A Modified Vector Space Model for Protein Retrieval

Mohammed Said Abual-Rub[†], Rosni Abdullah[††] and Nur'Aini Abdul Rashid[†††]

School of Computer Sciences, Universiti Sains Malaysia, 11800 Penang, Malaysia

Summary

This paper provides an enhancement of an existing method of information retrieval which is a modification of Vector Space Model for information retrieval. This enhanced model is modified to be applied on protein sequence data whereas the normal vector space model has been applied on text data. The results show that the enhanced model achieved very good results in performance but the setup time is somehow high for a large collection of protein sequences

Key words:

Information retrieval, Vector Space Model, Protein sequence, Bioinformatics.

1. Introduction

The Vector Space Model (VSM) is a standard technique in Information Retrieval in which documents are represented through the words that they contain. It was developed by Gerard Salton in the early 1960's to avoid some of the information retrieval problems. Vector spaces models convert texts into matrices and vectors, and then employ matrix analysis techniques to find the relations and key features in the document collection. It represents queries and documents as vectors of terms which can be words from the document or the query itself. The most important thing is to represent relevance between documents in this information space, which is achieved by finding the distance between the query and the document [1].

The weight of relevance of a query in the document can be calculated using some similarity measures such as cosine or dot product or other measurement.

Glenisson P.and Mathys J [4] have showed how the bag-of-words representation can be used successfully to represent genetic annotation and free-text information coming from different databases. They evaluated the VSM by testing and quantifying its performance on a fairly simple biological problem. They found that it can establish a powerful statistical text representation as a foundation for knowledge-based gene expression clustering [2].

In this work, we have modified the VSM technique to work with biological datasets. We used the document frequency (DF) instead of inverse document frequency (IDF). The results of the experiments show that the modified method give good results using precision evaluation measure.

2. Vector Space Model

The VSM relies on three sets of calculations. This model can work on selected index of words or on full text.

The calculations needed for the vector space model are:

1. The weight of each indexed word across the entire document set needs to be calculated. This answers the question of how important the word is in the entire collection.
2. The weight of every index word within a given document (in the context of that document only) needs to be calculated for all N documents. This answers the question of how important the word is within a single document.
3. For any query, the query vector is compared to every one of the document vectors. The results can be ranked. This answers the question of which document comes closest to the query, and ranks the others as to the closeness of the fit.

The weight can be calculated using this equation:

Eq1: $$w_i = tf_i * log\left(\frac{D}{df_i}\right)$$

where:

- tf_i = term frequency (term counts) or number of times a term i occurs in a document. This accounts for local information.
- df_i = document frequency or number of documents containing term i
- D = number of documents in a database.

The D/df_i ratio is the probability of selecting a document containing a queried term from a collection of documents. This can be viewed as a global probability over the entire collection. Thus, the $log\ (D/df_i)$ term is the

inverse document frequency, IDF_i and accounts for global information.

2.1. VSM Example

To understand Eq 1, let's use a trivial example. To simplify, let's assume we deal with a basic term vector model in which we:

1. Do not take into account WHERE the terms occur in documents.
2. Use all terms, including very common terms and stop words.
3. Do not reduce terms to root terms (stemming).

The following example [3] is one of the best examples on term vector calculations available online.

Suppose we query an IR system for the query "gold silver truck"
The database collection consists of three documents with the following content:
D1: "Shipment of gold damaged in a fire"
D2: "Delivery of silver arrived in a silver truck"
D3: "Shipment of gold arrived in a truck"
Q: "gold silver truck"
Retrieval results are summarized in Table 1 and Table 2

		Counts, tfi				
Terms	Q	D1	D2	D3	Dfi	D/dfi
A	0	1	1	1	3	3/3=1
Arrived	0	0	1	1	2	3/2=1.5
Damaged	0	1	0	0	1	3/1=3
Delivery	0	0	1	0	1	3/1=3
Fire	0	1	0	0	1	3/1=3
Gold	1	1	0	1	2	3/2=1.5
In	0	1	1	1	3	3/3=1
Of	0	1	1	1	3	3/3=1
Silver	1	0	2	0	1	3/1=3
Shipment	0	1	0	1	2	3/2=1.5
Truck	1	0	1	1	2	3/2=1.5

Table1: Retrieved results

	Weights, wi = tfi * IDFi			
Terms	Q	D1	D2	D3
A	0	0	0	0
Arrived	0	0	0.1761	0.1761
Damaged	0	0.4771	0	0
Delivery	0	0	0.4771	0
Fire	0	0.4771	0	0
Gold	0.1761	0.1761	0	0.1761
In	0	0	0	0
Of	0	0	0	0
Silver	0.4771	0	0.9542	0
Shipment	0	0.1761	0	0.1761
Truck	0.1761	0	0.1761	0.1761

Table2: Retrieved results

Vector space Model constructs the index tables as shown in Tables 1 and 2 by analyzing the terms of all documents into words as in Table 1 and find the frequency of each term in all documents; Table 2 does the same for the query.

