Improved Training for DLP Classifiers

Kushal Tayal (ktayal@stanford.edu)
Computer Science Dept., Stanford University
(With guidance from Michael Hart, Symantec Research Lab)

I. Introduction

Businesses, governments, and individuals leak confidential information, both accidentally and maliciously, at tremendous cost in money, privacy, national security, and reputation. Several security software vendors now offer "data loss prevention" (DLP) solutions that use simple algorithms, such as keyword lists and hashing, which are too coarse to capture the features what makes sensitive documents secret. Very few have used machine-learning techniques to identify sensitive documents. In this project, we plan to explore further machine learning possibilities for document protection.

When training classifiers for Data Loss Prevention, there are two fundamental problems that must be overcome:

1. Building a set of non-secret (negative) documents is nontrivial. Simply collecting the organizations public media is not sufficient to train a robust text classifier.
2. The solution needs to be light enough to be deployed at end nodes with minimal memory and computational requirements.

Most of the DLP products are set up and monitored by administrators. Relying on an administrator to provide all possibly relevant negative documents is an untenable proposition because we cannot assume they have significant experience in either Machine Learning or corpora development. In fact, poor collection of negative training may result in a classifier that suffers terrible performance.

II. Problem Definition

Some of the problems with using an administrator provided negative set, or in general, a manually constructing a negative set, are that of over fitting, high apriori expectation towards positive (private) class, etc. A paper by Michael Hart, Pratyusa Manadhatam, Rob Johnson titled "Text classification for Data Loss Prevention" addresses over-fitting but does not necessarily curate better training documents that have meaningful overlap with positive documents (e.g. negative documents that share features that are important to the identity of positive documents).

The goal of this project is to explore the utility of a universal negative set to counteract the flaws of handcrafted negative sets and result in a classifier that is robust (i.e. low false positive rate) against any negative document, regardless if it originates from the enterprise. We wish to amass a large amount of negative training documents (those that are public and accessible) and want to seek ways to both:

- Intelligently collect documents from this universal negative set to improve performance in comparison to a handcrafted negative set
- How to utilize different collection strategies to ultimately improve the classification performance

One more ambitious goal that we plan to investigate in tandem with the above is to come up with a generalized approach to identifying which features are prone to over-fitting and how to ameliorate this situation.

III. Data Collection

There is a need for good positive sets as well as universal negative sets. Positive sets are documents that are confidential in nature. As it is not possible to get hold of actual confidential documents, we collected some data sets from web, which were confidential at some point of time in the past but are now public. The following is the list of corpora we collected:

1. SEC filings (10 companies)
2. Company Web Data (mined public website of companies)
3. Enron email set, Sarah Palin emails
4. JKF - Khrushchev, WWII (Naval) message exchanges
5. Medical - PMC (journals), Medline (abstracts), UPMC (reports)
6. Federal Cases, Legal documents
7. Technical User Manuals
8. IRS Forms data set
9. Wikipedia (universal negative set)
10. Freedom of Information Act

These data sets are good proxies for the positive (private) and negative (public) data sets. SEC filings can serve as good positive sets for a company and the website data crawled for that company is a good manually gathered negative set that can be generated for it. However, for certain datasets like JFKs letters, medical journals, etc. there does not exist good sources of negative training documents and therefore an administrator could not simply go to a website and download them. The work presented in this report intelligently collects negative examples.

IV. Pre-Work

We had previously done some work in terms of analyzing the datasets for outliers and under-represented topics in the dataset. We used LDA based clustering using the tool mallet. We chose after multiple trial experiments, to cluster the data sets into 10 clusters and return 10 keywords per cluster. Based on the keywords returned for each cluster and the size of the cluster, we can determine if the cluster is an outlier.

Results:

Corpora	**Size**	**Discarded clusters**	**Size after**	**What was discarded**	**Significant Keywords**
IRS-Forms	2015	2	1796	Non-English files	form, tax, information, line, irs
public-ford	63	2	48	Twitter feeds Adobe download instructions	trailer, vehicle, tow, ford, cars
sec-chevron	284	1	279	Illegible data	chevron, board, stock, quarter, production
user-manuals	1824	2	1523	Non-English manuals Generic web page data	press, button, mode, model, data, virtual
public-symc	306	2	285	Generic license descriptions Vague documents	software, licensed, symantec, verisign, security

We have presented the results for a few of the datasets. The clustering exercise helps learn outliers (IRS-forms), transcription errors (sec-chevron), topically incoherent sets (public-ford). At this stage, the outlier sets were determined by manual inspection of keywords and the number of documents in that cluster. We will explore later ways to automate this.

