Learning Fast Classifiers for Image Spam

Mark Dredze
Computer and Information Sciences Dept.
University of Pennsylvania
Philadelphia, PA 19104
mdredze@seas.upenn.edu

Reuven Gevaryahu
Philadelphia, PA 19104
reuven@alumni.upenn.edu

Ari Elias-Bachrach
Philadelphia, PA 19104
ari@angelsofsecurity.com

Abstract

Recently, spammers have proliferated "image spam", emails which contain the text of the spam message in a human readable image instead of the message body, making detection by conventional content filters difficult. New techniques are needed to filter these messages. Our goal is to automatically classify an image directly as being spam or ham. We present features that focus on simple properties of the image, making classification as fast as possible. Our evaluation shows that they accurately classify spam images in excess of 90% and up to 99% on real world data. Furthermore, we introduce a new feature selection algorithm that selects features for classification based on their speed as well as predictive power. This technique produces an accurate system that runs in a tiny fraction of the time. Finally, we introduce Just in Time (JIT) feature extraction, which creates features at classification time as needed by the classifier. We demonstrate JIT extraction using a JIT decision tree that further increases system speed. This paper makes image spam classification practical by providing both high accuracy features and a method to learn fast classifiers.

1 Introduction

For nearly a decade, content filtering using machine learning has been a cornerstone of anti-spam systems (Sahami et al., 1998). Content filters, often based on naïve Bayes models, use features extracted from an email's text to classify a message as spam or ham. Over time spammers have attempted to outsmart content filtering by obscuring text, obfuscating words with symbols, and including neutral text to confuse filters. Each attack solicits new tricks from anti-spam researchers to restore filtering effectiveness. A new potentially devastating attack on content filters has emerged. Instead of obscuring the message's text, spammers have removed the text completely, neutralizing text analysis techniques. The spammer's advertisement appears in an attached image, readable to a human but hidden from the filter. Typically called image spam, the message text is now random, often extracted from web sources, such as news articles and message boards. Image spam has allowed spammers to design spam as CAPTCHAs against anti-spam systems. Since embedded text extraction and analysis is a difficult problem, filters cannot properly analyze these emails, allowing them to pass through to the user (Goodman et al., 2005; Secure Computing, 2006).

This work restores content filters through the creation of a new feature for content filters, namely *attached image is spam*. This feature allows the content filter to incorporate knowledge about image attachments into classification decisions. The new feature is generated by our system, which, given an image, predicts if the image is spam or ham based on simple and fast image tests. Our system performs on-par or beyond existing results in the literature.

While our system, as well as other approaches (Aradhye et al., 2005; Fumera et al., 2006; Wu et al., 2005), effectively classify images in a research environment, they present a logistical problem for production systems. Even with relatively fast feature tests, image processing is inherently slower than text feature extraction, making these research systems impractically slow for production use on critical systems. Worse yet, as image sizes increase, system speed will degrade.

We present a new algorithm designed to learn *fast* classifiers. Speed sensitive feature selection allows for the construction of features that are both fast and accurate, allowing image spam classification to function in a production environment. Additionally, we modify the standard implementation of learning systems to

yield further increases in classification speed with Just in Time (JIT) feature extraction, extracting features as needed on a per image basis. Our empirical evaluation shows our methods reduce computational cost of image processing to a tiny fraction of our baseline system, making practical deployment of systems a reality.

Our paper is structured as follows. First, we discuss image spam and describe the features we use for classification. We then present results from our system and compare them to previously published work. Next, we describe our speed sensitive feature selection algorithm, which yields a dramatic reduction in image processing time. Finally, we present JIT feature extraction to further reduce computation time. We then compare our approach to related methods and conclude with a discussion.

