Paper 1640-2015

Understanding Characteristics of Insider Threats by Using Feature Extraction

Ila Nitin Gokarn, Singapore Management University; Dr. Clifton Phua, SAS Institute Inc.

ABSTRACT

This paper explores feature extraction from unstructured text variables using Term Frequency - Inverse Document Frequency (TF-IDF) weighting algorithms coded in Base SAS®. Datasets with unstructured text variables can often hold a lot of potential to enable better predictive analysis and document clustering. Each of these unstructured text variables can be utilized as inputs to build an enriched dataset-specific inverted index and the most significant terms from this index can be used as single word queries to weight the importance of the term to each document from the corpus. This paper also explores the usage of hash objects to build the inverted indices from the unstructured text variables. We find that hash objects provide a considerable increase in algorithm efficiency and our experiments show that a novel weighting algorithm proposed by Paik (2013) best enables meaningful feature extraction. Our TF-IDF implementations are tested against a publicly available data breach dataset to understand patterns specific to insider threats to an organization.

INTRODUCTION

The Chronology of Data Breaches maintained by Privacy Rights Clearinghouse contains a comprehensive list of data breaches in the United States since 2005. The Privacy Rights data provides information on the types of data breach, types of host organization, locations and number of records stolen and this can lead to many interesting results from exploratory data analysis. However, despite the data being quite sanitized and well maintained, it does not provide many useful variables to aid in predictive analysis to classify the type of data breach – whether it was an insider threat, a hacker, a stolen portable device, a stationary device like a server, unintentional disclosure by an employee, payment card loss, or loss of physical data. Another drawback of the data is that a lot of crucial information is hidden in the description field in an unstructured format and no single regular expression can successfully extract all relevant information. Similarly, numerous datasets are supported by unstructured text fields which hold important information that can be mined. For example, we have the DARPA Anomaly Detection at Multiple Scales (ADAMS) which detects insider threats based on network activity, collections of weblogs, emails which have significant text variables. For this purpose, we explore the usage of domain-agnostic information retrieval methods to extract significant terms from the text variables. This would create term features which would then be assigned a term frequency weight using a Term Frequency-Inverse Document Frequency (TF-IDF) algorithm. This weight would indicate the importance of the feature to that record and will aid in better prediction and clustering of the records in the dataset.

This paper walks through a three-step process to create accurate decision trees and clusters – feature extraction using TF-IDF in a SAS environment; feature selection, and analysis using SAS® Enterprise Miner™. Term weighting already exists in the “Text Filter” node in Enterprise Miner (SAS, Getting Started with SAS Text Miner 12.1, 2012) but here, we expand the scope by providing more TF-IDF weighting options. This paper primarily contributes Base SAS implementations of known TF-IDF algorithms using hash objects for efficient term weighting as an addition to those existing in Enterprise Miner (Warner-Freeman, 2011). The results show that the best weighting method is Paik’s Multi-Aspect Term Frequency (MATF) (Paik, 2013), followed by Okapi BM25, a probabilistic model.

The subsequent sections detail a literature review in the field of information retrieval, methods used in our experiments, its results, future works and a conclusion to our experiments.

LITERATURE REVIEW

Term weighting is a central concept in the field of Information Retrieval (IR) and contributes greatly to the effectiveness and performance of any IR system (Salton & Buckley, 1988). Each term (which could be a word or a phrase) does not hold the same importance in the document. For example, articles,

conjunctions and other grammatical adages to a sentence appear extremely frequently in any document but usually do not hold any particular importance or significance in relation to that particular document. Calculating the importance of the term to the document involves three main characteristics: term frequency in the document (TF), the inverse document frequency factor (IDF) and the document length normalization factor (Manning & Raghavan, 2008). As a result of these three main factors, IR methods can be essentially classified into three categories:

1. **Vector Space Models** where both the document and the query on the document are considered as a vector of terms
2. **Probabilistic Models** where the probabilities of the existence of the query terms in the document are calculated
3. **Inference based Models** which instantiate a document with a weight and infer the query weights in comparison

All models have different strategies in tuning the precision and recall of the relevant terms. “In principle, a system is preferred that produces both high recall by retrieving everything that is relevant, and also high precision by rejecting all terms that are extraneous.” (Salton & Buckley, 1988). Term frequency (TF) aids in retrieving all those terms which occur most frequently in the document and therefore acts as a recall enhancing factor.

