Information Prioritizing: Ranking Transactions to Detect Anomalies

Jan Flores
Polytechnic University of Puerto Rico, Hato Rey, Puerto Rico, USA, jan_flores_guzman@yahoo.com

Alfredo Cruz
Polytechnic University of Puerto Rico, Hato Rey, Puerto Rico, USA, alfredcross@gmail.com

ABSTRACT

This paper focuses on the problem of prioritizing information. More specifically, it deals with applying the concept of detectors to help detect anomalies within a collection of individual items. The motivation for this work is the development of a real-world application to prioritize a log of transactions. The application should aid system administrators in detecting unusual behavior patterns that may indicate potential security risks. We discuss information management concepts such as information retrieval, filtering and categorization, to distinguish them from information prioritizing. Then, we delve into information prioritization concepts, application development details and the analysis of experimental results. After analysis, it is concluded that equivalence classing and the independence surrogate are critical for the naïve detector to become ideally suited for identifying abnormal behavior patterns within a set of transactions.

Keywords: anomaly detection, information management, prioritizing information, information ranking

1. INFORMATION EXPLOSION AND THE WORLD WIDE WEB

The Conference Proceedings will be produced directly from the camera-ready manuscripts received from authors. Therefore the authors should try to produce their paper, as closely as possible to this model paper. In today's world of global communication and ubiquitous connectivity, we are on the midst of what some call an "information explosion". Information explosion is the fast-paced increase in the amount of available information and the consequences this has. Of particular interest are the challenges that this information explosion can pose in terms of information management. If not properly managed, information explosion can lead users into problems like information overload. Alvin Toffler, American writer and futurist, popularized the term "information overload" to describe the difficulty of understanding issues and reaching decisions while in the presence of a substantially large amount of information (Toffler, 1965). The concept of information overload predates the public use of the internet. The advent and widespread proliferation of internet use have brought the information explosion and overload problems into the forefront of modern society. As information available on the internet and in private organization's computerized information systems has grown, the need for effective search and filtering mechanisms has evidenced itself time and time again.

Since the early 1990s, the community of internet users has been aware of the crucial need to search and categorize the information available at user's disposal. It was from this need that Archie was developed in 1990 by a group of students from McGill University in Montreal. Archie is considered to be the first Internet search engine (Li, 2002). Flash forward to 1996. Larry Page and Sergey Brin, graduate computer science students at Stanford, began work on an internet search engine called BackRub. After some time, the team settled on a new name for their product, Google. The name was picked to convey their "mission to organize a seemingly infinite amount of information on the web." On December 1998, a little over two years since its creation, PC Magazine recognized Google as the search engine of choice. The magazine reported that Google "has an uncanny knack for returning extremely relevant results." Google's success with PageRank, a critical algorithm in their search technologies, has

been readily apparent. Google's web search has become the leading search engine, consistently garnering approximately from 70% to 80% of online users' searches (Brandt, 2004).

PageRank allows Google to prioritize information in a highly accurate manner, saving users valuable time and effort when looking for information online. Increasing volumes of information and documents require effective management. What Google proved with their web search technologies is that proper management of information can greatly increase the value of information resources. Google achieved success in web search by implementing information retrieval techniques that allowed users to retrieve relevant sets of results from an extremely large information resource, the World Wide Web. The concept behind Google's PageRank algorithm is simple to understand. In very broad terms, the PageRank algorithm seeks to assign each document in a set of linked documents a numerical weight based on each of the document's relative importance within the set. This is the basic concept behind information prioritizing.

2. INFORMATION MANAGEMENT

The following are brief descriptions of various domains of information management related to information prioritizing. This research has served as the basis for this software development project.

2.1 INFORMATION RETRIEVAL

Information retrieval is perhaps one of the main and most extensive domains in information management research. Information retrieval is concerned with finding data of interest in a data stream. Information retrieval frequently relies on other specialized techniques such as categorization and prioritization to aid with accuracy and performance.

2.2 INFORMATION FILTERING

Information filtering deals with the filtering of an information source in response to one or more preferences. Instead of finding data of interest in the data stream as with information retrieval, information filtering is concerned with the removal of data from the stream that does not conform to profiles. Belkin and et al. introduce both, information retrieval and information filtering. The article explains in detail the differences between information filtering and information retrieval first by explaining what constitutes each. Then, it goes into what makes each one different from the other.