2 Image Spam Classification

A range of ad-hoc solutions to image spam proposed by the anti-spam community litter anti-spam blogs, including blocking gif images from known senders, blocking all gif images, using FuzzyOcr[1] and limiting advanced image processing to small images. Similarly, a number of research systems address this problem. Most rely on simple image processing (Aradhye et al., 2005; Wu et al., 2005) while others employ full optical character recognition (OCR) (Fumera et al., 2006). Most techniques attempt an understanding of the image contents, through embedded text detection, color analysis, or embedded text extraction. But even these techniques, which often rely on a one pass linear scan of the image, are slow since images are far larger than most emails.

Rapid changes in the composition of spam have spawned plugin learning systems where communities develop new rule modules, or features. One of the most popular anti-spam solutions is SpamAssasin and there are several sites hosting plugin rule modules. [2]. Our goal is to create a general purpose system compatible with a variety of anti-spam filters, such as SpamAssasin. Towards this end, we create a new binary prediction problem: Is this image spam or ham? The classification can then be fed into existing content filters as a feature. Others have followed this approach (Aradhye et al., 2005) and it has several advantages. First, it separates image classification from spam email classification, which is a difficult and well studied problem. Second, emails can contain multiple images and it is not clear how to combine them towards a single prediction (see Aradhye et al. (2005) for one approach). We treat each image separately, avoiding this difficulty. Finally, we do not commit to a specific content filtering system. Rather, we provide a single feature that can be integrated with any learning based anti-spam system.

We extend the methodology of text classification to image spam classification. Rather than using a complex analysis of meaning and intent, modern email text classification relies on simple features, including bag of words, header information and message structure (HTML). Classifiers model textual artifacts of messages, using the observation that spam messages tend to use the word "viagra" rather than understanding the intent of the message. We take a similar approach to image spam classification, mostly focusing on incidental properties of the image. As a result, our features are very fast since many do not require examining the image contents at all. In addition to these basic features, we include some more advanced features from the literature.

Our first group of features relies on metadata and other simple image properties.
File Format: The file format of the image from its extension, the actual file format (as determined from metadata) and whether they match.
File Size: The size of the image file in KB.
Image Metadata: All of the information contained in the image metadata, including whether the image has comments, number of images (frames), bits per pixel, progressive flag, color table entries, index value, transparent color, logical height and width, components, bands, etc.[3]
Image Size: The height, width and aspect ratio of the image.

The rest of our features perform a simple analysis using a single pass over the image.
Average Color: Computes the average red, blue and green color values for the image.
Color Saturation: Tests the color saturation of the images as described in Aradhye et al. (2005).
Edge Detection: A simplified partial implementation of a Sobel edge detector using 3 convolution matrices (Duda & Hart, 1973).
Prevalent Color Coverage: How often the most common color appears in the image. This is a simple test that may find solid backgrounds.
Random Pixel Test: Generate ten random colors

[1] A plugin for FuzzyOcr is available for SpamAssassin at `http://fuzzyocr.own-hero.net/`.

[2] See `spamassassin.apache.org` and `www.rulesemporium.com`

[3] For a full listing of the metadata fields in various image formats, see the links to detailed descriptions at `http://java.sun.com/j2se/1.5.0/docs/api/javax/imageio/metadata/package-summary.html`.

and check to see if they appear in the image. A picture is more likely to have a wider range of colors.

As is standard in many learning problems, we binarized our feature space by dividing real valued features into binary feature bins. Feature counts for each type are listed in Table 5.

3 Corpus

Email corpora are difficult to build due to the private nature of email communications. Some corpora exist for spam email classification but there is of yet no corpus for image spam. Therefore, we constructed our own image spam corpus.

For guidance, we investigated other approaches in the literature. Both Wu et al. (2005) and Fumera et al. (2006) construct an email spam corpus that include images. Both use emails from the SpamArchive corpus, a subset of which contain attached images. They each report different numbers of images, possibly because Wu et al. include HTML links to images. Additionally, Fumera et al. include a private email corpus. For ham mail, Wu et al. use the Ling-Spam corpus while Fumera et al. use Enron. However, neither of these legitimate mail corpora contain images. They justify their lack of ham images by classifying email directly. For our task, however, we require ham images.