However, the term importance cannot be measured by counting the TF in that document alone, it also requires some comparison to the TF across all the documents in the collection in order to ensure relevance and search precision of the term over the entire collection of documents. For this, the inverse document frequency (IDF) which deals with the document frequency of terms is required. IDF ensures that high frequency terms which occur throughout all documents are not favored in an effort to increase precision for extremely specific words.

Therefore, a reliable measure of term importance is obtained using a product of the term frequency and inverse document frequency commonly known as TF-IDF which ensures that the best words or terms are those with high term frequencies and low document frequencies (Salton & Buckley, 1988). The TF-IDF weight assigned to a term aids in tuning both recall and precision, and this standard TF-IDF is the 1st model in our experiments.

However, in IR, TF-IDF shows different behaviors over documents of varying lengths, as short queries tend to prefer short documents and long queries prefer long documents. In the past, a lot of research has been contributed to finding the most effective weighting scheme to best normalize the weight of a term, particularly by Singhal et al (1996). The document length has a large effect in the term weight in two ways:

1) Long documents use the same terms repeatedly leading to large term frequencies for certain terms.
2) Longer documents have a larger number of terms. This increases the possible number of matches between the query and document during retrieval.

Document normalization factors tend to penalize larger or longer documents in order to equalize the treatment of both long and short documents over different query lengths. The most successful of all normalization functions is the Pivoted Document Length Normalization suggested by Singhal et al (1996) and is the 2nd model to be evaluated in our experiment. This algorithm involves augmented normalization which normalizes the TF against the maximum term frequency in the document and uses a parameter s which acts as the pivoting factor between the probabilities of retrieval versus relevance.

The 3rd algorithm is a novel TF-IDF weighting proposed by Paik (2013) where he observes that a variation in the parameter s from Singhal et al (1996)’s formula for Pivoted Document Length Normalization leads to an imbalance while weighting terms against a mix of long and short documents. When the parameter s is set to a larger value, the algorithm prefers longer documents and when s is set to a smaller value the algorithm prefers short documents. Paik (2013) suggests that we take into consideration both short and long documents in order to not overcompensate shorter documents. The paper suggests Multi-Aspect Term Frequency (MATF), a product of an intra-document TF as well as a

document length normalized TF. The proposed work asserts that the intra-document factor prefers shorter documents, and the length regularized factor favors longer documents, and a product of both with an appropriate weighting does not support one at the cost of the other. This leads to the creation of a more accurate model (Paik, 2013). He puts forth four hypotheses:

1) **TF Hypothesis:** The weight of a term should increase with its TF but since it is unlikely to see a linear relationship, we dampen the weight using a logarithmic transformation

2) **Advanced TF Hypothesis:** The rate of change of weight of a term should decrease with a larger TF

3) **Document Length Hypothesis:** It is necessary to regulate the TF with respect to the document length to avoid giving longer documents any undue preference

4) **Term Discrimination Hypothesis:** If a query contains more than one word, the document retrieved should be based on the rare term present in the query itself, indicating a need for weighting in the query also

For single worded queries, as is with our experiments, probabilistic models do exceedingly well and show much better accuracy than fully weighted vector models (Salton & Buckley, 1988). Therefore, we also look at a classic weighted probabilistic model and Okapi BM25, a state-of-the-art probabilistic model making them our 4th and 5th model of interest respectively. An interesting observation is that Paik (2013) compared MATF to probabilistic models and noted better performance of MATF over Okapi BM25 so we are also interested in finding out if it held significance even for single word queries.

METHODS

Here, we discuss the methods in which the different weighting algorithms discussed in the Literature Review section are implemented in the SAS environment.