2.3 INFORMATION CATEGORIZATION

Information categorization or, more specifically, text categorization deals with the automatic categorization or labeling of documents. Document contents are evaluated to obtain potential text labels that categorize the document appropriately. This serves for information retrieval and filtering purposes. With regards to the processing required to perform text categorization, it has many similarities to the processing required for other text problems such as information prioritizing. There are a number of learning algorithms that prove useful not only for text categorization, but for other purposes as well (Yang, et al., 2000).

2.4 INFORMATION PRIORITIZATION

The main focus of the rest of this paper is the study of prioritizing information. Prioritizing information is a simple concept. It deals with the ranking of information according to some criterion of interest. In most implementations, information prioritizing deals with the ranking of large sets of data points. The points ranked highest are given the most priority and are likely to be selected or pursued further. Data points ranked lowest in terms of priority are more likely to be ignored. This is the abstract concept of information prioritization.

3. REAL-WORLD PROBLEM

Imagine a web information system that faculty members of a university system use to manage information concerning their academic production and achievements. System administrators want to detect potential security

risks such as account breaches or users having unusual problems. To accomplish this task, administrators have the system's transaction log at their disposal. The transaction log collects two types of information within the system. It records user environment details such as the user's IP address and web browser, and it also logs all user interactions including logins and logoffs, as well as information added, edited and deleted. The web information system has been in use since 2004 and the transaction log has more than 150,000 entries. This may not be a particularly large collection of data to inspect, but the team of system administrators is relatively small. This amount of information represents a rather large and difficult workload for a team with so few people.

Systems administrators would benefit from an application that would prioritize the information in the transaction logs to their specific criteria.

4. SOFTWARE SOLUTION

An information ranking application was designed and developed to aid the system administrators in performing their system security assessment tasks. Let's examine the application, its design and development more closely. As it was mentioned before, prioritizing information deals with the ranking of information according to some criterion of interest. Data points ranked highest are given the most priority and pursued further, while data points ranked lowest are likely to be ignored. In this real-world scenario, what the information ranking application looks to accomplish is to evaluate user behavior patterns to detect anomalous user behavior and prioritize the log entries accordingly. From these details, we can clearly establish that rarity is the criterion of interest. In other words, the selection criterion, in this case, is the statistical frequency of the behavior or action for the user. The more unusual or rare the behavior, the higher it should rank. To accomplish this, a series of detectors were implemented in the application.

4.1 DETECTOR CONCEPT

The main functionality requirement of the application is the prioritization of transactions. For this purpose, the concept of detectors was used as a guide (Helman, et al, 1997). This work manages to recreate and apply the concepts and examples to help manage this real-world situation. A phenomenon detector is a mechanism to score individual transactions based on particular feature values of the transactions. These scores induce a prioritization of the transactions. Once transactions are scored, users can pursue transactions from those scoring highest to lowest. Helman and et al. first establish the simplest of detectors, known as the naïve detector. The naïve detector creates a model of data based on frequency counts of the transactions. After establishing the naïve detector, the authors refine it by introducing the concepts of equivalence classing and the independence surrogate. This detector equates interest with rarity. This should make it an excellent choice for the application being developed.

4.2 DETECTOR A: THE NAÏVE DETECTOR

The naïve detector is the simplest of the detectors. It utilizes frequency counts to score transactions as shown in (1).

$$r(x) = \frac{m_u(x)}{n_f(x)} \tag{1}$$

The detector computes the likelihood ratio *r(x)* of the transaction by dividing the probability associated with *x* under the uniform surrogate as described in (2), by the raw frequency count estimate as seen in (3).

$$n_f(x) = \frac{count(DB, x)}{|DB|} \tag{2}$$

$$m_U(x) = \frac{1}{|S|} \tag{3}$$

The raw frequency count (2) is computed by counting the number of transactions in the training subset *DB* similar to selected transaction *x* and dividing this by the total amount of transactions in the training subset *DB*. The

uniform surrogate (3) is computed by dividing 1 by the total amount of transactions in the transaction universe *S*.

