An Improved Classification Method for the Common OLE File by N-gram Analysis and Vector Space Model

Hong-Rong Yang, Ming Xu, Ning Zheng

Institute of Computer Application Technology, Hangzhou Dianzi University, P. R. China. Email: mxu@hdu.edu.cn

Keywords: N-gram, OLE file, vector space model.

Abstract

Identifying file type by file extension is fallible. Another magic bytes method for these files, which have similar header information, such as the common-used MS Office OLE file, may not distinguish one type from another. In this paper, an efficiently classification method for the common OLE files was proposed. In order to overcome the shortcoming of the original N-gram analysis technique which can not easily tell ambiguous file types apart, the N-gram analysis and the vector space model were combined together to identify the common OLE files. The characteristic items were extracted from the most frequency byte values of each file class, and then the cosine value of two vectors was used to catalogue ambiguous file types. The experiment results demonstrate that our mechanism is effective in identifying the office OLE files, and obtain better performance than the common n-gram method.

1 Introduction

Nowadays, it's an increasingly important work to identify file type efficiently. There are three primary methods for file type identification. The fastest and simple method is according to its file extension. But this way is not convincing because file extension can be easily changed manually. Another common-used method is magic byte technique. The magic bytes are specific to binary files and rely on matching signatures that vary in length from 2 to 46 bytes in file headers [3]. However, in some cases, we can't give a special file type using magic byte. (for example, WORD, EXCEL and PPT). The third method to identify file type is using the character distribution. This is used to detect different file types by the content of the file.

N-gram, which is a common-used statistical method, has been used in file classifying field successfully. We have tried to make use of this method to show the average character distribution of the three types of files which have said above. Unfortunately, the result displays that the top three most frequency byte values of these three types of files are the same, those are "0", "255", "1". Not only these three file types, but also anther two types of MS office OLE files (VISIO, ACCESS) produce the similar character distribution. Moreover, there is also a large overlap between other top byte values of these five types of OLE files, which has been shown in figure 1. Here, X axis: bytes from 0 to 255, Y axis: average frequency of byte values. Since the byte value 0 is used often to pad files in various formats and takes up a large proportion of total file, one may ignore this value and focus on the remaining byte value distribution.

WORD, EXCEL, and PPT are all MS OLE 2.0 compound document file format [6] containing the fixed first 8 bytes (D0 CF 11 E0 A1 B1 1A E1) and similar character definition. VISIO and ACCESS which are also MS Office OLE file types would have similar character distribution with the above three file types. Not surprisingly, widely-used N-Gram analysis may not be very effective in this case. How to category files which have similar character distribution with N-gram-based method and achieve a high accurate rate are what we desire to solve.

In our work, we integrate the N-gram technique with the vector space model to classify OLE files. Our main contributions are summarized as follows: proposing an improved classification method which combines N-gram analysis technique and vector space model together, proposing an efficient approach which can extract characteristic byte values from each file type that have similar character distribution, demonstrating the availability of our classification methods by identification the five types of common OLE files and obtain a satisfying result.

The rest of this paper is organized as follows: Section 2 describes previous work on file classifying using N-gram technique. Section 3 discusses our methodology, and in section 4, we present results on the efficacy of our approach and compare it with CNG method. In section 5, we conclude and outline future works.

2 Related work

McDaniel and Heydari [5] introduced content based file type detection algorithm, which adopted byte frequency analysis and automatically generating "fingerprints" of file types based on a set of known input files. Wei-Jen Li, Ke Wang, and Salvatore J. Stolfo [10] proposed N-gram-based method to categorize file types. In their work, they represented all members of the same file type by a set of statistical 1-gram models and applied Mahalanobis Distance to calculate the similarity between the test file and the model. For experiment, they considered WORD, EXCEL, and PPT as one file type rather than differentiate them. Our work is just classifying them including other two types of common OLE files. V. Keselj, F. Peng, N. Cercone, and C. Thomas [9] presented Common N-Gram analysis (CNG) method to automated authorship attribution based on byte n-gram models.

