

Efficient Approach to find Bigram Frequency in Text Document using E-VSM

Ankit Bhakkad
PICT, Pune-411043,
Maharashtra, India

S. C. Dharamadhikari
Associate Professor
PICT, Pune-411043,
Maharashtra, India

Parag Kulkarni, PhD.
Anomaly Solutions Pvt. Ltd.
Pune, Maharashtra, India

ABSTRACT

This paper proposes a novel and efficient approach to calculate bigram frequency which uses E-VSM as basis to represent text document. E-VSM: Enhanced-Vector Space Model is nothing but an extension to simple VSM which stores positions of tokens in addition to their frequency in document. Many recent methodologies in Information Retrieval and Text Mining have used bigram along with unigram since bigram gives more information gain than unigrams. Also recent efforts to provide more richer text document representation than simple 'Bag of Words' have also used bigram along with unigram. Proposed approach to calculate bigram frequency outperforms state-of-art in terms of time complexity. Analysis show that proposed approach improves time complexity to significant extent.

General Terms

Information Retrieval, Text Mining, Text Processing.

Keywords

E-VSM, bigram, trigram, n-gram, frequency count.

1. INTRODUCTION

Information Retrieval(IR) and Text Mining are all about extracting the useful information from corpus of text. It is very obvious to understand that phrase(n-gram) gives more information than just single word. There are even cases in which phrase if broken in to constituent words becomes completely irrelevant to its original meaning.

For example, "Bill Gates", when represented in the Vector Space Model which is based on 'Bag of words'(BOW) principle as separated words lose their meaning completely. Words "bill", "gates" when used separately, one can only guess that document may be relevant to accounting or gardening, which has absolutely no relevance to its original meaning. But when phrase "bill gates" is used, one will hardly mistake its meaning, its relevance to computer software[5].

Therefore in the past two decades sufficient effort has been put to come up with a richer document representation than the simple Bag-Of-Words (BOW) [2] [5] [6] [7]. There exist mainly two approaches to do it: the first one uses bigrams in addition to unigrams while the second one excludes unigrams and uses only bigrams.

Text categorization, topic and keyword extraction are the fundamental task in IR. After years of unsuccessful attempts to improve the text categorization results by applying

bigrams, it is certain that there is a limitation in usability of bigrams for text categorization. Even after there have been numerous attempts to improve document representation, the simple word-based BOW remained very effective. Even today BOW based representation gives one of the best classification results for well known Reuter-21578 dataset [7].

This implies that classification results can only be improved by extending BOW and not by replacing it. This paper proposes an approach which uses E-VSM as its basis to represent text document. It facilitates the use of bigram by providing an efficient way to calculate frequency of all existing bigrams in document. At the same time it maintains the simplicity of BOW since it uses E-VSM.

The rest of the paper is organized as follows : Section 2 discusses related work; Section 3 describe E-VSM in brief, in Section 4 contains proposed algorithm to calculate bigram frequency using E-VSM, Section 5 gives the analysis of algorithm , in Section 6 experimental results are discussed and finally Section 7 concludes the work.

2. RELATED WORK

There has been lot of work done to improve text categorization using bigram in last 25 years. Few of the relevant work are as follows:

R. Bekkerman et al.[5] explains that simply by replacing unigram by bigram does not give good results in text categorization. Rather it can be more helpful to extend existing word-based text representation.

In [6], Tan et al. presents an approach which selectively chooses bigram based on various frequency thresholds to improve categorization results.

Even to be predictive regarding which word comes next to a particular word in a document i.e. to find collocation[1] starting with the given word also depends on appearance frequency of different bigram having given word as their first element. Contingency table[1] is a commonly used data structure to compactly express various possibilities for appearance of different terms in bigram.

In topic and keyword extraction, bigram and trigram frequency count play very important role. In [3] after preprocessing the document that is tokenization, normalization and removal of stop words, top 20 highest

frequency bigram and trigram from document name, document header(first 20 words of document) and complete document as a whole are used as the strong candidates. This requires to calculate frequency for all possible bigram and trigram in document.

E-VSM[4] is an extension to original Vector Space Model. It enhances the capabilities of VSM at the same time maintain its the simplicity. Following Section describes it in detail.

3. E-VSM : Enhanced-Vector Space Model

E-VSM is a simple extension to original VSM. As VSM it is also based on BOW model only. It stores the positions of the words in the document in addition to only appearance frequency as in VSM.

Following is E-VSM:

$$D_j = \{ \{f_1, (p_1, p_2, \dots, p_{f_1})\}, \{f_2, (p_1, p_2, \dots, p_{f_2})\}, \dots, \dots, \{f_i, (p_1, p_2, \dots, p_{f_i})\} \} \quad (1)$$

Where, D_j is vector representation of document j ,
 f_i is the frequency of the i^{th} term in document j
 (p_1, \dots, p_{f_i}) represents the positions at which i^{th} term appears in document j .

4. PROPOSED ALGORITHM TO CALCULATE BIGRAM FREQUENCY

E-VSM stores all the positions of every word where it is repeated in the document. Length of position vector is nothing but the appearance frequency of that particular word. These position vectors can be exploited to find the frequency of n -gram in document. Bigram is most commonly used n -gram in text processing. Following is algorithm to calculate the bigram frequency.