In actuality, the naïve detector is satisfactory for some applications, but for our real-world scenario, it has a number of problems. First of all, when the transaction universe *S* is large compared to the training subset *DB*, the frequency counts in will be low and provide unrealistic estimates of true probabilities (Helman, et al, 1997).

Additionally, in the real-world case we are pursuing, as with many other applications, transactions tend to be largely unique in nature. This causes the transaction count to be 1 for most every transaction, in other words, singleton transactions.

Lastly, detectors based on raw transactions frequency have no legitimate basis by which to prioritize data points.

4.3 DETECTOR B: THE NAÏVE DETECTOR AND EQUIVALENCE CLASSING

To improve on the naïve detector's prioritization accuracy, the concept of equivalence classing is introduced. Equivalence classing aggregates individual transactions into transaction classes. Transactions that are part of the same transaction class are treated as indistinguishable entities. This aids in the elimination of possible singleton transactions. Frequency counts of transaction classes more accurately reflect class probabilities than do frequency counts of raw transactions represent raw transaction probabilities (Helman, et al, 1997).

Value clustering is the most appropriate application of equivalence classing for the development of my application. This transformation clusters together transactions with distinct feature values into classes. Transaction features with values from a continuous domain (such as time or floating point column values) can be clustered into established ranges or buckets. Schapire and et al. (1998) provide another viewpoint on establishing of classes.

Appling this concept to (1), we obtain (4).

$$pd_U((feature1, feature2, \dots)) = \frac{\theta_U(x)}{\theta_U((feature1, feature2, \dots))} \quad (4)$$

Let's review a simple example of how the elements of (4) work to compute scores for a couple of transactions.

Example: Detector B

Suppose we have a training subset DB that consists of 100 transactions. These transactions were clustered into classes by the values of the features *username* and *action*. The feature *username* has the value set {*UserAda,UserBob*} and the feature *action* has the value set {*add,edit*}. The following are the transaction counts for each of the four possible transaction classes in the training subset *DB*:

frequency(UserAda,add) = 6; frequency(UserAda,edit) = 88; frequency(UserBob,add) = 2; frequency(UserBob,edit) = 4.

Now, we will compute the detector values for the (*UserAda,add*) and the (*UserBob,add*) transaction classes. First, we compute θ_N as shown in (5) and (6).

$$\theta_N((UserAda, add)) = \frac{count(DB, (UserAda, add))}{|DB|} = \frac{6}{100} = 0.06 \quad (5)$$

$$\theta_N((UserBob, add)) = \frac{count(DB, (UserBob, add))}{|DB|} = \frac{2}{100} = 0.02 \quad (6)$$

Next, we compute the uniform surrogate $\theta_U(x)$ for this example (7). Being the uniform surrogate, it assigns a uniform probability to all transactions classes.

$$\theta_U(x) = \frac{1}{transactionclasses} = \frac{1}{4} = 0.25 \quad (7)$$

Now, we can compute the detector values pd_U for both of these transaction classes as shown in (8) and (9).

$$pd_U((UserAda, add)) = \frac{\theta_U(x)}{\theta_N((UserAda, add))} = \frac{0.25}{0.06} = 4.1667 \quad (8)$$

$$pd_U((UserBob, add)) = \frac{\theta_U(x)}{\theta_N((UserBob, add))} = \frac{0.25}{0.02} = 12.5 \quad (9)$$

By comparing the results of (8) and (9), it is clearly evident that (*UserBob,add*) is the most suspicious transaction class because the detector scored it much higher than (*UserAda,add*). This result can be easily verified against the training sample. In fact, (*UserBob,add*) is the transaction class with least frequency in the training sample.