V. Negative Set Generation

We need a way to generate negative sets that can be good negative sets in comparison to the positive set we have and the machine learning process. A negative set can be considered good, if it contains representative usage of features with respect to non-confidential documents as well as good counter examples of language found in the positive documents. Clarifying further, we would want documents in negative set to have similar data as in the positive set (e.g. company name, product names, etc.), however these are public documents as opposed to the positive (private) set. To recap, it would be incredibly expensive to generate high quality examples manually. The user would not be in a position to provide a good negative set having these qualities, and even if he does, the negative set could in high probability not be representative of the negative documents that will be tested after learning the model.

To carry on this experiment, various different negative sets can be used -

1. A hand-constructed negative set using the pages crawled from the company website
2. A random dump of Wikipedia articles
3. Category specific Wikipedia document dump (Using the Categories as defined under Wikipedia)
4. Keyword search based Wikipedia document dump (Search Wikipedia for documents having specific keywords. These keywords can be used from the set of keywords returned from the previous clustering experiment)

To move ahead with searching the Wikipedia for specific keywords, we used Lucene to index Wikipedia articles. Lucene has certain search query syntax rules that can improve the search quality. For initial runs, we generated the search query manually by looking at the prominent clusters formed after clustering and using some prominent keywords from it.

VI. Automating Outlier detection & Negative set generation

To facilitate the use of the above technology in the DLP product scenario, and to eventually assist the user to automate things without manual inspection of clusters and keywords to determine outliers, or different topic subsets, there is a need to develop an algorithm for it.

We used the clusters output by LDA, and looked at the overlap of keywords per pair of clusters. Then each cluster is paired with the cluster with which they have highest overlap (ranking system). If we treat the clusters as nodes, and this pairing as an edge between two nodes, this process forms a forest. I then pick the tree in the forest having the highest number of total documents and treat the rest as outliers or topically different data sets.

```
Dataset: user-manuals-10

0      236    model data time models analysis al level based prior
1      249    www user http information pdf search title manual pdfride
2      437    press button mode menu set camera select image power
3      212    click select file page user text menu button list
4      29     data register cache instruction bit processor user system address
5      200    virtual server system machine user network windows page file
6      213    press device click phone key select menu tap screen
7      113    bit register data set mode address pin user table
8      52     de la es el en se ca le del
9      83     file data number set command user register variable line

0 : [0] = 236
1 : [1] = 249
2 : [2, 3, 5, 6] = 1062
4 : [4, 7, 9] = 225
8 : [8] = 52
```

As we can see from the above result, there are 5 trees in the forest and the tree [2,3,5,6] forms the most significant one. Inspecting the keywords returned for these clusters also verifies this. The other trees have some level of topical incoherency with the selected tree. For instance, tree [8] has all non-english documents, tree [4,7,9] are all user-manuals but are more about system level products as is evident from the keywords of those clusters. Tree [1] is also a clear outlier set with some form of generic web content in that cluster.
To summarize, we first use LDA to divide the dataset into various clusters (fixed to 10 in my testing) and then use the algorithm described above to combine related clusters. This helps identify most significant sets and outlier sets. This helps us divide the data in a meaningful number of clusters rather than a pre-determined number of clusters.

The next step to automate is to build the keyword search based Wikipedia negative set. We now use the most significant cluster set, and look at the keywords for those clusters. We develop a ranking system to score the keywords based on its importance to the set and the occurrence of it across these clusters. We then select the top 4 keywords and generate the Lucene query using a combination of all words with a boost multiplier calculated using the ranking score of the involved words.

```
press : 18.0
click : 16.0
select : 15.0
menu : 12.0
(press AND click AND select AND menu)^3.39 OR (press AND click AND select)^2.72
OR (press AND click AND menu)^2.56 OR (press AND select AND menu)^2.5 OR (click
AND select AND menu)^2.39 OR (press AND click)^1.89 OR (press AND select)^1.83
OR (click AND select)^1.72 OR (press AND menu)^1.67 OR (click AND menu)^1.56 OR
(select AND menu)^1.5
```

VII. Machine Learning

We used Berkshire Hathaway data set as a test case. The SEC filings forms the positive (private) class, and as from above, the negative class could be one of the following –