2.2 Similarity Analysis

There are many different methods to measure how similar two documents are, or how similar a document is to a query in VSM. These methods include the: cosine, dot product, Jaccard coefficient and Euclidean distance. In this paper we will use the cosine measure which is the most common.

The similarity measure for the previous example in section 2.1 can be calculated as follows:

1. For each document and query, compute all vector lengths (zero terms ignored)

$$|D_1| = \sqrt{0.4771^2 + 0.4771^2 + 0.1761^2 + 0.1761^2} = \sqrt{0.5173} = 0.7192$$
$$|D_2| = \sqrt{0.1761^2 + 0.4771^2 + 0.9542^2 + 0.1761^2} = \sqrt{1.2001} = 1.0955$$
$$|D_3| = \sqrt{0.1761^2 + 0.1761^2 + 0.1761^2 + 0.1761^2} = \sqrt{0.1240} = 0.3522$$

$$\therefore |D_i| = \sqrt{\sum_i w_{i,j}^2}$$

$$|Q| = \sqrt{0.1761^2 + 0.4771^2 + 0.1761^2} = \sqrt{0.2896} = 0.5382$$

$$\therefore |Q| = \sqrt{\sum_i w_{Q,j}^2}$$

2. Compute all dot products (zero products ignored):

$$Q \bullet D_1 = 0.1761 * 0.1761 = 0.0310$$
$$Q \bullet D_2 = 0.4771 * 0.9542 + 0.1761 * 0.1761 = 0.4862$$
$$Q \bullet D_3 = 0.1761 * 0.1761 + 0.1761 * 0.1761 = 0.0620$$

$$\therefore Q \bullet D_i = \sum_i w_{Q,j} w_{i,j}$$

3. Calculate the similarity values:

$$\text{Cosine } \theta_{D_1} = \frac{Q \bullet D_1}{|Q| * |D_1|} = \frac{0.0310}{0.5382 * 0.7192} = 0.0801$$

$$\text{Cosine } \theta_{D_2} = \frac{Q \bullet D_2}{|Q| * |D_2|} = \frac{0.4862}{0.5382 * 1.0955} = 0.8246$$

$$\text{Cosine } \theta_{D_3} = \frac{Q \bullet D_3}{|Q| * |D_3|} = \frac{0.0620}{0.5382 * 0.3522} = 0.3271$$

$$\therefore \text{Cosine } \theta_{D_i} = \text{Sim}(Q, D_i)$$

$$\therefore \text{Sim}(Q, D_i) = \frac{\sum_i w_{Q,j} w_{i,j}}{\sqrt{\sum_j w_{Q,j}^2} \sqrt{\sum_i w_{i,j}^2}}$$

Contribution

We can easily see from the previous example that the normal VSM will not be suitable for the protein sequence data. This is because it uses the IDF in calculating the weights, and as we saw in the example, IDF gives weight zero if the term appears in all documents and that is used for the stop or common words such as: a, an, the, of,... etc Since these words are very common they exist in all documents, IDF gives these words rank 0; because usually the words that are in all documents are not relevant. However, in protein there are no stop words as in text data. So, the original method is not suitable for protein data because the existence of a term in all protein sequences gives a meaning and a weight must be given to this term.

In this paper a small modification on VSM is proposed to fit for protein sequence data; that is to use DF instead of IDF, where DF is the frequency of the term in all documents (i.e. in how many documents this term exists). This will give each document its relevance based on the frequency even if this term exists in all documents so it will be suitable for protein data. We will use the cosine similarity measure, which is the most common and has been proved by most researchers to give the best results for similarity [5].

3. Implementation

We have implemented the algorithm described in section 2 in C programming language.

Experiments were run on a group of proteins that are known to be related. We tested the system on four protein families: ribosomal protein L1, ribosomal protein L2, ribosomal protein L3, ribosomal protein L4, where each family has 50 proteins.