4.4 DETECTOR C: THE NAÏVE DETECTOR, EQUIVALENCE CLASSING AND THE INDEPENDENCE SURROGATE

The final improvement for the naïve detector is the addition of the independence surrogate. The independence surrogate induces a level of independence among the classes in the calculation of the naïve detector. This makes the calculation increase in accuracy, since the detector is able to differentiate classes that show unusual behavior for the class, instead of unusual behavior within the whole transaction set. To achieve this, we must substitute the uniform surrogate $\theta_U(x)$ for the independence surrogate $\theta_I(x)$. Appling this concept to (4), we obtain (10).

$$pd_I((feature1, feature2, \dots)) = \frac{\theta_I((feature1, feature2, \dots))}{\theta_N((feature1, feature2, \dots))} \quad (10)$$

Let's go over an example of how the elements of (10) work to compute scores for a couple of transactions.

Example: Detector C

Supposing the same setup as in the previous example, we will substitute the uniform surrogate $\theta_U(x)$ for the independence surrogate $\theta_I(x)$. First, we compute the independence surrogate $\theta_I(x)$ for the (*UserAda,add*) and the (*UserBob,add*) transaction classes as shown in (11) and (12).

$$\theta_I((UserAda, add)) = \frac{frequency(UserAda)}{|DB|} * \frac{frequency(add)}{|DB|}$$
$$\theta_I((UserAda, add)) = \frac{94}{100} * \frac{8}{100} = 0.94 * 0.08 = 0.0752 \quad (11)$$

$$\theta_I((UserBob, add)) = \frac{frequency(UserBob)}{|DB|} * \frac{frequency(add)}{|DB|}$$
$$\theta_I((UserBob, add)) = \frac{6}{100} * \frac{8}{100} = 0.06 * 0.08 = 0.0048 \quad (12)$$

Now, we compute the detector values pd_I for the transaction classes just as before, making substituting the uniform surrogate $\theta_U(x)$ for the independence surrogate $\theta_I(x)$ as shown in (13) and (14).

$$pd_I((UserAda, add)) = \frac{\theta_I((UserAda, add))}{\theta_N((UserAda, add))} = \frac{0.0752}{0.06} = 1.2533 \quad (13)$$

$$pd_I((UserBob, add)) = \frac{\theta_I((UserBob, add))}{\theta_N((UserBob, add))} = \frac{0.0048}{0.02} = 0.24 \quad (14)$$

Now, by comparing the results of (13) and (14), we can conclude that (*UserAda,add*) is the most suspicious transaction class, because it received a higher score than (*UserBob,add*). It is easy to verify this against the training sample, since *add* is a relatively unusual *action*, for the user with *username UserAda*.

The independence surrogate adds another layer of improvement to more accurately score unusual behavior patterns based on independent transaction classes.

4.5 APPLICATION IMPLEMENTATION DETAILS

The application developed to prioritize the transactions of the real-world web system implements all three

detectors as shown by Figure 1. System administrators can choose to prioritize the list of transactions by any of the three detector scores.

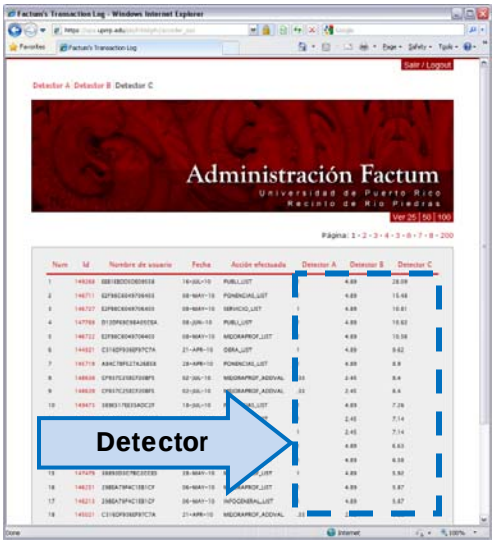

Figure 1: Actual application screen displaying transactions with their corresponding detector scores

Detector A consists of the naïve detector. Detector B adds the improvement of equivalence classing to the naïve detector. Detector C further incorporates the concept of the independence surrogate to Detector B. This was done to study and compare the results of each detector against the others and draw conclusions regarding the conceptual improvements to the naïve detector.

The programming of each detector was encapsulated into a function. Each of the functions receives the id of the transaction and returns the computed value of the detector for that transaction.

Figure 2 shows additional functionality that was developed in this cycle.

