 30 days and down to 0.05 in 130 days. With the radical parameter, on the other hand, the weight gets to 0.5 in 7 days and to 0.05 in 30 days. With the transparent parameter, the value is always 1.0.

With all the weight values, we evaluate queries in the case of 100, 300 and 500 dimensions.

The 9 results of the average precision factors of 11 points are shown in figures 3 (radical case), 4 (gentle case) and 5 (transparent case).

By taking average values of these results, we show, in a table 2, our expected values of the 9 average precision factors of 11 points.

Dimension	100	300	500
radical	0.968	0.980	0.992
gentle	0.979	0.992	0.997
transparent	0.982	0.998	0.995

Table 2: Average Values of Average Precision of 11 Points

In our experiments, we often see no similar recent articles to a given query vector and no

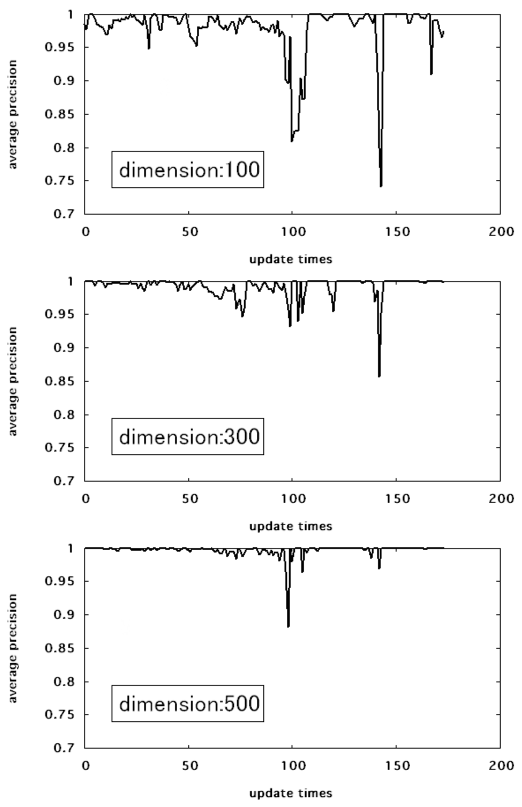

Figure 3: Query Accuracy with weight $\exp(-t/10)$

correct document is found. In this case we ignore this query and remove answer from our results. A table 3 contains the number of ignored queries in all the 199 queries.

To see the variances of 9 query results, we examine the maximum and minimum values of the average precision factors of 11 points. In our case, we see all the maximum values are 1.0. Thus we show the minimum precision (i.e., the maximum errors) in a table 4.

4.5 Discussions

Let us look at the figure of the average precisions of 11 points. Then we see the values drop because of the articles with lower similarity (i.e., 0.5 or so) in a transparent parameter case. By incremental information we might have many articles with lower similarity and precision drops. With appropriate weight scheme, these articles disappears (the similarity become less than 0.5) and the precision gets better. In fact, with radical parameter the precision drops but locally. With gentle/transparent parameters, the precision goes down

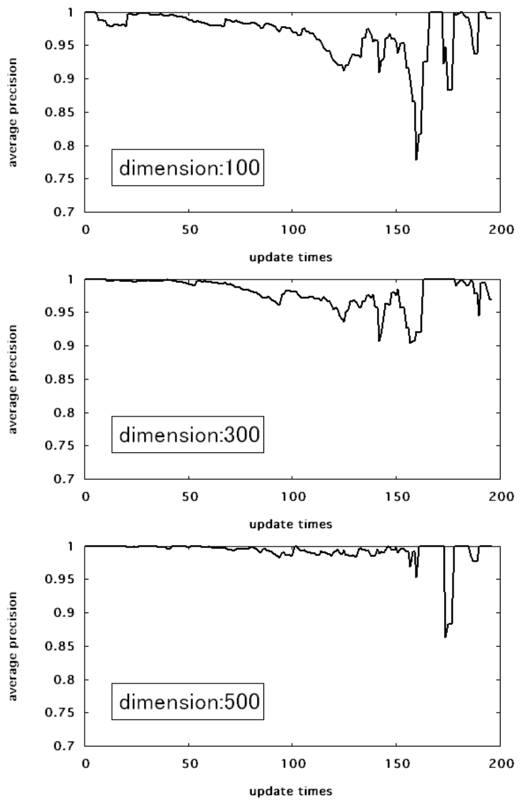

Figure 4: Query Accuracy with weight $\exp(-t/45)$

eventually.