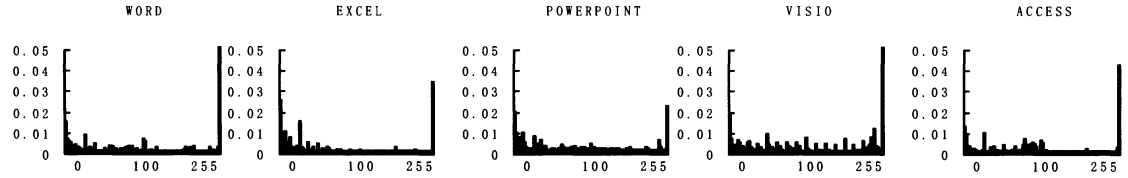

Figure 1: File binary distribution. (the 255's frequency of WORD and VISIO is larger than 0.05)

Table 1: The top 15 frequency in 1-gram of the five types OLE files(1-gram is denoted by its ASCII value)

File type	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
word	0	255	1	32	4	128	2	132	5	3	7	48	8	6	15
excel	0	255	1	2	32	10	8	3	16	5	6	4	253	48	12
ppt	0	255	1	2	4	15	32	8	3	240	45	16	39	128	241
visio	255	0	1	240	2	63	235	64	127	4	191	3	15	5	32
access	0	255	1	32	2	101	4	127	128	114	116	111	110	97	3

The CNG method represents each text sample as a bag of character *n*-grams taking into account case-sensitive information. Although quite simple, the CNG method has achieved remarkable performance in text classifying field. The distance measure used to compare the text models is using KNN (k-nearest neighbour) algorithm with k=1. However, this algorithm is not stable enough for classifying quite a few file models which have similar char distribution.

3 Methodology

3.1 N-gram analysis and vector space model

N-grams are generalized words consisting of N consecutive symbols [4, 8]. A set of the top M most frequency character *n*-grams are able to capture complicated stylistic information on the lexical, syntactic, or structural level. N-gram analysis has been widely used in a number of text classification tasks, such as indexing, information retrieval, error correction, text compression, language identification and so on. However, 1-gram statistical analysis may not effective enough to distinguish one type from another. In these cases, a higher order gram (2-gram or 3-gram) analysis may perform better. Considering computation speed and performance, we mainly focus on 1-gram analysis in this paper.

The vector space model has been widely researched in the information-retrieval research community and produced some highly successful results. In the statistically based vector-space model, a file is conceptually represented by a vector of keywords extracted from the file, with associated weights representing the importance of the keywords in the file and within the whole document collection. There are two important factors to build a reliable vector space model. One is the mechanism of extracting keywords from files. Another is the algorithm of determining the weight value of each keyword. Several ways to compute weight values have been proposed. A common-used approach is called tf*idf method [2]. Because the 256 ASCII characters occur in every file, the idf would be taken no account of while determining the weight of the keywords.

3.2 Extracting characteristic item by Set Difference

Table 1 displays the top 15 most frequency in 1-gram of the five types sample OLE files , which demonstrates that there is a large overlap between their character distributions.

In order to exact characteristic items from each file type efficiently, we need to normalize the top M most frequency characters without considering the same characters containing in the top ones. Set Difference can suffice this. Set Difference is a basic set operation, and returns a new set which represents the difference between the base set and another set. By using Set Difference, we can easily pick particular characters out of the top ones. And then, we might do the same work on several groups and obtain the public ones which represent this file type.