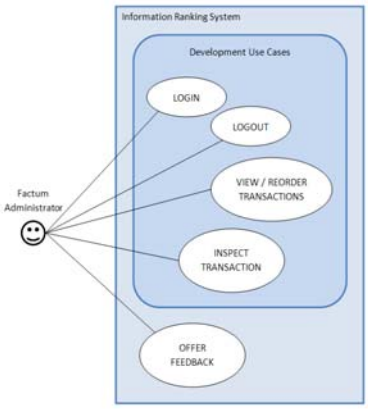

Figure 2: Application use case diagram

5. EXPERIMENTAL RESULTS

In this execution run, the application evaluated a subset of 5,000 transactions from the log of the real-world web system. Each transaction was evaluated by each of the three detectors and assigned a separate score by each detector. The feature set selected for this execution run was {*username, procedure_performed*}.

To generate the frequency distributions for each detector, the detector scores were divided into six quantiles or bins. Then, the frequencies were computed for each of the bins. Figures 3, 4 and 5 summarize the frequency

distributions for each of the detectors in the form of histograms.

Figure 3 details the results for Detector A. From it, we can gather that a significantly large amount of transactions gathered around the three highest bins. The lowest three bins have a very small amount of transactions.

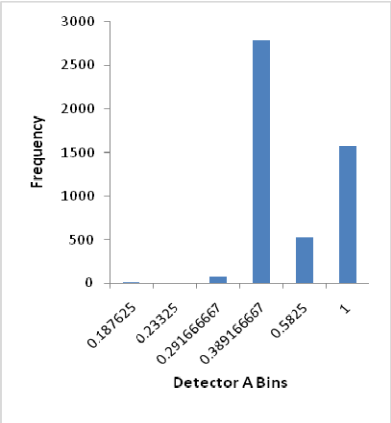

Figure 3: Histogram for Detector A

From the histogram for Detector B in Figure 4, we can observe that the largest single group of transactions clusters around the highest bin. The rest of the bins, however, have a larger and similar amount of transactions than with Detector A.

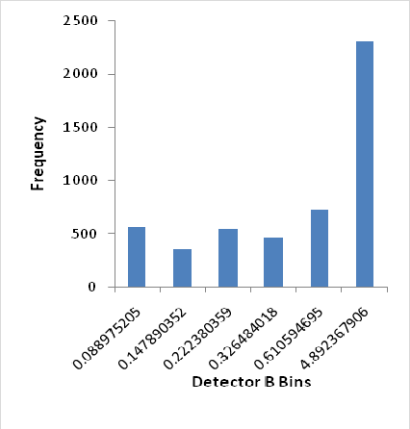

Figure 4: Histogram for Detector B

From the histogram of Detector C in Figure 5, we can gather that the largest amount of transactions cluster around the three lowest bins. The transactions in these bins are the most likely to be ignored since their priorities are relatively low. The amount of transactions steadily decrease from the second bin as the bins cluster transactions with higher scores.

Table 1 presents a comparative view of some basic descriptive statistics for each detector.

The range for Detector A is only 0.875. This represents a very small range of values in which to assign priorities. In other words, the scores are spread within a small range to effectively and clearly differentiate each other. Additionally, at only six, the amount of different scores generated for transactions is relatively small, which further compounds the problem. The scores generated by Detector A established a mean of 0.559. Looking at Figure 3, this would put the mean of the distribution somewhere within the fifth bin. Having most of the transactions cluster around the highest bins (the bins that hold the highest scored transactions) means almost all transactions are scored as suspicious. Clearly, this does not accomplish what we need in terms of detecting unusual behaviors.

Table 1: Descriptive statistics

Statistics	Detector A	Detector B	Detector C
Mean	0.559	1	0.5644
Standard deviation	0.3040	1.2548	0.9178
Minimum	0.125	0.0480	0.0056
Maximum	1	4.8924	28.0896
Range	0.875	4.8444	28.084
Scores	6	43	871