Algorithm 1 : Bigram Frequency Count Algorithm

Input : Two elements of E-VSM i.e. a bigram
 $\{ \{f_1, P_1(p_{11}, p_{12}, p_{13}, \dots)\} , \{f_2, P_2(p_{21}, p_{22}, p_{23}, \dots)\} \}$

Output: Frequency count of input bigram

initialize variable *bigramFreqCount* to zero

while P_1 or P_2 does not come to end **do**

if element in P_1 is smaller than element in P_2 **then**

if element in P_2 is one greater than element in P_1 **then**
 increment *bigramFreqCount*;
 go to next element in both P_1 and P_2 ;

else
 go to next element in P_1 ;

else
 go to next element in P_2 ;

end while

return *bigramFreqCount*;

Above algorithm uses the stored positions of individual words in text document to calculate bigram frequency. It loops through elements in position vector of both input elements from start and stops when any one of them reaches the end i.e. number of comparisons required to find the frequency of any bigram is equal to length of smallest position vector. Here variable *bigramFreqCount* gives the frequency of input bigram in respective text document.

5. ANALYSIS

Time complexity calculations show that proposed approach to find bigram frequency is very much effective as compared to commonly used way.

In a text document containing n number of words, there exists $(n-1)$ bigram. Commonly used way to find the frequency of any one existing bigram, whole document has to be scanned and bigram has to compared against the words. That means for one bigram there are $(n-1)$ comparison(last word of document is not compared). Therefore time complexity to find frequency of all $(n-1)$ existing bigram will be $O(n^2)$.

In the proposed approach as stated previously, number of comparisons to calculate frequency of any one bigram is equal to length of smallest position vector among its constituent words. Since there are total n words i.e. total n positions, sum of the number of elements in all position vectors will be n . Therefore time complexity to find frequency of all $(n-1)$ existing bigram will be just $O(n)$.

Also in proposed algorithm there are only comparisons between integers (positions) while in commonly used approach strings are compared. Integer comparisons are always faster and take constant time while string comparisons are slower and depend on the lengths of strings under consideration. This further improves the speed of algorithm.

6. EXPERIMENTAL RESULTS

Experiments are performed on text documents of various length and the results are tabulated in Table 1 below. Results clearly proves the analysis given in Section 5. In Table 1, first column gives the length of various text documents considered for experiments, column 2 and 3 gives the time required to calculate the frequency of all existing bigram in document using general approach and proposed approach respectively.

Table 1. Experimental Results

Length of Text Document	General Approach (in milli seconds)	Proposed Approach (in milli seconds)
1000	25	3
3000	200	7
5000	570	14.6

Implementations are done in Java language using Netbeans 7.2 IDE. Experiments are performed on a computer with Microsoft Windows XP (SP2) operating system installed and hardware configuration Intel Core(TM)2duo CPU, T5750 @ 2GHz processor, 2GB of RAM.

7. CONCLUSION

This paper proposes a fast and efficient algorithm to calculate frequency of all existing bigrams in given text document. It uses E-VSM as the basis to represent text documents in vector

form. Algorithm exploits the positions of the words stored in E-VSM to find bigram frequency. Analysis shows that it improves the time complexity to $O(n)$ as compared to $O(n^2)$ of commonly used approach. Also experiment results prove the analysis.

This work can be easily extended to calculate frequency of n-gram for higher values of n. Also we further aim to build efficient system to support collocation prediction and skipgram.

8. ACKNOWLEDGEMENT

We thank Prof. M. Emmanuel, Head of Dept. of Information Technology, PICT, Pune for constant help and support.

9. REFERENCES

- [1] Matthew A. Russel, "Mining the Social Web", O'Reilly (2011), chapter 7, pp 224-229
- [2] Braga, Igor, Maria Monard, and Edson Matsubara (2009), "Combining unigrams and bigrams in semi-supervised text classification", Proceedings of Progress in Artificial Intelligence, 14th Portuguese Conference on Artificial Intelligence (EPIA 2009), Aveiro, pp. 489-500.
- [3] Yashodhara Haribhakta, Arti Malgaokar and Dr. Parag Kulkarni, "Unsupervised Topic Detection Model and Its Application in Text Categorization", 2012 ACM 978-1-4503-1185-4/12/09
- [4] Ankit Bhakkad, S.C. Dharmadhikari, Parag Kulkarni and M. Emmanuel, "E-VSM : Novel Text Representation Model to Capture Context-based Closeness between two Text documents", Proceedings of 7th International Conference on Intelligent Systems and Control (ISCO 2013), Coimbatore, India, pp. 345-348.
- [5] R. Bekkerman and J. Allan., "Using bigrams in text Categorization", Technical Report IR-408, Department of Computer Science, University of Massachusetts, Amherst, MA, 2004.
- [6] M. Tan, Y. F. Wang, and C. D. Lee., "The use of bigrams to enhance text categorization", Information Processing and Management, 38(4):529–546, 2002.
- [7] T. Dumais, J. Platt, D. Heckerman, and M. Sahami, "Inductive learning algorithms and representations for text categorization", In Proceedings of CIKM'98, 7th ACM International Conference on Information and Knowledge Management, pages 148–155, Bethesda, US, 1998. ACM Press, New York, US.