3.3 Categorizing and testing method

Each files model's characteristic items can be merged to build the characteristic vector which represent the difference between these file types. The aim is to identify a particular file type while given an unknown file. Thus a proper method is required to categorize different file types. The common ranking function of processing vector is the cosine measure, which determines the similarity between the test file's vector and the file model's vector. The similarity between a file model F_i and a test file T is defined as

$$sim(T,F) = \sum_{j=1}^{v}\left(W_{T,j} \times W_{i,j}\right) / \sqrt{\sum_{j=1}^{V} W_{T,j}^2 \times \sum_{j=1}^{V} W_{i,j}^2} \qquad (1)$$

The dimension of vector space is V. $W_{T,j}$ is the weight of character j in the test file, and is defined in a similar way as $W_{i,j}$(that is equivalent to $tf_{i,j}$).By calculating the cosine value between the test file's vector and each file model's vector, we can easily find out the biggest cosine value, and judge the test file belongs to the corresponding file type.

4 Evaluation

4.1 Preparing for dataset

First of our experiment, for each file type, we need to prepare a great number of training samples. We collected the needed sample files from Internet by Google. However, we found

that randomly choosing files may not be a perfect way to build file model. If we choose quite a few large-size WORD files, the character distribution can be shown a great difference from the uniform one which take account of little large-size ones. Toward this end, we limit that the proportion of large-size samples in the total samples we select must be lower than a threshold (i.e. 20%).

Finally, we collect 100 files for each type, the size of which varies from 9 to 4,744 kilobytes. While doing this work, we have eliminated a few too large files (larger than 5,000 kilobytes) which may disturb the experiment results. We give detail of sample selecting, for example, the number of samples selected in different sizes, which displays in table 2. For the reason that we use 5-fold cross-validation, the number of files of different sizes can easily be divided by 5.

After having gotten samples ready, we can calculate the average 1-gram's frequencies of the training samples of each type and build file model for the following work. Using 2gram as file's model, we couldn't find any better performance than 1-gram's. More training samples and syntax information are required for our future work.

Table 2: The number of samples selected in different file sizes

File type	*0-500K*	*500K-1M*	*1M-5M*
word	60	20	20
excell	80	15	5
ppt	40	40	20
visio	80	15	5
access	50	30	20

4.2 Generating and optimizing of characteristic vector

Five types of MS OLE files (WORD, EXCEL, PPT, VISIO, and ACCESS), which have similar char distributions as Figure 1 plots, are tested in our experiments.

We put each file type's top M frequency into a set, which is denoted by A_i (i=1, 2, 3, 4, 5). The characteristic vector is obtained by merging each file type's characteristic items, which can be extracted by computing the set difference among them. The characteristic vector is defined as follows:

$$V = \bigcup_{i=1}^{5}(A_i - \bigcup_{j=1, j\neq i}^{5} A_j) \qquad (2)$$

4.3 Experimental results

Our experiment's platform is under UNIX, which is convenient for programming. The test tool we developed is written in C, and used to extract n-gram (n=1) models of each file type. There are three primary parameters in our experiments: profile size, file size and truncation size. The profile size, L, determines the number of the most frequent 1-grams of a file. The truncation size is size of small portion of file. We decide to choose profile size ranging from 8 to 20, and obtain the characteristic vector, computed by Equation (2). The cosine similarity, computed by Equation (1), is measured between the test file's vector and file model's vector, and the class with the biggest similarity is chosen.

We perform 5-fold cross-validation [1]. The data is evenly partitioned into 5 folds. Each size of fold is 20. 4 of these datasets are used for training and the remaining dataset is used for testing. The process is repeated 5 times, each time using a different testing dataset. The results in these 5 evaluations are averaged to obtain the final result.

A The different profile sizes

To compare the results using different profile sizes, we choose 8, 11, 15, and 20 top frequency 1-grams for each file type, and obtain different dimensions of characteristic vectors which have been given in table 3 as follow.