3.1 Results and Evaluation

The program has been tested on a collection of 200, 1000, 5000 and 10000 documents, where the document is a protein sequence as in Figure 1. We have a file for protein sequences that we want to search in, a file to input the query and an output file that gives us the retrieved results. The test of the program has been applied as follows:
We chose a sequence of amino acids as a query from the collection of protein sequence, for example from L1, and match it with the whole file and see the results. The relevant documents would be those from L1, because we get the query from L1.

```
>NF01724288 Ribosomal protein L3
[Desulfovibrio vulgaris]
MAEKMGILGRKIGVTRIFASDGSAVAVTVIK
AGPCPVTQVKTVATDGYDAIQIAFDEAKEKH
LNKPEIGHLAKAGKGLFRTLREIRLEAPAAYE
VGSELDVTLFATGDRVKVSGTSIGKGYQGVM
RRWNFAGSKDTHGCEKVHRSGGSIGNNTFPG
```

Figure 1: one protein sequence [6]

3.2 Evaluation

We used the standard IR evaluation to evaluate the algorithm. The precision gives the metric percentage of the number of relevant documents retrieved to the documents retrieved.

$$\textbf{Precision} = \frac{\textbf{Number of relevant documents retrieved}}{\textbf{Number of retrieved documents}}$$

This measure gives us how accurate the method is from the number of relevant documents we retrieved. If the precision = 1, this implies that the algorithm has successfully identified all relevant documents.

Applying this method on 200 documents with 50 relevant documents and using 10, 20 and 50 as the query length, we get the following results:

- For a query of length 10 amino acids:

Cut point	Precision
Top 5	0.80
Top 10	0.5
Top 50	0.56
Top 100	0.34

Table 3: Precision for 10 Amino Acids query

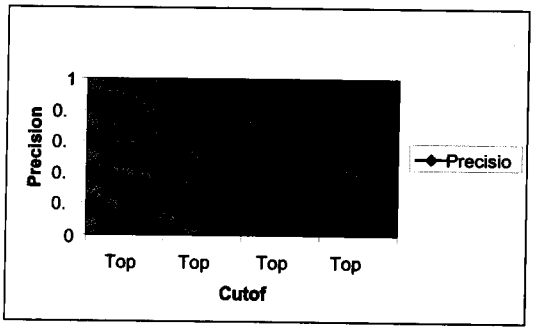

Figure 2: Precision value of a query of length 10 Amino Acids

- For a query of length 20 amino acids:

Cut point	Precision
Top 5	1
Top 10	0.80
Top 50	0.60
Top 100	0.38

Table 4: Precision for 20 Amino Acids query

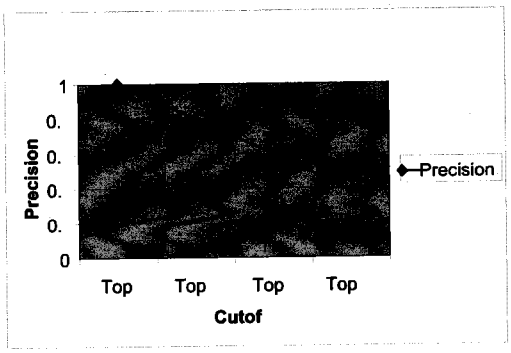

Figure 3: Precision value of a query of length 20 Amino Acids

- For a query of length 50 amino acids:

Cut point	Precision
Top 5	1
Top 10	1
Top 50	0.66
Top 100	0.39

Table 5: Precision for 50 Amino Acids query

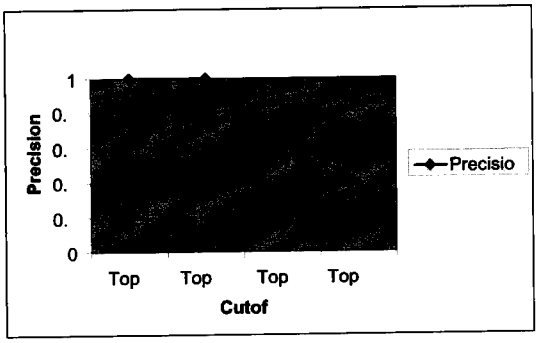

Figure 4: Precision value of a query of length 50 Amino Acids

We can see from Tables 3- 5 that the precision for a query of length 10 is 80 % this is because the query length is not long enough and can be found in many protein sequences, whereas for a query of length 20-50, the precision is 100% for a cutoff = 5-10, and reach 39% for a cutoff = 100, and this is good results for precision measure.

3.3. Setup Time

The setup time is the time for constructing the index tables showed in Tables 1 and 2 in addition to the executing time of the program starting from entering the query asking for the retrieved documents until it gives the retrieved documents.