Aradhye et al. (2005) share our task of directly classifying images and construct a pure image corpus. They collect 1245 spam images from a few hundred spam emails received by the authors. For ham images, they downloaded images from Google Images using the keywords "baby", "photo", "graphics" and "screenshot". Each one of these sets was used for a separate evaluation. We differ in our ham image collection for several reasons. A learning system may be able to discern a spam image from a picture of a baby or general photographs but not from a wide variety of images. Additionally, we have no indication that these images are representative of actual images in ham mail. In fact, we found our ham corpus to be significantly more varied, as we discuss below. We also felt that several hundred images was insufficient for a thorough evaluation.

3.1 Our Corpus

A clear definition of spam and ham images is difficult. Spam has been defined as unsolicited bulk email, but how can this definition extend to images? For example, many spam emails contain images without or with a minimal amount of text. Additionally, legitimate emails contain text images as advertisements for events or text in pictures. To avoid a tricky and subjective annotation, we applied a simple definition: If an image arrives in an email that is spam, it is spam. If it arrives in a ham message, it is ham. We rely on the conventional definition of spam and ham email. This standard gives a clear labeling of the data.

We constructed two spam datasets, one based on personal spam and one based on publicly available spam. For personal spam, we collected spam emails from 10 email accounts across 10 domains and a catch all filter on two domains over the period of one month. Every attached image (gif, jpg, png and bmp) was extracted for the personal spam corpus, including emails that contained multiple images. Next, we extracted the images from the subset of the SpamArchive corpus used by Fumera et al. (2006).[4] For ham, we collected email from two users from a two year time frame and extracted all attached images. Since our ham images come from actual user email, they are a more realistic sample. We were unable to build a public ham corpus since public email image data is unavailable.

Typically in learning tasks, duplicate examples are removed from the evaluation set so that a system does not learn on instances that appear in the test data. However, in many spam classification evaluations, duplicate or highly similar emails are included to reflect the real world nature of spam. In fact, we noticed that the data, especially SpamArchive, contain duplicate or highly similar images. In some cases the same image appeared twice, such as a corporate logo, but more commonly two highly similar images appeared with only minor changes, as is common in image spam. In keeping with the construction of standard machine learning corpora, we aggressively removed similar and duplicate images by constructing a signature for each image based on its dimensions and the values of 10 pixels drawn from the image. Images with duplicate signatures were removed. This removed both duplicate and similar images since spammers often use the same image with small variations, such as changing the background, the font of the letters, or adding random lines or patterns in the image. We also removed images smaller than 10x10 pixels since these are often used as blank spacers in HTML documents. In total, we had 2359 images in our personal ham corpus, 2173 images in our SpamArchive corpus, and 1248 images in our personal spam. These dataset are summarized in Table 1.

The personal spam archive contained numerous emails that were typical of image spam: text written on varying backgrounds describing stock market scams. The

[4]The authors would like to thank Giorgio Fumera for making this data available. SpamArchive.org has been shut down and is now a parked domain showing advertisements.

Corpus	*Unique Images*	*Total Images*
Personal Ham	2359	2550
Personal Spam	1248	3239
SpamArchive Spam	2173	9503

Table 1: A summary of the datasets used in our evaluations. We removed duplicates and similar images with a simple heuristic to construct the unique image version of each dataset.

SpamArchive corpus contained many advertisements, some without text. For example, some pictures of women without text were included in a spam email advertising an adult web site. Our decision to use real ham images revealed a wide variety of images beyond photographs. While many of the images were personal pictures, other images included company logos, flyers, clipart, advertisements for departmental and extra-curricular events, comics, etc. The diversity of our dataset creates a more realistic scenario to test our methods. We include a few sample ham and spam images from our corpus in Figure 1.