- Website data – 329 documents
  (consisting of Reports, Shareholder Letter, News feeds)
- Random dump of Wikipedia – 1000 documents
- Category-based – 131 documents
  (Categories used are 'financial accounting', 'financial economics' and 'financial journals')
- Keyword search – 620 documents
  (Search query - (berkshire AND notes AND financial AND exchange)^2.56 OR (berkshire AND notes AND financial)^2.06 OR (berkshire AND notes AND exchange)^2.06 OR (berkshire AND financial AND exchange)^2 OR (notes AND financial AND exchange)^1.56 OR (berkshire AND notes)^1.56 OR (berkshire AND financial)^1.5 OR (notes AND financial)^1.06 OR (berkshire AND exchange)^1.5 OR (notes AND exchange)^1.06 OR (financial AND exchange)^1)
- Wikipedia Mix (random + category + selective): 1751 documents
- All Mix (Wikipedia mix + hand crafted): 2080 documents

Among the four base data sets, we would expect good performance with the hand-constructed negative set as that is what ideally the scenario would be when the VML rule is being used in our product. The random Wikipedia dump would be expected to perform average/mediocre. So these can serve as our boundaries and we can compare the performance of the remaining negative sets with respect to these.

We decided to use SVM as the classification algorithm and used "weka" for doing the test runs. The 60-20-20 strategy of building a training, cross-validation and testing set is used to decide the vectorization settings and optimal feature selection settings.

We used the following settings on weka-

a. StringToWordVector (lowercase, Snowball Stemmer, Alphabetic Tokenizer, Stop-word list)
b. Used LibLinear as the classification algorithm and ChiSquaredAttributeEval with a Ranker threshold set to "11".

Results: (TP = True Positives, TN = True Negatives)

Train\Test	**k-cross**	**private (TP)**	**public (TN)**	**wiki-category (TN)**	**wiki-random (TN)**	**wiki-search (TN)**
public	98.6	100	73.61	59.5	89.7	39.67
wiki-random	99.92	98.63	55	97.7	100	100
wiki-category	99.77	100	49.4	94.81	51.5	32.9
wiki-search	99.89	98.63	57.14	75.57	31.3	94.03
wiki-mix	99.95	98.63	77.81	100	100	100
all-mix	99.27	95.89	100	100	100	100

We observe from the above results is that the True Positive rate is almost 100 for the all the negative sets. The True Negative (or False Positive) rate differs highly across the negative sets.

The principle we want to test with this experiment is to check tolerance of the classifier to company's public web documents that would be very similar to the private documents. The classifier is trained without knowledge these documents. For this reason, we concentrate on the public test set column. We observe from the above numbers, that it performs fairly well with the mixture of Wikipedia document sets. Assuming we have no negative set provided by the customer, we can use a combination of random, category and search based Wikipedia articles as a negative set. This provides tolerance against all generic Wikipedia documents, as well as 78% tolerance against the expected negative set.

To evaluate further, we performed testing against the *Gutenberg* data set (a set of books) and the *sec-latest* data set (latest dump of SEC filings – no specific company) to check the robustness of the *wiki-mix* negative training set case. *Gutenberg* set got classified as public straight away, but *sec-latest* showed a poor accuracy of 28.4%. As there were no SEC type documents used in the negative set while training, this is expected. As further evaluation, splitting the *sec-latest* set 25-75 as test and train sets and then adding the 75% into the wiki-mix negative set for training, gave 100% accuracy on the 25% test set. In the traditional use of DLP product, the user provides a negative set and the best negative set possible would be to use the handcrafted public negative set. Testing *Gutenberg* against the *public* negative training set gave a very poor accuracy of 5.26% and testing *sec-latest* gave an accuracy of 17.2%. The Wikipedia negative set strategy already performs better than the traditional strategy. We got similar performance difference with some other generic public data from the data sets collected earlier.

VIII. Conclusion

Using a mix of these automatically generated negative sets works out really well. It is robust against the tricky public negative documents and also against other public documents not used in training.

We can use LDA clustering and the automation algorithm to help identify logically correct number of topics in document set. This eventually helps understanding the data set better as it can unearth outliers, transcription errors, subtopics, underrepresented subtopics, etc.

IX. Future Work

We have done some analysis to examine the prominent features and also the overlap of features between training and test sets. There was a ~60% overlap of features and we wish to explore the effects of this. Another approach in terms of feature analysis is to utilize the features only from the positive set and not use features from the negative set. This will avoid over-fitting to the exact negative sets used for training and possibly improve robustness.

One possible alternative to the strategy of mixing the various negative sets is to move in the direction of ensemble classification. We already have the framework ready for it ready and are in process to evaluate the results and dig deeper.

Lastly, we also want to study some cheaper ways to perform clustering (probably distributed ways), as LDA is expensive. One way to tackle this is to use random projections or reflective random indexing.