The range of scores from Detector B is 4.8444. This shows that Detector B has a range more than 5.5 times larger than the range of Detector A. Additionally; Detector B was able to generate 43 different scores for the transactions in the set. This is approximately 7.1 times more scores generated than Detector A. Both of these factors should allow for a better spread of the scores across the potential range. This is confirmed by the histogram in Figure 4. However, having an approximate mean of 1 establishes the largest cluster of transactions around the sixth bin. Although a slight improvement over Detector A, this detector still represents a problem in terms of scoring the majority of transactions as suspicious. The range of values generated by Detector C was of 28.084. This makes it approximately 5.8 times larger than the range that Detector B was able to generate. Detector C also generated 871 distinct scores within that range. This fact means that Detector C was able to generate approximately 20.3 times the amount of scores than Detector B. Also, another important detail is that the mean for the frequency distribution generated by Detector C is 0.5644. This puts it around the third and fourth bins, clustering the largest amount of scores towards the lower bins, as evidenced in Figure 5. This much larger range works to better spread the results. These two facts prove to be a dramatic improvement of Detector C over Detector B.

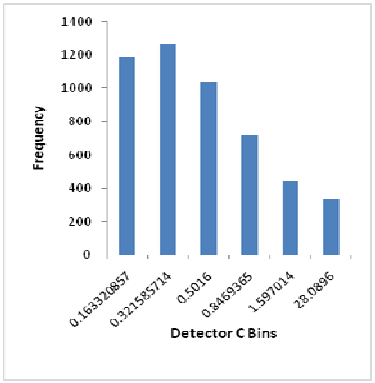

Figure 5: Histogram for Detector C

It is quite clear the way in which each detector progressively spreads more scores over to the lower-scored bins. Also of note is the dramatic reduction of frequencies that Detector C induced in the highest-scored bins. We can clearly see how much influence each detector had on the frequencies by viewing the frequency percentages per bin in Table 2.

If we look at the percent of transactions in the highest three bins (bins 4, 5 and 6), we can see how much improvement each detector has over the previous. Detector B was able to reduce the transactions in the highest bins by 27.8% over Detector A. Furthermore, Detector C reduced this amount even further; reducing the transactions by 40.1% over Detector B and an impressive 67.9% over Detector A. Detector C also shows a potential trend to progressively reduce the frequency of transactions as they score higher. This is a highly desirable trait in a detector for the purpose of detecting anomalies.

Table 2: Frequency percentages

Bins	Detector A	Detector B	Detector C
1	0.3%	11.5%	23.7%
2	0.2%	7.2%	25.4%
3	1.5%	11.1%	20.8%
4	55.8%	9.4%	14.4%
5	10.6%	14.7%	8.9%
6	31.5%	46.2%	6.8%

6. CONCLUSION

We think about the concept of anomalous behavior as behavior that strays far from that which is considered normal or expected. When this concept is applied to statistical data, anomalous data is considered to be that which strays far from the average or mean. Intuitively, it is known that the majority of transactions in an information system should be mundane and of little interest for our purposes. The outlying data is the data of most importance for this development experiment.

In essence, the development project looked to create a means to positively identify transactions considered to be unusual in terms of relative frequency of occurrence. The mechanism used to accomplish this task was the development of detector functions that would assign a relative score to each transaction.

The work of detector development was based on work published in [6]. In their work, the Helman and et al. presented the simplest of detectors, the naïve detector, to which they made a couple of improvements to serve their purpose. The first improvement to the naïve detector was the addition of equivalence classing. The second improvement was the addition of the independence surrogate. Let us evaluate the effects that each of these improvements have on the naïve detector and how they contribute to our purpose.

Detector A, the naïve detector, managed to score 98% of the transactions in the higher half of the range of scores. This result actually goes against the concept of anomaly detection. From this, we can conclude that Detector A is of very little practical use in our environment.

Detector B adds equivalence classing to Detector A. The addition of this equivalence classing to the naïve detector helped in the mitigation of its shortcomings. Detector B scored a higher amount of transactions in the lower half of the range. However, it still scored a significantly large amount of transactions (approximately 70%) in the upper half of the score range. Clearly, further improvement is still required for Detector B to be a useful mechanism for our purpose.