4 Evaluation

We evaluated our feature set with several learning models. We used Maximum Entropy (logistic regression in the binary case) which has comparable state of the art performance with Support Vector Machines (SVM) on a wide range of tasks. However, since most real-world spam systems rely on simpler generative models for classification we also included a naïve Bayes classifier. Discriminative models, such as Maximum Entropy, tend to outperform simpler generative models given enough training data (Ng & Jordan, 2002). We found this to be the case in our setting as well. We included a third classifier, an ID3 decision tree, to represent a different learning approach. All three algorithms were implemented with Mallet and we used the default settings unless otherwise noted (McCallum, 2002). We relied on the Java 1.5 imageio library for image processing and feature extraction.

Each classifier was evaluated 10 times using an 80/20 train test split on three permutations of our datasets, a) personal ham and spam, b) personal ham and SpamArchive and c) all data. Results were measured using both accuracy and the spam F1 score and appear in Table 2. We choose these metrics so as to provide a highly accurate feature to a content filter, similar to Aradhye et al. (2005), which can then be optimized for the necessary level of precision.

Our system performed very well on this task; Maximum Entropy accuracy exceeded 89% on all datasets. For personal ham and spam, performance improves to 98% accuracy. SpamArchive was more difficult. As expected, naïve Bayes performs worse than Maximum Entropy while the decision tree is a close second. These results seem to compare well with other approaches in the literature, although an accurate comparison is difficult both because of a lack of shared data and different evaluation methods. Accuracy results for binary image classification in Aradhye et al. (2005) varied in the low to upper 80% range for a SVM, whereas our comparable Maximum Entropy model varied from 89% to 98% on our three larger datasets.

Additionally, we observed that Wu et al. (2005) and Fumera et al. (2006) used SpamArchive images without removing duplicates or similar images, instead removing duplicate emails. However, as our analysis showed, there was considerable duplication in the SpamArchive data. As we discussed above, including duplicates or similar messages in spam corpora is common since it more accurately represents real world settings. For example, consider a content filter that correctly detects one spam message but misses a second. If all messages with similar (but not duplicate) content are removed from a corpus, system accuracy would be 50%. However, if real world spam distributions heavily favored emails of the second type, the actual accuracy of this system would be much lower. Therefore, we considered an evaluation of our system on all available data without removing duplicate or similar images an important test of its real world accuracy. Additionally, we sought a more direct comparison to other systems evaluated on the SpamArchive dataset that did not remove any duplicate or similar images. We re-evaluated our system on the full version of each of our datasets, which greatly increased the number of images in the SpamArchive dataset. Accuracy increased slightly on personal spam but few new images were added. However, our accuracy on SpamArchive increased to 97%, reaching a level comparable to personal spam (Table 3). This evaluation on the observed distribution of spam images demonstrates that our system is highly accurate in a real world setting.

4.1 Feature Analysis

To analyze feature predictiveness, we computed the mutual information for each feature with the target label (spam/ham). Another common technique for showing good features is to select features by model weight. However, this does not measure individual feature effectiveness; instead it shows one possible weighting of a large set of features working together. A listing of the top 24 features appears in Table 4. The top features were mostly taken from the metadata of the image, features that are simple and fast to compute.

Figure 1: A sample of the ham (left) and spam (right) images in our corpus. Many images are difficult to judge as spam or ham. Corporate logos and comics are often included in ham messages while the picture of the woman is from a spam advertisement. We rely on a labeling of the email itself to simplify this decision.

Model	*Corpus*	*Accuracy*	*F1*
MaxEnt	PHam/PSpam	.98 (.004)	.97 (.005)
	PHam/SpamArc	.89 (.008)	.89 (.009)
	All	.91 (.006)	.93 (.006)
NB	PHam/PSpam	.88 (.015)	.85 (.019)
	PHam/SpamArc	.76 (.011)	.74 (.011)
	All	.80 (.007)	.83 (.007)
DT	PHam/PSpam	.97 (.001)	.95 (.012)
	PHam/SpamArc	.85 (.015)	.84 (.015)
	All	.87 (.020)	.89 (.010)

Table 2: Results for our system using three different classifiers: Maximum Entropy (MaxEnt), naïve Bayes (NB) and an ID3 Decision Tree (DT). Each classifier was evaluated on three datasets: Personal Ham (PHam) with Personal Spam (PSpam), Personal Ham with SpamArchive (SpamArc), and all data together (All). Results are averaged over 10 trials using an 80/20 train/test split with standard deviation in parenthesis. Similar/duplicate images were removed from this evaluation.