Feature extraction can be carried out by individually evaluating each feature or term (univariate selection) and seeing their independent relevance to the dataset, which leads to ranked feature selection (Guyon & Elisseeff, 2007). This approach is generally considered efficient and effective, especially when dealing with large numbers of extracted features as is with term extraction from a text variable. However, this leads to two potential drawbacks due to the assumed independence between the extracted features:

1) Features that are not relevant individually may become relevant in conjunction with other features

2) Features which are individually relevant may actually be redundant

To overcome these drawbacks, we create the inverted index of all the unique terms in the entire dataset in Algorithm 1 ExtractInvertedIndex detailed below and construct all the features using TF-IDF weights in Algorithm 2 CalculateTF-IDF. We then leave the feature selection and evaluation to pre-existing algorithms in Enterprise Miner (SAS, Getting Started with Enterprise Miner 12.1, 2012). The constructed features which are now regarded as inputs to the model are first clustered, then selected and evaluated based on information gain while building the decision tree in Enterprise Miner. The feature cluster which gives the best split in the tree is selected and built on further. Clustering of the features mimics multivariate feature selection which makes simple assumptions of co-dependence of features or terms based on their co-occurrence in the document corpus. This multivariate selection overcomes the two drawbacks met when employing univariate selection.

In Algorithm 1 ExtractInvertedIndex, we incorporate SAS hash objects to avoid complex PROC SQL steps or the creation of numerous tables to compute term, document and collection frequencies (Eberhardt, 2010). In the first step, the documents are processed in sequence. At the beginning of a document evaluation iteration, a document specific hash *h1* is constructed and each term (as key) is stored along with the term frequency (as value) in that document. Once the entire document is stored into the hash *h1*, we iterate through *h1* and check if the term exists in the global hash *h* which stores the document frequency or the number of documents the term appears in over the entire corpus of documents. If the term is there, we increment the document frequency by one, if not, we add the word to the hash *h*. The global hash *h* is then stored in a table *OUT* using the hash object in-built function called

output. The table *OUT* is then stripped of all stop words which are usually grammatical adages, articles, conjunctions, prepositions and such. The inverted index is then created by swapping the table *OUT* into another hash object which takes the document frequency as the key and the term as the value. This index is then written out to another table called *INDEX* with the keys (document frequencies) sorted in descending order (SAS, Using the Hash Object, 2014). We also add a few words specific to Privacy Rights dataset to the stop list in an effort to remove all non-specific words or descriptions from consideration such as the words “specific”, “motive”, and “unknown” along with special non-ASCII characters in the dataset.

Algorithm 1 ExtractInvertedIndex

1. **Require:** A dataset *s* with *n* variables $V_0,..,V_n$ of which V_x is an unstructured text variable, for example “Description”
2. **Ensure:** Returns a dataset *s* with *m* observations, one for each term extracted from Description and two variables, the term and document count
3. *s* ← dataset of *n* variables of which V_x is an unstructured text field
4. *h* ← global hash object with key *k* as term and data value *d* as document count
5. *h1* ← local hash object with key *k1* as term and data value *d1* as term count in that document
6. initial *j* ← 1
7. *word&j* ← extract j^{th} word from document
8. **do while** *word&j* not equal to blank
9. *k1* ← *word&j*
10. **if** *k1* exists in hash *h1* **then**
11. *d1* ← *d1*+1
12. **end** of **if**
13. **else**
14. add *k1* to hash *h1* and set *d1* to 1
15. **end** of **else**
16. *j* ← *j*+1
17. *word&j* ← extract word j^{th} from document
18. **end** of **do while**
19. *hiter* ← iterator for document hash *h1*
20. **do while** *hiter* has next in hash *h1*
21. *k* ← next term in hash *h1*
22. **if** *k* exists in hash h **then**
23. *d* ← *d*+1
24. **end** of **if**
25. **else**
26. add *k* to hash *h* and set *d* to 1
27. **end** of **else**
28. **end** of **do while**
29. table *OUT* to hold output of hash object *h*
30. **return** *OUT*
31. **remove** stopwords from table *OUT*
32. *h2* ← hash object with key *k2* as document count and data value *d2* as term (inverted index)
33. read contents of *OUT* into *h2* sorted in descending order of document count
34. table *INDEX* to hold output of hash *h2*
35. **return** *INDEX*

Table 1: Algorithm 1 ExtractInvertedIndex

The table *INDEX* which is obtained from the above piece of code contains two columns, document frequencies in descending order and their corresponding terms. As shown in Algorithm 2 CalculateTF-IDF, once this index is created we select the top-30 terms by virtue of their document frequencies and add them as features by turn to the original dataset assuming that 30 terms would allow for the creation of an approximately sufficient feature subset. This number of features selected (e.g. 30 selected in this paper) can be varied for different datasets in order to create approximately sufficient feature subsets. Then, for each feature added, we calculate the TF-IDF of the feature with respect to each document. Hence the exact calculation of TF-IDF differs over the weighting methods but the overall structure of the code remains the same. The equation for each weighting function is detailed in in Exhibit 1: TF-IDF Model Equations. The MATF algorithm only has a slightly different *INDEX* table that is generated – it contains an additional column called collection frequency for each of the terms in addition to document frequency, in accordance with the MATF model equation.

