# Fast File-type Identification *

Irfan Ahmed, Kyung-suk Lhee, Hyunjung Shin, ManPyo Hong
Ajou University, South Korea
{irfan, klhee, shin, mphong}@ajou.ac.kr

## ABSTRACT

This paper proposes two techniques to reduce the classification time of content-based file type identification. The first is a feature selection technique, which uses a subset of highly-occurring byte patterns in building the representative model of a file type and classifying files. The second is a content sampling technique, which uses a subset of file content in obtaining its byte-frequency distribution. Our initial experiments show that the proposed approaches are promising even the simple 1-gram features are used for the classification.

## Keywords

Byte frequency distribution, file type identification

## 1. INTRODUCTION

File types are usually identified by the file extensions or the magic numbers in the file header. However, these methods can easily be deceived by changing the file extension or altering the magic number. Therefore, especially in the presence of adversaries, a more reliable solution is needed to identify a file type. Analyzing file contents to find distinguishable patterns among file types is a viable alternative, but not widely used yet. One of the reasons is that it is inefficient and time-consuming. Existing techniques for instance, Fingerprint [2],Fileprint [1] of this approach generate the byte-frequency distribution of a file and classify the type of a file using statistical or data mining techniques. Time to obtain the byte-frequency distribution of a file may take long time because it scales with the size of the file. Also, analyzing the byte-frequency distribution to classify the file may require a large memory space and computation time, especially if we want to analyze not the individual bytes but a sequence of $n$ bytes ($n$-gram); as we increase $n$, the number of distinct byte patterns increases exponentially.

This paper proposes two approaches to reduce the classification time and the time to obtain the byte-frequency distribution (i.e. features) of a file. Firstly, we proposes a feature selection technique. It uses a subset of high-frequency byte patterns as features which may be sufficient to build the representative model of a file type. Using a subset of high-frequency byte patterns may even increase the classification accuracy, if low-frequency byte patterns are noises, not the data contributing to the classification. Secondly, instead of using the entire file content, we propose a content sampling technique that uses a certain percentage of a file to reduce the time in obtaining byte-frequency distribution of a file.

To prove the effectiveness of the feature selection technique, we tested it on the six most popular classifiers: neural network (NN), linear discriminant analysis (LDA), K-means, K-nearest neighbor (KNN), Decision tree (DT), and support vector machine (SVM).

*supported by the Ubiquitous Computing and Network(UCN)Project, Knowledge and Economy Frontier R&D Program of the Ministry of Knowledge Economy(MKE) in Korea and a result of subproject UCN 09C1-C5-20S.
H.Shin would like to gratefully acknowledge support from Post Brain Korea 21 and the research grant from Korean Government (MOEHRD, KRF-2008-531-D00032).


*SAC'10* March 22-26, 2010, Sierre, Switzerland.


## 2. PROPOSED APPROACHES

### 2.1 Feature selection technique

Assuming that a few byte patterns that most frequently occur may be sufficient to represent the file type, we propose to use a subset of high-frequency byte patterns as features. Since each file type has different set of high-frequency byte patterns, we merged the sets of high-frequency byte patterns into a unified set of features for all the file types, which is to be fed into the classifier.

To merge them we tried two strategies: union and intersection. The union combines the feature sets of all the file types, and the intersection extracts the common set of features among the file types. The result of union operation may include low-frequency byte patterns for certain file types, if those byte patterns occur frequently in other file types. In contrast, the result of intersection operation guarantees that only the high-frequency byte patterns are included, but some of the high-frequency byte pattern will be omitted if they do not occur frequently in all the file types. Table 1 shows the feature sets resulting from union and intersection operation.

### 2.2 Content sampling technique

Obtaining the byte-frequency distribution may take huge amount of time if the whole file is used. Instead of using the

| % of high frequency byte patterns | Number of patterns after Union | % of features selected (Union) | Number of patterns after Intersection | % of features selected (Intersection) |
|---|---|---|---|---|
| 10 | 102 | 39.84 | - | - |
| 20 | 155 | 60.54 | - | - |
| 30 | 202 | 78.90 | 2 | 0.78 |
| 40 | 236 | 92.18 | 6 | 2.34 |
| 50 | 247 | 96.48 | 15 | 5.85 |
| 60 | 253 | 98.82 | 33 | 12.89 |
| 70 | 256 | 100 | 51 | 19.92 |
| 80 | - | - | 75 | 29.29 |
| 90 | - | - | 122 | 47.65 |
| 100 | - | - | 256 | 100 |