To calculate the setup time of the program, a collection of 200, 1000, 5000 and 10000 documents has been used, taking into account that this setup time is for the first run which includes the constructing of the indexes.

Document collection	Setup time (seconds)
200	7
1000	42
5000	226
10000	790

Table 6: Setup time in seconds

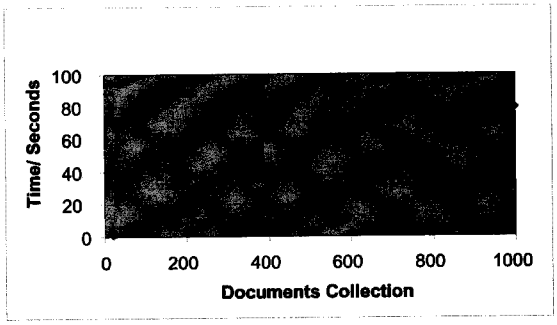

Figure 5: Setup Time

We can see from Table 6 that the setup time is quite reasonable for small documents up to 5000, but after that the setup time increases rapidly.

This can be improved by parallelizing the program distributing the data on multiple nodes which will decrease the setup time.

4. Conclusion

In this paper a modified model of VSM which is applied on protein sequences data has been introduced. The modified method achieved good results and good performance for retrieving the protein data. Using the precision evaluation measure, it gives a precision of 1 for a cutoff =10, and 0.39 for a cutoff of 100. The results show that for a small document collection the setup time is reasonable, but for large collection it gives very big setup time. Our next step is to test the program on larger data and compare the performance of this modified method with other similar methods. We also intend to explore the application of parallel techniques to reduce the large setup time.

Acknowledgments

This research has been funded by Universiti Sains Malaysia short-term Research Grant for "Parallelisation of Protein Sequence Comparison Algorithms Using Hybridised Parallel Techniques" [305/pkomp/613114].

References

[1] Van Rijsbergen, Keith,"*Information Retrieval*", Butterworths London, 1979.

[2] P. Glenisson, P. Antal, J. Mathys, Y. Moreau, B. De Moor, "*Evaluation of the Vector Space Representation in Text-Based Gene Clustering*", Pacific Symposium on Biocomputing 8 pp391-402, 2003.

[3] E. Garcia, "*Description, Advantages and Limitations of the Classic Vector Space Model", 2006.*

[4] P. Glenisson, P. Antal, J. Mathys, Y. Moreau, B. De Moor, "*Evaluation of the Vector Space Representation in Text-Based Gene Clustering*", Pacific Symposium on Biocomputing 8 pp391-402, 2003.

[5] Wahlan, Mohammed Salem Farag, "*Comparison and fusion of retrieval schemes based on different structures, similarity measures and weighting schemes", 2006.*

[6] http://www.blaststation.com/help/bs2/en/win/Chap1/01_Start.html

Mohammed Abual-Rub received his Bachelors degree in Computer Science from Yarmouk University, Irbid, Jordan in 1996 and Masters Degree in Computer Science from Universiti Sains Malaysia, Penang, Malaysia in 2007. He is currently a research officer at the Parallel and Distributed Processing Research Group and a PhD candidate as well under the supervision of Associate Professor Dr. Rosni Abdullah at Universiti Sains Malaysia.

Rosni Abdullah received her Bachelors Degree in Computer Science and Applied Mathematics and Masters Degree in Computer Science from Western Michigan University, Kalamazoo, Michigan, U.S.A. in 1984 and 1986 respectively. She joined the School of Computer Sciences at Universiti Sains Malaysia in 1987 as a lecturer. She received an award from USM in 1993 to pursue her PhD at Loughborough University in United Kingdom in the area Parallel Algorithms. She was promoted to Associate Professor in 2000. She has held several administrative positions such as First Year Coordinator, Programme Chairman and Deputy Dean for Postgraduate Studies and Research. She is currently the Dean of the School of Computer Sciences and also Head of the Parallel and Distributed Processing Research Group which focus on grid computing and bioinformatics research. Her current research work is in the area of parallel algorithms for bioinformatics applications.

Nur'Aini Abdul Rashid received a Bsc from Mississippi State University,USA and Msc from University Sains Malaysia , Malaysia and has submitted her Phd thesis at University Sains Malaysia, all in computer science. Her Phd research involved analysing and managing of proteins sequence data. Currently she is a senior lecturer at the School of Computer Sciences at University Sains Malaysia. Nur'Aini research interests include parallel algorithms, information retrieval methods and clustering algorithms.