Algorithm 2 CalculateTF-IDF

Require: A dataset *s* with *n* variables $V_0,..,V_n$ of which V_x is an unstructured text variable, for example "Description"; and dataset *INDEX* from **Algorithm ExtractInvertedIndex** with 2 variables of which every record under variable V_2 is a feature
Ensure: A dataset with *n*+30 variables of which $V_{n+1},..V_{n+30}$ are features

```
1. create table TEMP with top-30 words from INDEX
2. &&n& ← number of records in TEMP as macro variable
3. &&k&i ← i^th feature from TEMP as macro variable
4. &&d&i ← i^th data value from TEMP as macro variable
5. for all i =1 to &&n& with incremental counter of 1
6.   initial j = 1
7.   word&j ← extract j^th word from document
8.   initial this_word_count = 0
9.      initial total_word_count = 0
10.     do while word&j is not blank
11.        if word equal to feature &&k&i. then this_word_count ←
   this_word_count+1
12.        total_word_count ← total_word_count+1
13.        increase counter j
14.        word&j ← extract word j^th from document
15.     end of do while
16.     if this_word_count>0 then
17.        tf ← calculate the term frequency
18.        idf ← calculate the inverse document frequency
19.        TF-IDF ← tf*idf
20.        &&k&i ← TF-IDF
21.     end of if
22.     else
23.        &&k&i ← 0
24.     end of else
25.   end of i loop
26.   drop all unnecessary columns
27.   return s
```

Table 2: Algorithm 2 CalculateTF-IDF

As observed by Paik (2013), models like Singhal et al (1996)'s Pivoted Document Length Normalization model are very sensitive to change in the parameters used in the model equations that aid in normalization. Different tweaks in parameters yield different results. The baselines of the parameters used are elaborated below:

1) **Classic TF-IDF with no normalization:** This model has no parameter but deviates from the classic model where cosine normalization is used. This is because the document lengths in the corpus in this particular dataset are fairly homogenous in length and therefore do not require much normalization

2) **Pivoted length normalization model:** This model is a high-performing model in the vector space collection of models, the parameter *s* has been set to the default 0.05 (Singhal et al, 1996)

3) **Best weighted probabilistic model:** The augmentation factor has been set to the default 0.5 although some research work has been done on the using a 0.4,0.6 pairing in normalizing the term frequency (Manning & Raghavan, 2008)

4) **Okapi BM25:** Whissell and Clarke found that out of the three parameters used in Okapi BM25, *k1* should be higher than the default 1.2 as it leads to better clustering of the documents. Following this prerogative, we assign *k1* the value 1.8 and *b* the default 0.75 (Whissell & Clarke, 2011)

5) **Multi-Aspect TF-IDF (MATF):** This model has a weighting factor *w* to balance out the weights of short and long queries on documents. Since we have only one word queries, we set *w* to 1 (Paik, 2013)

The final dataset *s* is imported into Enterprise Miner for further analysis. The datasets are initially passed through the Variable Cluster node which reduces the dimensionality of the data, and also allows for some assumptions on co-dependence of the features based on co-occurrence in the dataset as detailed earlier.

The data records are then partitioned into 60% towards training, 20% to training and 20% to testing. Next, the data is passed into the Decision Tree node where the selection criteria is set to Testing: ROC Index. The Receiver Operating Characteristic (ROC) curve is a visual method to compare classification models (Han, 2012). It gives the trade-off between true-positives to false-positives in the classification problem which means that the ROC curve depicts the ratio of the times the model has accurately classified a positive case versus the time the model has mistakenly classified a negative case as positive in a binary target scenario. The ROC Index or Area under the ROC curve gives the accuracy of the model using the testing data (rather than the training data which tends to be over-fit).