**Table 1: Percentages of the high-frequency byte patterns (features), and the number of features chosen by union and intersection.**

| File types | NN | DT | LDA | K-means | KNN | SVM |
|---|---|---|---|---|---|---|
| ASP | (99) | 98 | (99) | 32.5 | 98.5 | 95.5 |
| DOC | (88) | 80.5 | 81.5 | 58 | 87 | 75.5 |
| EXE | (99) | 95.5 | 91 | 91.5 | 96 | 87 |
| GIF | 99.5 | 94.5 | 90.5 | (100) | 91 | 90.5 |
| HTML | 53 | 60 | 60 | 18.5 | (73.5) | 66 |
| JPG | 67 | 84 | 92 | 8 | 90 | (92.5) |
| MP3 | 98 | 93.5 | 98 | 14 | (99.5) | 96 |
| PDF | 94 | 94 | 90.5 | 78 | (97) | 95 |
| TXT | 80 | 79 | 3 | (95.5) | 90 | 88.5 |
| XLS | 89.5 | 88.5 | 87 | 84.5 | 82.5 | (95) |

**Table 2: The accuracies of classifiers for each file type, using 40% of byte patterns obtained by union operation. Circled cells show the highest accuracy for each file type.**

entire file content, Assuming that partial content of a file may be enough to generate a representative byte-frequency distribution of the file type, we propose to sample the file content to reduce the time in obtaining byte-frequency distribution..

To evaluate the effectiveness of this approach, we tried sampling a initial contiguous bytes and sampling a few small blocks in random locations in a file. The first method is the fastest way of sampling, but the obtained data are location-dependent and hence may be biased. The second method gathers location-independent data and thus free from such problem, but is slower than the first method (albeit much faster than using the whole file) because files are sequentially accessed medium.

## 3. EMPIRICAL RESULTS

It is noticed that union operation produces more accurate representative byte patterns of file types than intersection operation. Table 2 shows the classification accuracy of the

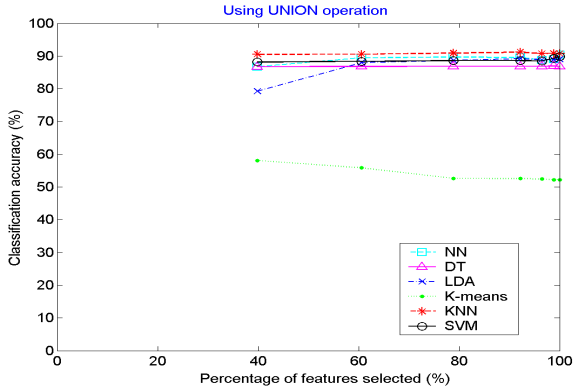


**Figure 1: The average classification accuracies of the six classifiers. Features are selected using union.**

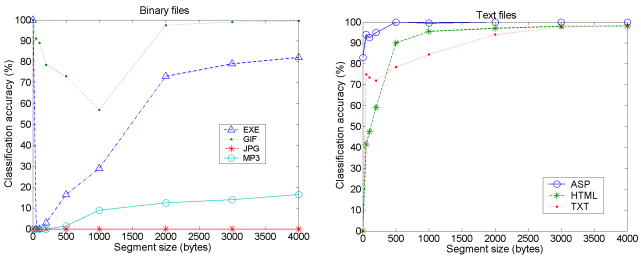


**Figure 2: Classification accuracy using the initial contiguous bytes as the sampled content.**

given algorithms for each sampled file type using the patterns obtained from the union operation . As seen from the table, among the file type, HTML was the least accurate file type to detect (73.5% accuracy using KNN). Our further investigation showed that HTML is often confused with ASP, TXT or XLS.

Figure 1 shows the average classification accuracies of the classifiers (using features obtained from union operations). The figure shows that KNN is the most accurate classifier among the tested algorithms. It produces about 90% accuracy using 40% of features that are obtained using union operation. It is also observed that If we use intersection operation, we can further reduce the number of features with little compromise of accuracy. For instance, we can achieve 88.45% accuracy using just 20% of features.

Since KNN achieves optimal accuracy, we used it (with 40% of byte patterns) in our experiments for the content sampling technique. Figure 2 shows the result when initial contiguous bytes were sampled. As the figure shows, the levels of accuracy are not high enough for many file types. File types belonging to the text group appear to yield accurate results. However, it is because that text files are usually small so our experiment used the most of the file content. The second approach (obtaining byte frequency of a file from random locations) gets similar results.

We have measured the processing time of KNN using Manhattan distance. It is found that it can save about 50% of time when using 40% of byte patterns.

## 4. CONCLUSIONS

Based on the empirical results, we conclude that the feature selection approach is highly effective as it acheives high accuracy using small number of features. Moreover, it can substantially save classification time. On the other hand, our initial experiment with the content sampling technique did not yield high accuracy.

In sum, the proposed approaches showed promising results even with 1-gram features.

## 5. REFERENCES


[1] Li, W. J., Wang, K., Stolfo, S., and Herzog, B. Fileprints: Identifying file types by n-gram analysis. In *Workshop on Information Assurance and security (IAW'05)* (United States Military Academy, West Point, New York, USA, June 2005), pp. 64–71.

[2] McDaniel, M., and Heydari, M. H. Content based file type detection algorithms. In *proceedings of the 36th Annual Hawaii International Conference on System Sciences* (January 2003), vol. 9, p. 332a.