Model	*Corpus*	*Accuracy*	*F1*
MaxEnt	PHam/PSpam	.99 (.003)	.99 (.003)
	PHam/SpamArc	.97 (.003)	.98 (.002)
	All	.97 (.004)	.98 (.002)
NB	PHam/PSpam	.89 (.011)	.91 (.009)
	PHam/SpamArc	.83 (.008)	.89 (.005)
	All	.85 (.004)	.91 (.003)
DT	PHam/PSpam	.96 (.007)	.96 (.007)
	PHam/SpamArc	.93 (.007)	.95 (.004)
	All	.93 (.007)	.96 (.004)

Table 3: Results for our system using the same tests as in Table 2 only similar/duplicate images are included.

5 Learning Fast Classifiers

We now turn our attention to a related problem, image classification speed. While our features do not include advanced processing tasks, they are still more time consuming than text processing. Additionally, anti-spam research is fast paced due to its adversarial nature. We expect spammers to develop new and unforeseen ways to circumvent existing image classification systems with clever image tricks. In the future, it may be necessary to include more complex, and slower, image processing. Therefore, we not only want to learn accurate classifiers, but also *fast* classifiers.

Classification systems typically have two stages, feature extraction and classification. For a linear classifier, classification involves a multiplicative computation with the weight and feature vectors, a fast operation. Feature extraction can be much more time consuming, especially when features depend on complex tests. Text extraction is very fast; image processing is slow by comparison. This difference can make image classification impractical for production systems.

Not all features are needed for classification. A common technique in machine learning is feature selection, learning with a subset of features, which can reduce the feature space dramatically. Only the most predictive features are kept since a smaller set may be simpler

pixelAspectRatio=0	number_of_bands!=1
transparentClrFlg=fls	meta_green=0
imageTopPosition=0	meta_red=0
imageLeftPosition=0	meta_blue=0
interlaceFlag=false	backgroundColorIndex=0
userInputFlag=false	htableId=0
sortFlag=FALSE	acHuffTable=0
file_extension=jpg	samplePrecision=8
disposalMethod=none	approxHigh=0
index=1	QtableSelector=0
index=0	componentSelector=1
small_edge_detector	dcHuffTable=0

Table 4: The top binary features as computed by mutual information with the target label. Most features are taken from the image metadata.

to learn. Since feature selection reduces the feature space, it reduces the amount of time spent in feature extraction, enabling faster production performance.

The most common technique for feature selection is greedy inclusion by mutual information, an information theoretic measure. Mutual information represents the knowledge about a random variable (the label spam/ham), given another random variable (a feature). The mutual information for two discrete random variables is defined as:

$$MI(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) log \frac{p(x,y)}{p(x)p(y)}$$

where $p(x, y)$ is the joint probability of events x (the feature) and y (the label), drawn from the event spaces X and Y. Feature selection computes the mutual information score for each feature with the label in the training data and selects the top k features. This does not guarantee the best subset but works well in practice. While other feature selection methods may work better on this task, a penalty term can be added to those as well.

While this selection algorithm will yield faster extraction by reducing the number of features, we can modify it to be sensitive to the computational cost of features. Consider two features, one highly accurate but slow and another moderately accurate but fast. Normally, feature selection would favor the former because of accuracy, but for a fast system the latter is superior. We augmented the feature selection scoring function to include timing information, yielding $score(x) = (1 - \alpha)(1 - MI(x, y)) + \alpha \times t_x$, where t_x is the average time to extract feature x in milliseconds in the training data and α is a tradeoff between speed and accuracy. We greedily order features by our new scoring function and select the top k. We will refer to our new algorithm MI_t.