Enter Detector C, which adds to the naïve detector the independence surrogate in addition to equivalence classing. The combination of improvements resulted in a great reduction in the number of transactions scored within the higher half of the range. Only about 30% of transactions scored in the top half of the range. Another characteristic evident in the analysis is that the frequency of the scores generated by Detector C show a progressive decrease as transactions achieve higher scores. The mean for the frequency distribution of Detector C is towards the lower scoring bins, making the higher scoring bins the most outlying and lower frequency bins. Both of these characteristics represent a behavior ideally suited for identifying abnormal behavior patterns within a set of transactions. Detector C was both, more selective when scoring transactions than the other detectors, as well as offering much finer granularity in the resulting scores.

Of the three detectors developed for this project, Detector C far outperformed the others. It showed desired behavior in more aspects than the rest. It displayed behavior that is considered highly ideal for the purpose of prioritizing transactions according to rarity. However, this doesn't mean that Detector C stands for no further improvement.

7. FUTURE WORK

This is the work that can be done in future development cycles to improve on detector accuracy. Further development and testing would be required to implement and evaluate these measures:

Detector improvement: The scores resulting from each detector could be combined into a new, single score. This new score would constitute a whole new detector. Each detector would be assigned a weight factor in the computation. Based on the current results achieved in this analysis, Detector A would receive the smallest weight, followed by a slightly larger weight for Detector B and finally, Detector C would have the highest weight in the calculation. The purpose would be to improve on the accuracy of Detector C. Perhaps the results from Detector C could be evaluated by other well-known text processing algorithms such as Rocchio, InRoute or INQUERY (Yang, et al., 2000), (Schapire, et al., 1998) and (Callan, 1996). This could further refine the results.

User feedback: The incorporation of user feedback into the calculation of detectors could provide significant improvements according to Belkin & et al. Users could offer positive or negative feedback depending on their judgment. Transactions with high scores that prove to be genuine anomalies would receive positive feedback. Transactions with high scores that prove not to be anomalous would receive negative feedback. The feedback could be used to establish more complex detectors and help refine their accuracy. The utilization of user feedback to establish profiles is an intriguing concept. This would be used to filter the prioritized results from detectors using one or several of the techniques and algorithms in (Schapire, et al., 1998) and (Callan, 1996). User feedback may also prove useful in extracting better patterns from the data as shown in (Jo, et al., 2007). If we were to take this further, we could also apply the concepts in (Jo, et al., 2007) to attempt to predict potential behavior patterns. This could be the basis of a proactive anomaly detection system.

REFERENCES

Belkin, N. J., et al., "Information Filtering and Information Retrieval: Two Sides of the Same Coin?", *Communications of the ACM*, 35(12), pp.29-39.

Brandt, R., (2004). Starting Up: How Google Got its Groove, *Stanford Magazine from stanfordalumni.org*, November / December 2004. http://www.stanfordalumni.org/news/magazine/2004/novdec/features/startingup.html. (Accessed: May 27, 2010.)

Callan, J., (1996). "Document Filtering with Inference Networks", *Proceedings of the 19th Annual international ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 262-269.

Helman, P., et al., (1997). "A Statistically Based System for Prioritizing Information Exploration Under Uncertainty", *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 27(4), pp. 449-466.

Jo, H., et al., (2007). "Prioritized Traffic Information Delivery Based on Historical Data Analysis", *Proceedings of the 2007 IEEE Intelligent Transportation Systems Conference*, pp. 568-573.

Li, W., (2002). The First Search Engine, Archie, http://www.isrl.illinois.edu/~chip/projects/timeline/1990archie.htm. 09-21-02. (Accessed: May 27, 2010.)

Schapire, R. E., et al., (1998). "Boosting and Rocchio Applied to Text Filtering", *Proceedings of the 21st Annual international ACM SIGIR Conference on Research and Development in information Retrieval*, pp. 215-223.

Toffler, A., (1965). "The Future as a Way of Life", *Horizon magazine*, 7(3).

Yang, Y., et al., (2000). "Improving Text Categorization Methods for Event Tracking", *Proceedings of the 23rd Annual international ACM SIGIR Conference on Research and Development in information Retrieval*, pp. 65-72.

Authorization and Disclaimer

Authors authorize LACCEI to publish the paper in the conference proceedings. Neither LACCEI nor the editors are responsible either for the content or for the implications of what is expressed in the paper.