In the Decision Tree node in Enterprise Miner, all values are left to default except the handling of missing values where the documents with missing values are assigned to the most correlated branch. Additionally, all inputs or features selected by the tree during evaluation is allowed to be used only once, again to prevent over-fitting of the tree. A breakdown of the inputs are elaborated in Exhibit 2: Dataset Variables. The target variable is set to the breach type (variable “Type” in dataset).

All the models are evaluated together and drawn into a single Model Comparison node to discern the most successful weighting algorithm based on the ROC Index. The entire model layout is depicted in Exhibit 3: Enterprise Miner Model Layout.

RESULTS

Here we present the results of the Model Comparison node in Enterprise Miner as shown in Exhibit 3: Enterprise Miner Model Layout. Among all the results shown in Table 3: Results of TF-IDF Evaluations, we make the following key observations:

1) The weighted probabilistic model does indeed perform much better than the fully weighted vector space models despite having pivoted document length normalization. This is attributed to the fact that we are using single word queries as opposed to complex multi-worded queries which themselves would also require weighting in the query (Salton & Buckley, 1988).

2) Pivoted normalization model and the probabilistic model do not have very different results and so this does not attribute to the notion that probabilistic models are better for single word queries as compared to classic weighted models

3) Raw term frequencies are usually not too indicative of the term importance and the more dampening the TF undergoes, the better the weighting model performs. This can be seen in the following table as well, the more complex the term frequency component gets, the better the model performs

Model	Training: ROC Index	Testing: ROC Index	Validation: ROC Index
No TF-IDF	0.575	0.434	0.598
Classic TF-IDF	0.651	0.661	0.516
Pivoted length normalization TF-IDF	0.686	0.583	0.571
Classic probabilistic	0.656	0.681	0.543
Okapi BM25	0.729	0.605	0.593
Multi-Aspect TF-IDF (MATF)	0.731	0.718	0.607

Table 3: Results of TF-IDF Evaluations

The merits of feature extraction from an unstructured text field to support a dataset are evident upon observing the nature of the decision tree. Based on the values obtained in the results table, we focus our attention on the MATF model which seems to have the best performance among all weighting algorithms.

There are three branches emerging from the dataset “TFIDF_Modified” created by the MATF model:

1) Original variables and features in the dataset are examined using StatExplore node
2) A decision tree is built after clustering of the features
3) The clusters in the data are explored using Segment Profiler node

In the StatExplore branch, we get a comprehensive view of the variables in the dataset and a breakdown of the target types against each variable. Table 4: Categorical Variables against Insider Threats gives us a view of how the variables interact with the target of Insider Threats. Out of all cities, New York contributes to 6.48% of all insider threats followed by 2.47% of the threats from Baltimore. The most susceptible are Medical Institutions and Government Departments in California and Florida on a state level and New York and Baltimore on a city level.

Variable	Mode	Mode Percentage	Mode 2	Mode Percentage 2
City	New York	6.48	Baltimore	2.47
State	California	12.04	Florida	10.19
Type of Organization	Medical Institutions	35.80	Government Departments	18.83

Table 4: Categorical Variables against Insider Threats

In the decision tree branch, we first cluster the features in this dataset are divided into 11 major clusters as seen from the results of the Variable Clustering node shown in Exhibit 4: Variable Clustering Plot. Each upon closer observation indicate a specific component of a person’s identity, one variable cluster concentrates on names and social security numbers, another cluster on medical records, another cluster on credit cards and so on. These clusters are used as inputs to the decision tree shown in Exhibit 5: Decision Tree generated from features built by Multi Aspect Weighting. We see that the most prominent branch in the tree portrays that “Laptops”, “Stolen”, by/from “Employees”, “Contained” components of personal identity. Non-Medical organizations saw more of a theft of “Social”, “Security”, “Numbers” whereas Medical Organizations saw more of an Insider Threat to records with a focus on “Birth”, “Dates”,

“Names”, and “Addresses”. This insight is not discernable from a tree built without this feature extraction which only yields a tree with splits based on Entity or the kind of organization, city, state and number of records breached.