Feature Type	*Feats.*	*Time (ms)*
Average Color	44	162.4
Color Saturation	10	163.0
Edge Detection	32	12.7
File Format	332	1.1
File Size	6	1.1
Image Metadata	7195	1.4
Image Size	18	1.0
Prevelent Color Coverage	26	162.7
Random Pixel Test	11	167.3

Table 5: The number of binary features and average time for each feature type. Feature types were broken into 23 tests for feature selection.

Computing t_x presents a problem for binary features since a single feature test, such as *image_height* can generate multiple binary features (*image_height<100*, *image_height<200*, etc.) Furthermore, groups of features are typically computed together since they rely on a shared processing task, such as extracting all the information from the metadata. Consistent with these behaviors, we captured timing information for feature *tests*, where a feature test can generate multiple binary features (ex. all the metadata features are generated by the metadata test and all of the edge detection features are generated by the edge detection test). We used a total of 23 feature tests. Average time for each feature was the generating feature test's average time in the training data. If a feature was greedily selected, all other features produced by the same test became "free" and their time went to 0. A list of the feature types, number of binary features, and average time per type is shown in Table 5.[5]

6 Evaluation

We compared our speed sensitive feature selection algorithm (MI_t) against two baselines: our existing system (baseline) and standard mutual information feature selection (MI), equivalent to $\alpha = 0$. For MI and MI_t we set $k = 500$ and for MI_t we set $\alpha = .1$. For each trial, we measured system accuracy and average extraction time per image. Timing information was estimated on an Intel Xeon 3.20 GHz CPU with 4 GB of RAM using standard Java 1.5 timing functions. For clarity, Table 6 shows results for Maximum Entropy classification only since timing information is identical for all classifiers. For regular feature selection, we observe a speed gain of 20% since fewer features are used.

[5]We did not capture processing time that was common to all features, such as reading the image from disk, since we are concerned with the relative speed reduction between approaches. Such information is difficult to measure without an instrumentation of the Java library itself.

However, our speed sensitive method yields a dramatic speed increase, reducing computation time from seconds per image to just 3-4 milliseconds, about 1/1000 the time of our baseline system. While performance drops slightly, about 1-3 percentage points, our system is now fast enough to be practical in a production environment.

In further experiments, α revealed itself insensitive to tuning because of our feature set's composition. Our features tended to be either very fast or slow. Including any amount of timing information eliminated the slow features (edge detection, color saturation, etc.) and left mostly metadata features. Increasing α only slightly reordered the remaining fast features. A wider variety of advanced slower features may increase the sensitivity of α. Given this information, it may be possible for our feature set to manually eliminate features. Since we know which features are slow we can simply exclude them manually. However, as features become more we obviously prefer an automated method. As image spam becomes more complex, it will become difficult to manually find a balance between speed and accuracy. Our approach allows a system designer to include all possible features and then automatically tune speed and accuracy, rather than doing so manually.

7 Just in Time Feature Extraction

We now explore a complementary approach to feature selection for increasing system speed. Regardless of the size of our feature set, not all features are needed to classify a single image. For example, there may exist a high precision feature that identifies an image as spam if the feature is present but indicates no identification if absent. Similarly, the system may need to observe just a few features of a spam image to determine that it is spam. In content filters, a single feature, such as a blacklisted sender, is sufficient for correct classification.

This intuition is best captured in a decision tree model. A decision tree classifier is a binary tree where each node corresponds to a feature and each edge corresponds to the feature value (e.g. positive or negative). Classification begins with the root node, moving to the left or right child depending on the feature's value. The classifier recurses down the tree until reaching a leaf, which contains the predicted label. For an n node tree, a classifier need only examine the $log\ n$ features on the traveled path on average.