Table 3: The different vectors with different profile sizes(L)

L	*characteristic vector*
8	{128,132,10,8,15,63,64,240,101 }
11	{132,7,10,6,16,63,64,235,191,101,114,116}
15	{132,7,48,10,253,12,45,241,63,64,235,191,114, 116,111,110,97}
20	{132,214,74,104,10,253,12,29,27,45,95,39,13, 63,235,191,223,18,114,116,110,97,65,115,105}

The result using 5-fold cross-validation is displayed in table 4. The performance is best while profile size is 15. The lowest accuracy is no less than 91%, and the best one is higher than 99%. As mentioned in the subsection 3.2, byte frequency is regarded as weight value. We have shown five types of file's weight values of characteristic vector (L=15) in table 5.

Table 4: The classifying accuracy using different profile size(L)

L	*word*	*excel*	*ppt*	*visio*	*access*
8	69.9%	81.7%	72.3%	82.5%	70.6%
11	82.3%	91.6%	89.8%	94.6%	82.3%
15	91.6%	98.1%	94.4%	99.6%	93.5%
20	84.6%	90.7%	83.5%	95.5%	81.1%

B The different file sizes

To observe the files mistaken as other types, we display the classifying accuracy of different file sizes in table 6. Although byte frequency has been normalized by file size, Office files can embedded mutually and disturb the building of file model. We use 15 as profile size. In table 6, a great mass of mistakes are made in large-size files. Thus it is not surprising that WORD, PPT, and ACCESS, which include more large-sizes ones, obtain lower accuracy than the other two types. We can add deeper analysis to our future work to identify appropriate file sizes to improve detection performance.

Table 6: Classifying accuracy of different file sizes

File type	*0-500K*	*500K-1M*	*1M-5M*
word	98.3%	85.0%	80.0%
excel	98.7%	100%	80%
ppt	97.5%	95.0%	85.0%
visio	100%	100%	80.0%
access	98.0%	93.3%	80.0%

Table 5: The weight value of characteristic vector with profile size=15 (the value has been enlarged 10000 times)

File type	7	10	12	45	48	63	64	97	110	111	114	116	132	191	235	241	253
word	55	31	35	16	54	17	28	26	24	39	20	25	66	18	14	15	21
excel	38	108	53	14	56	15	53	8	10	12	9	22	6	4	1	2	68
ppt	40	42	41	61	43	26	27	26	29	30	25	31	31	31	20	51	25
visio	34	28	23	19	44	98	82	43	22	60	35	33	23	74	83	65	30
access	37	27	21	32	43	05	38	47	48	50	.58	53	10	7	19	4	13

C The different truncation sizes

In previous related works, truncation is commonly used to fix a small portion of entire file, such as first 200 bytes. It can be efficacious while the test files have dissimilar file headers. For our work, we seek to determine whether the result produced by different truncation sizes is better than the entire one's. In this test, we also use **15** as profile size. The result is shown in table 7, which indicate that classifying under different truncation size produces no better result than considering the entire file. WORD, EXCEL, and PPT obtain dissatisfying accuracies while the truncation sizes are small, especially 50. All the three types of files belong to MS compound document, which has fixed-size (512bytes) file header and uniform encoding. That is the reason why these three types can't easily be differentiated by first hundreds of bytes. In contrast, ACCESS, which contains dissimilar header with other 4 types, performs better under small truncation size.

Table 7: Classifying accuracy under different truncation size

Truncation Size	*word*	*excel*	*ppt*	*visio*	*access*
50	0%	0%	0%	1.2%	100%
512	8.3%	96.7%	23.3%	89.6%	98.8%
700	88.3%	81.5%	50.5%	96.7%	99.2%
900	93.8%	85.2%	42.8%	97.5%	97.5%
all	91.6%	98.1%	94.4%	99.6%	93.5%

D Comparing our approach with CNG method

In CNG method, the top M frequent n-grams with their normalized frequencies represent a class model. The instance is classifies using k-nearest neighbour algorithm [9] with k = 1, which takes account of all the top ones, including quite a few characters appear in the top ones of each file type. We perform an experimental evaluation of our approach and CNG method (n=1) by measuring their Macro-F_1 value [11], which considering both precision rate and recall rate. Figure 2 displays the different results produced by these two methods. As figure 2 plots, X-axis denotes different truncation sizes (in bytes), and Y-axis denotes Macro-F_1 value. In general, higher Macro-F_1 values are obtained by our approach. Our improved classification method is demonstrated to be given a good evidence for identifying file types with similar character distribution.