In the data clustering branch, we see 6 predominant data clusters as shown in Exhibit 6: Cluster Segments generated from features built by Multi-Aspect Weighting. The first significant cluster contains those records which pertained to the theft of Social Security Numbers, Birth Dates and Addresses of people but the cluster is characterized by the State in which the theft occurred and the total records stolen. The second cluster pertains to identity theft with the differentiating variable from the first cluster as “Employees” which could suggest that the breach was either carried out by employees or it affected only employees. The third cluster pertains to identities exposed from laptops, the fourth cluster on credit card and address information either accessed or exposed, the fifth cluster on medical records being breached. The final sixth cluster is on people being affected by a breach in medical institutions, which is by far the most sensitive and frequent kind of breach particularly perpetuated by insiders to the institution. These clusters give a rather clear picture of the characteristics of the main types of clusters in the data. In comparison, the original dataset without extracted features produces clusters based only on the Entity or the kind of organization which suffered the breach which are not too informative, as seen in Exhibit 7: Cluster Segments generated from original dataset. These insights allow organizations to better understand the characteristics of insider threats and data breaches.

FUTURE WORK

The term weighting algorithms implemented in Base SAS can be improvised and further strengthened using a variety of mechanisms which aid in better extraction. The building of the inverted index as in Algorithm “ExtractInvertedIndex” can be done based on a taxonomy as is in the Text Parsing node in Enterprise Miner in order to further filter out those words which are truly significant to the documents. Similarly, Algorithm “CalculateTF-IDF” can have a fourth part in addition to term weighting, document weighting and length normalization which could be a relevance feedback mechanism that will guide the algorithm during extraction. Such a feedback mechanism is suggested in variations of Okapi BM25 (Manning & Raghavan, 2008). Finally after all the features have been constructed, other forms of dimensionality reduction can be used instead of Variable Clustering, like Single Value Decomposition (SVD) in order to build text topics which would then carry a collective weight of all the terms in that topic. A novel way to add on to the existing term weighting algorithms would be by integrating a Google Search with every record or event. Cimiano and Staab (2004) proposed “Searching by Googling”, a novel way to add on to individual knowledge by drawing on the community using Google API in order to overcome knowledge acquisition bottlenecks and to strengthen the meta-data of any dataset. They suggested PANKOW (Pattern based Annotation through Knowledge on the Web) which would build lexical patterns indicating a semantic or ontological relation and then would count their occurrences using Google API thereby giving another measure of weight to the record itself (Cimiano & Staab, 2004). The dataset evaluated in this paper has more of high level information and therefore exact details for each record cannot be ascertained. However, the same term weighting algorithms can be run on network or access logs obtained by an organization to help in feature extraction, from which insider threats can be detected when comparing the extracted patterns to an existing black list of known scenarios.

CONCLUSION

In this paper, we propose SAS TF-IDF implementations using hash objects which aid in efficient feature extraction from unstructured text variables. These term weighting implementations are then evaluated in SAS Enterprise Miner by building decision trees comprising of the selected features. The ROC index (AUC) obtained from the testing dataset is used to choose the preferred model of feature weighting. Our experiments show the Multi-Aspect Term Frequency (MATF) proposed by Paik (2013) to be the best performing term weighting function and the decision tree built upon these features with MATF weighting proves to be insightful and gives a clear picture of the nature of data breaches conducted by an organization’s own employees.

APPENDICES

EXHIBIT 1: TF-IDF MODEL EQUATIONS

Model	Equation
Classic TF-IDF	$\sum_{t \in q \cap D} TF.log\frac{N}{DF(t,C)}$
Pivoted length normalization TF-IDF	$\sum_{t \in Q \cap D} \frac{1+\ln(1+\ln(TF(t,D)))}{1-s+\frac{s.doclen}{avgdoclen}} .TF(t,Q) .ln\frac{N+1}{DF\ (t,C)}$
Classic probabilistic	$\sum_{t \in q \cap D} 0.5+\frac{0.5\ .\ TF(t,D)}{MaxTF(t,D)} .log\frac{N-DF(t,D)}{DF(t,D)}$
Okapi BM25	$\sum_{t \in Q \cap D} log\frac{N}{DF(t,D)}.\frac{TF(t,D).(k+1)}{TF(t,D)+k.((1-b)+b.\frac{doclen}{avgdoclen})}$
Multi-Aspect TF-IDF	$\sum_{t \in q \cap D} (w.BRITF+(1-w).BLRTF).IDF(t,C).\frac{AEF(t,C)}{1+AEF(t,C)}$ However, since w = 1, $\sum_{t \in q \cap D} (BRITF).IDF(t,C).\frac{AEF(t,C)}{1+AEF(t,C)}$ Where BRITF = $\frac{RITF}{1+RITF}$;RITF = $\frac{log_2(1+TF(t,D))}{1+log_2(1+AvgTF(t,D))}$; BLRTF = $\frac{LRTF\ (t,D)}{1+LRTF(t,D)}$; LRTF = TF(t,D).$log_2(1+\frac{ADL(C)}{doclen})$ IDF = $log_{10}\frac{N}{DF(t,C)}$; AEF = $\frac{CollectionTF(t,C)}{DF(t,D)}$