We present Just in Time (JIT) feature extraction, which only extracts the features needed on a per image basis. This replaces the standard two stage classification pipeline (extraction and prediction), with a single prediction stage that runs extraction only when the features are needed. Since only a subset of features may be needed for prediction, this reduces the time spent on extraction. JIT feature extraction does not depend on feature selection as a JIT classifier can reduce processing time no matter the feature space.

We evaluated JIT extraction by constructing a JIT decision tree classifier whereby the scoring method used by the decision tree is MI_t. This ensures that faster features appear higher in the tree. Other standard modifications to decision tree creation can be used, but we limit our study to our single modification. All other decision tree settings are the same as above (the default used in Mallet.) Features are extracted when a feature's node is reached and processing time per image is computed as in Section 5. We chose to evaluate JIT extraction using a decision tree since it is intuitive to understand. However, a similar approach exists for linear classifiers, like Maximum Entropy and naïve Bayes. By sorting the weight vectors in non-increasing order, a classifier could compute when the predicted label would not change through inclusion of unseen smaller weighted features.

We constructed a JIT decision tree and evaluated it on the feature sets produced by our three feature selection algorithms: all features (baseline), MI and MI_t. For baseline and MI, we used mutual information and for MI_t we used our speed sensitive measure to select the nodes in the ID3 decision tree learning algorithm. For each of our ten trials we computed the average feature extraction time per test image for a normal decision tree and our JIT decision tree. Table 7 shows the timing information for each feature set for both a normal decision tree and our JIT decision tree. We do not show accuracy numbers since JIT extraction does not affect classifier accuracy. In every case, JIT extraction reduced the processing time per image, even for our already fast system. Processing time per image approaches 2 milliseconds, which is close to the resolution of instrumentation of the Java runtime environment.

8 Related Work

Several attempts have been made to address image spam. Wu et al. (2005) and Wu (2005) use a combination of visual features of the image to detect characteristics common in spam images, such as embedded text and banner features. Their features are combined with message text features and a one-class SVM distinguishes when ham emails are outside the spam class. Similarly, Aradhye et al. (2005) develop features to detect embedded text and certain background types consistent with spam. They use an SVM to discern between a collection of ham and spam images. Fumera

	Baseline		*MI*		MI_t	
Corpus	*Acc.*	*Time*	*Acc.*	*Time*	*Acc.*	*Time*
PHam/PSpam	.98	5008	.97	3156	.96	4.4
PHam/SpamArc	.89	4660	.87	4390	.87	3.9
All	.91	3629	.88	2915	.88	3.8

Table 6: Results for feature selection using the Maximum Entropy classifier. For each dataset, the accuracy and average time per test image in milliseconds is shown for our three methods: All features (baseline), feature selection (MI) and speed sensitive feature selection (MI_t). We removed duplicate and similar images for this evaluation.

	Baseline		*MI*		MI_t	
Corpus	*Norm.*	*JIT*	*Norm.*	*JIT*	*Norm.*	*JIT*
PHam/PSpam	4934	1213	3209	1455	4.4	3.6
PHam/SpamArc	4660	11.5	4389	10.7	3.9	2.6
All	2730	12.6	2119	6.1	3.5	2.5

Table 7: Results for a decision tree classifier using our three methods of feature selection compared with a JIT feature selection decision tree. For each dataset, average processing time per test image in milliseconds is show for a normal (Norm.) and JIT classifier averaged over 10 runs. In each case, the JIT classifier speeds up the baseline method.

et al. (2006) take a different approach to the problem, processing each spam image using OCR to extract embedded text instead of surface features. Extracted terms are combined with terms in the message and fed into a SVM for classification.

These approaches use some form of embedded text detection since most spam images contain text. Text detection and extraction from images and video has been studied extensively in the vision community (Agnihotri & Dimitrova, 1999; Du et al., July 2003; Gllavata et al., 2003).

Other approaches rely on existing alternatives to content filtering, such as reputation based filtering (Secure Computing, 2006). Additionally, they develop an image fingerprinting technique to identify similar text in multiple images.