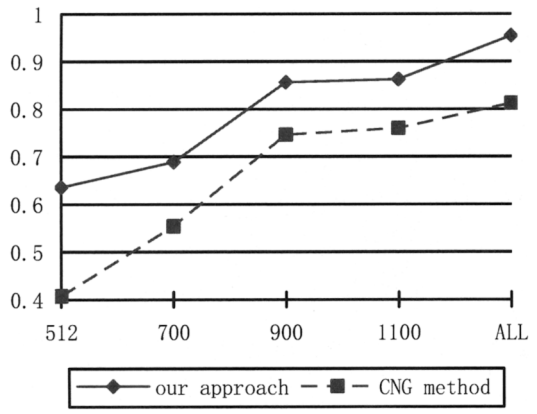

Figure 2: The Macro-F_1 value of two methods

5 Conclusion and future work

In this paper, we propose an improved classification method for common OLE files which have the similar character distribution. Our approach is based on n-gram analysis and integrates it with vector space model. The key technique of this method is extracting characteristic items effectively from each file types. We have applied this mechanism to differentiate five common OLE files and obtain better performance comparing with CNG method. We believe that this classification method can also be applied in other group of files, such as EXE and DLL. Moreover, in malcode detection field, by n-gram analysis, there is large overlap between the Mahalanobis distances of virus blocks and normal blocks in infected PDF documents [7]. We hope that our approach can also be effective in this case, which is part of our ongoing work.

Acknowledgements

This work is supported by the Natural Science Foundation of Zhejiang Province of China under Grant No.Y106176 and the Science and Technology Projects of Zhejiang Provincial under Grant No.2007C33058.

References

[1] C Manning, H Schuetze. "Foundations of Statistical Natural Language Processing", The MIT Press, (1999).

[2] Dik l. Lee, Huei Chuang, Kent Seamons. "Document ranking and the vector-space model". Software IEEE. **14(2)**:67-75, (1997).

[3] Douglas J. Hickok, Daine Richard Lesniak and Michael C. Rowe Ph.D. "File Type Detection Technology", University of Wisconsin-Platteville, (2005).

[4] M Damashek. "Gauging similarity with n-grams: language independent categorization of text", *Science*, **267(5199)**:843-848, (1995).

[5] McDaniel, M. Hossain Heydari. "Content Based File Type Detection Algorithms", HICSS, (2003).

[6] "OpenOffice.org's Documentation of the Microsoft Compound Document File Format", Available at http://sc.openoffice.org/compdocfileformat.pdf, (2007).

[7] Sal Stolfo, Ke Wang and Wei-Jen. Li. "Fileprint Analysis for Malware Detection." Technical report Columbia University, (2005).

[8] T. Abou-Assaleh, N. Cercone, V. Keˇselj and R. Sweidan. "N-gram- based Detection of New Malicious Code", COMPSAC, (2004).

[9] V. Keˇselj, F. Peng, N. Cercone, C. Thomas. "N-gram-based Author Models for Authorship Attribution", PACLING, (2003).

[10]Wei-Jen Li, Ke Wang and Salvatore J. Stolfo. "Fileprints: Identifying File Tytes by n-gram Analysis", 6th IEEE SMC Information Assurance Workshop. **2005-06**:64-71.

[11] Y.Yang, X.Liu. "A Re-examination of text categorization methods", Proceedings of ACM SIGIR Conference of the Research and Development in Information Retrieval, (1999).