Where,

t = term

D = document

C = collection of documents (corpus)

N = number of documents in corpus

Q = query

TF = term frequency

DF = document frequency

MaxTF = Maximum term frequency

CollectionTF = Collection term frequency

doclen = Document length

avgdoclen = Average document length

EXHIBIT 2: DATASET VARIABLES

Variable	Values
City	New York City, Washington D.C. etc
Entity	BSO - Businesses - Other
	BSF - Businesses - Financial and Insurance Services
	BSR - Businesses - Retail/Merchant
	EDU - Educational Institutions
	GOV - Government and Military
	MED - Healthcare - Medical Providers
	NGO - Nonprofit Organizations
Source	DatalossDB.com etc
State	California, Florida etc
Type of Breach	Insider (INSD) - Someone with legitimate access intentionally breaches information - such as an employee or contractor.
	Unintended disclosure (DISC) - Sensitive information posted publicly on a website, mishandled or sent to the wrong party via email, fax or mail.
	Hacking or malware (HACK) - Electronic entry by an outside party, malware and spyware.
	Payment Card Fraud (CARD) - Fraud involving debit and credit cards that is not accomplished via hacking. For example, skimming devices at point-of-service terminals.
	Physical loss (PHYS) - Lost, discarded or stolen non-electronic records, such as paper documents
	Portable device (PORT) - Lost, discarded or stolen laptop, PDA, smartphone, portable memory device, CD, hard drive, data tape, etc
	Stationary device (STAT) - Lost, discarded or stolen stationary electronic device such as a computer or server not designed for mobility.
	Unknown or other (UNKN)

