Using NLP techniques for file fragment classification

Simran Fitzgerald
George Mathews
Colin Morris
Oles Zhulyn

File fragment classification

- Files can be stored in a fragmented manner on the disk
- When filesystem information is missing or corrupted, carving files from file fragments requires some automated way of classifying file fragments according to file type

Previous work I

- Classification using linear discriminant analysis [Calhoun&Coles '08; Veenman '07]
- K-nearest-neighbors (KNN) with nearest compression distance as distance metric [Axelsson '10]
- KNN with Euclidean distance, where file fragments represented by vectors of statistical measurements [Conti et al. '10]

Previous work II

- K-means clustering on histogram of byte values of prefix of file [Li et al. '05]
- Support vector machines on histogram of byte values in file fragment [Li et al. '10]

Our contribution

- Applying supervised machine learning techniques from NLP (natural language processing) to file fragment classification
- Experiments yield promising results

Data set

- Derived from freely available Garfinkel corpus (govdocs1)
- Uniformly sampled files such that would have at least 10 files made up of at least 10000 file fragments, for each file type
- Discarded first and last fragment of each file [Calhoun&Coles '08; Conti et al. '10; Li et al. '10]
- Uniformly sampled 9000 fragments for each file type to create data set

File types

- Selected 24 well-known file types from among the most well-represented file types among govdocs1 files
- Including low entropy (e.g. text) and high entropy (e.g. compressed) files
- TODO: Develop rigorous methodology for selecting file types of forensic interests

File types in Garfinkel corpus

File type	Total number of files	Approximate total number of fragments
pdf	231232	268199787
ppt	49702	251176072
txt	78285	99266791
jpg	109233	73326493
doc	76616	60692770
xls	62635	58754238
ps	22015	56580524
html	214568	25839084
gz	13725	17766643
xml	33458	16982435
...	...	...

File fragment size

- Generally, safe to assume file fragments are 512-bytes in size
- Without filesystem information, cannot safely assume larger size
- Smaller file fragments are harder to classify [Gopal et al. '11]

Support vector machines (SVMs)

- Represent file fragments as high-dimensional vectors, where each dimension is some “feature” of the file fragment
- During training, SVM takes feature vectors with known file types, and partitions feature space such that each partition corresponds to a file type
- During testing, check into which partition feature vector falls

Features: Unigrams and bigrams

- A unigram is a single byte within a file fragment
- A bigram is a pair of consecutive bytes within a file fragment
- Histogram of unigrams constitutes 256 features [Li et al. '05; Veenman '07; Li et al. '10]
- Histogram of bigrams constitutes 256^2 features [Gopal et al. '11]

Features: Shannon entropy

- Measure of uncertainty associated with a random variable [Shannon '48]
- Conti et al. ['10] found entropy of bigrams to be effective for classifying file fragments

Features: Hamming weight

- Total number of ones divided by total number of bits in file fragment [Conti et al. '10]

Features: Mean byte value

- Mean value of all the bytes in a file fragment [Conti et al. '10]

Features: Kolmogorov complexity

- Minimum amount of data needed to specify a file fragment [Kolmogorov '65; Lempel&Ziv '76]
- Can be approximated by compressing the file fragment (e.g. with bzip2 algorithm) [Seward '01; Veenman '07]

Features: NLP features

- Average contiguity between bytes (i.e. the average distance between consecutive bytes)
- Longest contiguous streak of repeating bytes

Experiments I

- Three experiments (k = 1000, 2000, 4000)
- Uniformly sample k file fragments for each file type to produce data set of 24·k file fragments
- Randomly partition data set into training and testing sets (in, roughly, 9-to-1 ratio)
- Ensure fragments from the same file do not appear in both the training and testing sets

Experiments II

- Train libsvm [Chang&Lin '11] on training set and test on testing set
- Repeat ten times, each time resampling the 24·k file fragments

Experiments III

- We used linear kernel [Li et al. '10], default parameters, and did not scale the feature vectors
- Optimal parameters and scaled feature vectors should yield better results
- SVM optimized for linear kernel (e.g. liblinear) can dramatically decrease training time

Results I

k	Average accuracy	Standard deviation
1000	47.5%	1.56%
2000	48.3%	3.11%
4000	49.1%	3.15%

Results II

- Prediction accuracy of 49.1%
- Baseline is 1/24 ≈ 4.17% (random chance)
- Axelsson ['10] achieved 34% on 28 file types
- Veenman ['07] achieved 45% on 11 file types
- Gopal et al. ['11] achieved about 33% using macro-averaged F_1 metric
- Using same metric, our accuracy is 48.0%

Results III

File type	Prediction accuracy
csv	99.7%
ps	98.5%
gif	95.8%
sql	94.9%
html	94.8%
...	...
ppt	13.6%
xlsx	6.8%
docx	5.5%
pps	4.7%
pptx	2.3%

Results IV

- Best performance on low entropy file fragments, and worst on high entropy file fragments, which agrees with previous work [Conti et al. '10; Li et al. '10]

Some more (preliminary) findings

- Histogram of bigrams provided most significant contribution to classification accuracy
- Histogram of unigrams, contiguity, and Hamming weight were also important (but to a much lesser extent)
- Take this with grain of salt

Conclusion & Future work

- Need to focus efforts on classifying high entropy file fragments
- Consider applying various machine learning boosting techniques, such as AdaBoost [Freund&Schapire '97]
- Consider first classifying according to broad categories (e.g. Encrypted vs. Machine Code vs. Text) [Conti et al. '10], and using a separate specialized classifier within each category