The average of the average precision factors of 11 points is more than 90% in all the cases, and RP provides us with good accuracy for querying text stream. Especially we can say higher dimensionality means better accuracy.

Weight scheme tells us how we manage temporal aspect of stream data where transparent parameter is the best and gentle parameter the worst. However transparent parameter causes continuous degradation while radical parameter drops locally.

The number of ignored queries is the least with weight on transparent parameter but the most with radical parameter. This is because only recent articles keep alive in radical case as time goes by.

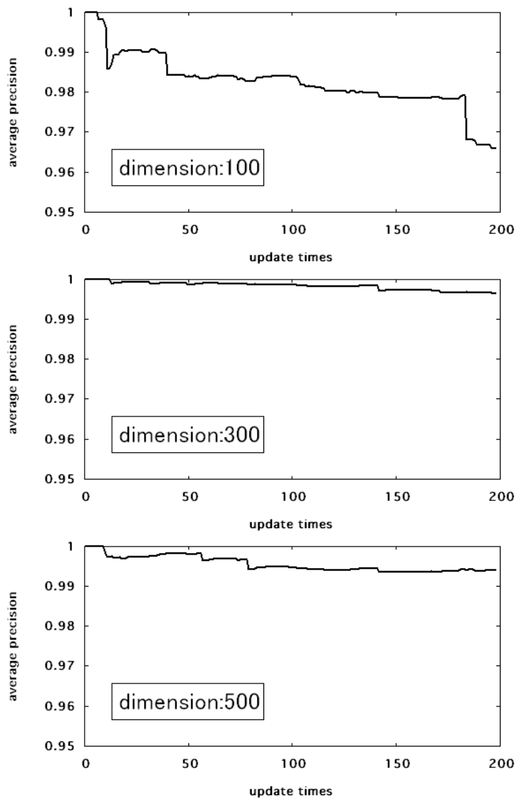

Figure 5: Query Accuracy with $a = \infty$

5 Conclusion

In this investigation, we have examined RP technique and shown how useful it is for querying text stream in terms of query efficiency and accuracy. We have also compared RP with LSI and we have discussed RP is really suitable for text stream.

In this work, we have considered limited number of documents but may have huge amount of documents in practical test stream. Since dimensionality reduction relieves memory usage, we could take this advantage for better accuracy and efficiency.

References

[Ac01] Achlioptas, D.: Database-friendly random projections, In Proc. *ACM Symp. on the Principles of Database Systems*, pp 274-281, 2001.

Parameter	Number of Ignored Queries
radical	25
gentle	2
transparent	0

Table 3: Ignored Queries

Dimension	100	300	500
radical	0.259	0.143	0.119
gentle	0.222	0.096	0.136
transparent	0.034	0.004	0.007

Table 4: Maximum Errors in Queries

[BDO95] Berry, M. W., Dumais, S. T. and O'Brien, G, W.: Using linear algebra for intelligent information retrieval, *SIAM Review*, Vol. 37, No. 4, pp. 573-595, 1995.

[BM01] Bingham, E. and Mannila, H.: Random projection in dimensionality reduction: Applications to image and text data, *Proc. 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2001)*, pp 245-250, 2001.

[De90] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K. and Harshman, R. A.: Indexing by latent semantic analysis, *journal of the American Society for Information Science*, Vol 41, No. 6, pp. 391-407, 1990.

[GF98] Grossman, D. A. and Frieder, O.: Information Retrieval – Algorithms and Heuristics, Kluwer Academic Press, 1998

[GV89] Golub, G. H. and Van Loan, C. F.: Matrix Computations, The Johns Hopkins University Press, 1989.

[Ka98] Kaski, S.: Dimensionality reduction by random mapping: Fast Similarity Computation for Clustering, In *Proc. Int. Joint Conf. on Neural Networks (IJCNN)*, Vol 1, pp. 413-418, 1998.

[Ki02] Kita, K. et al.: Algorithms in Information Retrieval, Kyorotsu, 2002 (in Japanese)

[Oh03] Oh'uchi, H., et al.: Document Retrieval by Word Meanings, In Proc. *PACRIM*, 2003

[Pa98] Papadimitriou, C. H., Raghavan, P., Tamaki, H. and Vempala, S.: Latent semantic indexing: A probabilistic analysis, In Proc.*17th ACM Symp. on the Principles of Database Systems*, pp 159-168, 1998.

[Zi49] Zipf, G. K.: The human behavior and the principle of least effort, Addison-Wesley, 1949.