To our knowledge, we present the first algorithm for speed sensitive feature selection. However, others have considered learning with limited resources. Viola and Jones (2002) use a method similar to JIT feature extraction for real time object detection. A set of high precision features test for the presence of an object in an image. Boosting orders the features in a cascade, running them sequentially until one of the rules fire. This setup enables real time recognition since most of the tests do not run for correct detection. Other work has focused on developing learning algorithms with memory constraints (Dekel et al., 2004).

9 Conclusion

We have presented a feature set that accurately identifies image spam across multiple datasets and classification models. The prediction from our system exceeds 90% accuracy, achieves 99% accuracy on our personal spam and ham messages, and can be used to enhance existing content filters for more robust spam classification in the presence of image spam. Additionally, evaluations on data reflecting a real world distribution over spam images yielded upwards of 97% accuracy. Our results compare favorably to existing systems and offer a new approach towards the problem. Additionally, we presented two methods to improve classification speed. First, we modified a popular feature selection algorithm to be speed sensitive. Our new algorithm reduces the feature set so that overall performance is maintained and computation time per image is greatly reduced. We then introduced JIT feature extraction, which selects features at test time for classification. This method does not affect system performance but greatly reduces the average processing time per image. We demonstrated JIT feature extraction by creating a JIT decision tree. Combining our two methods yields a highly accurate system that processes images in 2-3 milliseconds instead of 3-4 seconds. Overall, our approach makes real time classification of image spam a reality, both in terms of effectiveness and practicality. Despite the efficacy of our features on a range of spam over several years, we anticipate that spammers will eventually circumvent our features, like most spam systems. However, our feature selection methods ensure that the most effective and fastest features are used, including features developed in future systems.

10 Acknowledgments

The authors would like to thank all those who supplied spam to our corpus. Dredze is supported by a NDSEG fellowship.

References

Agnihotri, L., & Dimitrova, N. (1999). Text detection for video analysis. *Proc. IEEE Workshop on Content-Based Access of Image and Video Libraries. CBAIVL 99*.

Aradhye, H. B., Myers, G. K., & Herson, J. A. (2005). Image analysis for efficient categorization of image-based spam e-mail. *Proceedings of the 2005 Eighth International Conference on Document Analysis and Recongition (ICDAR '05)*.

Dekel, O., Shalev-Shwartz, S., & Singer, Y. (2004). The power of selective memory: Self-bounded learning of prediction suffix trees. *NIPS*.

Du, Y., Chang, C.-I., & Thouin, P. D. (July 2003). Automated system for text detection in individual video images. *Journal of Electronic Imaging*, *12*, 410–422.

Duda, R., & Hart, P. (1973). *Pattern classification and scene analysis*. John Wiley and Sons.

Fumera, G., Pillai, I., & Roli, F. (2006). Spam filtering based on the analysis of text information embedded into images. *Journal of Machine Learning Research*, *7*, 2699–2720.

Gllavata, J., Ewerth, R., & Freisleben, B. (2003). A robust algorithm for text detection in images. *Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis*.

Goodman, J., Heckerman, D., & Rounthwaite, R. (2005). Stopping spam: What can be done to stanch the flood of junk e-mail messages? *Scientific American*.

McCallum, A. K. (2002). Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Ng, A. Y., & Jordan, M. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *NIPS*.

Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998). A bayesian approach to filtering junk email. *AAAI Workshop on Learning for Text Categorization*.

Secure Computing (2006). Image spam: The latest attack on the enterprise inbox. http://www.securecomputing.com/image_spam_-WP.cfm.

Viola, P., & Jones, M. (2002). Robust real-time object detection. *International Journal of Computer Vision*.

Wu, C.-T. (2005). Embedded-text detection and its application to anti-spam filtering. Master's thesis, University of California- Santa Barbara.

Wu, C.-T., Cheng, K.-T., Zhu, Q., & Wu, Y.-L. (2005). Using visual features for anti-spam filtering. *IEEE International Conference on Image Processing*.