EXHIBIT 3: ENTERPRISE MINER MODEL LAYOUT

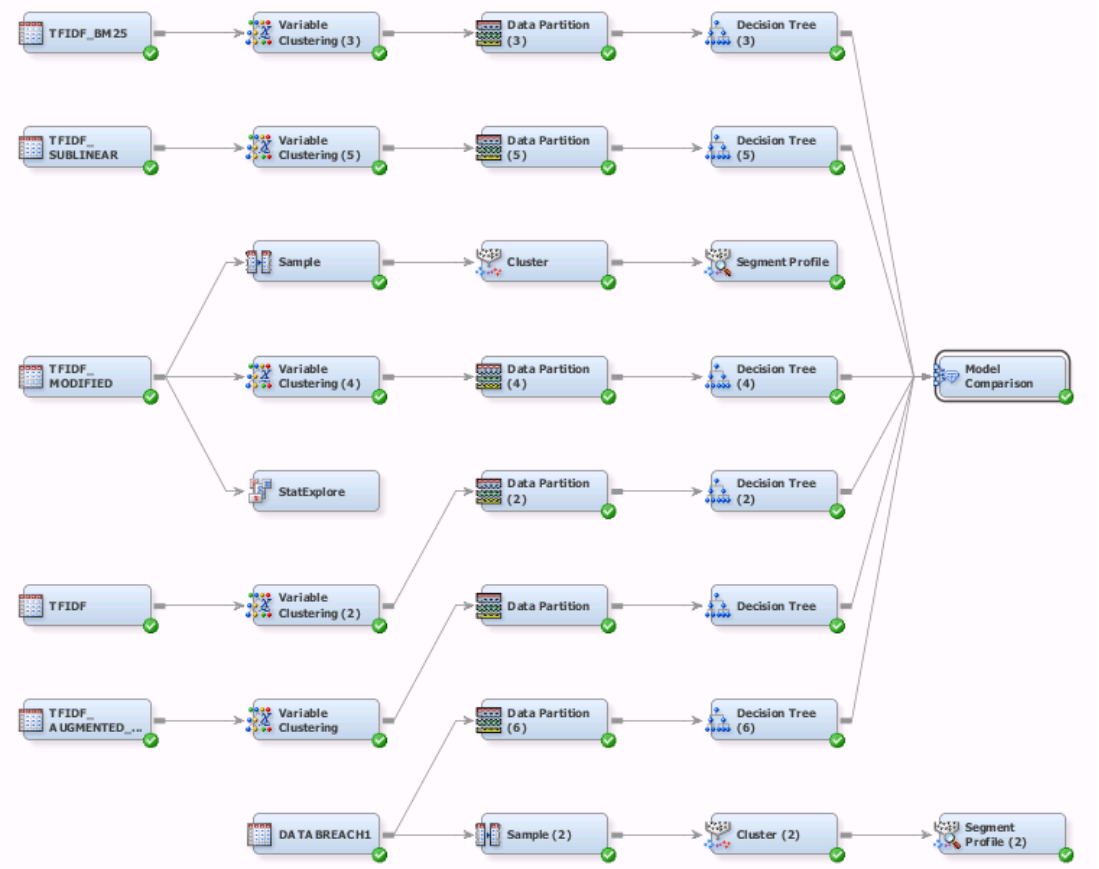


EXHIBIT 4: VARIABLE CLUSTERING PLOT

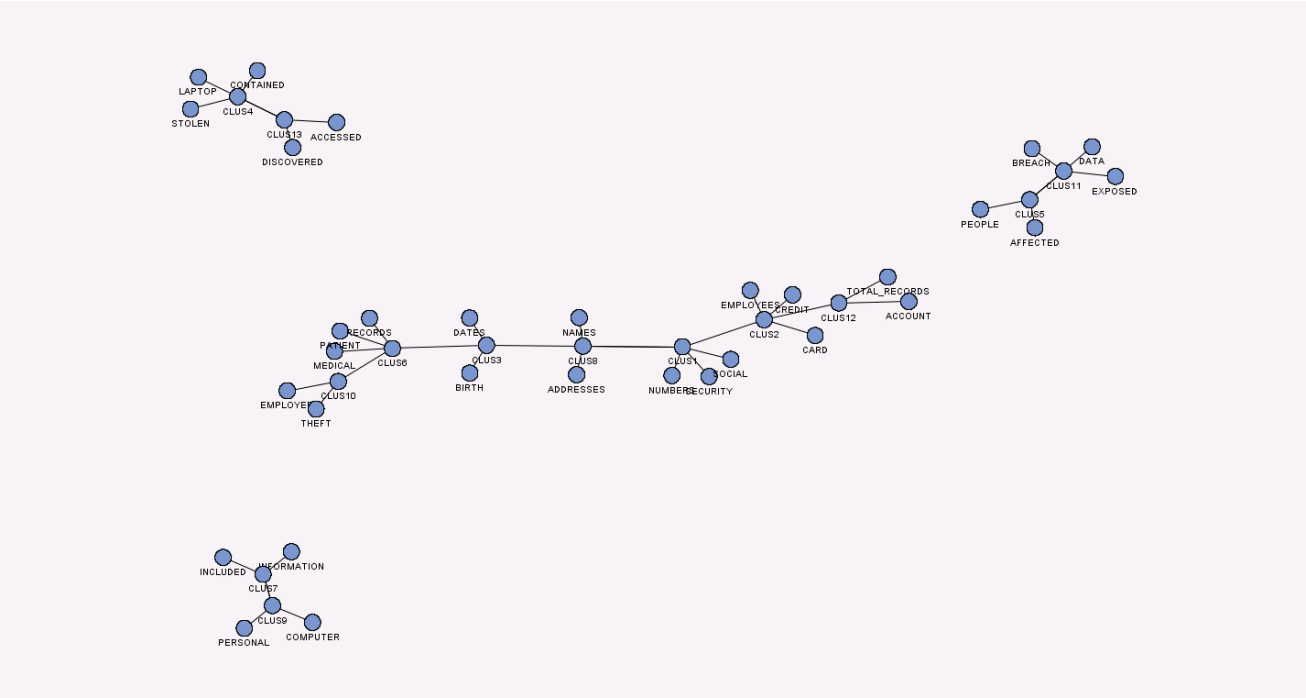
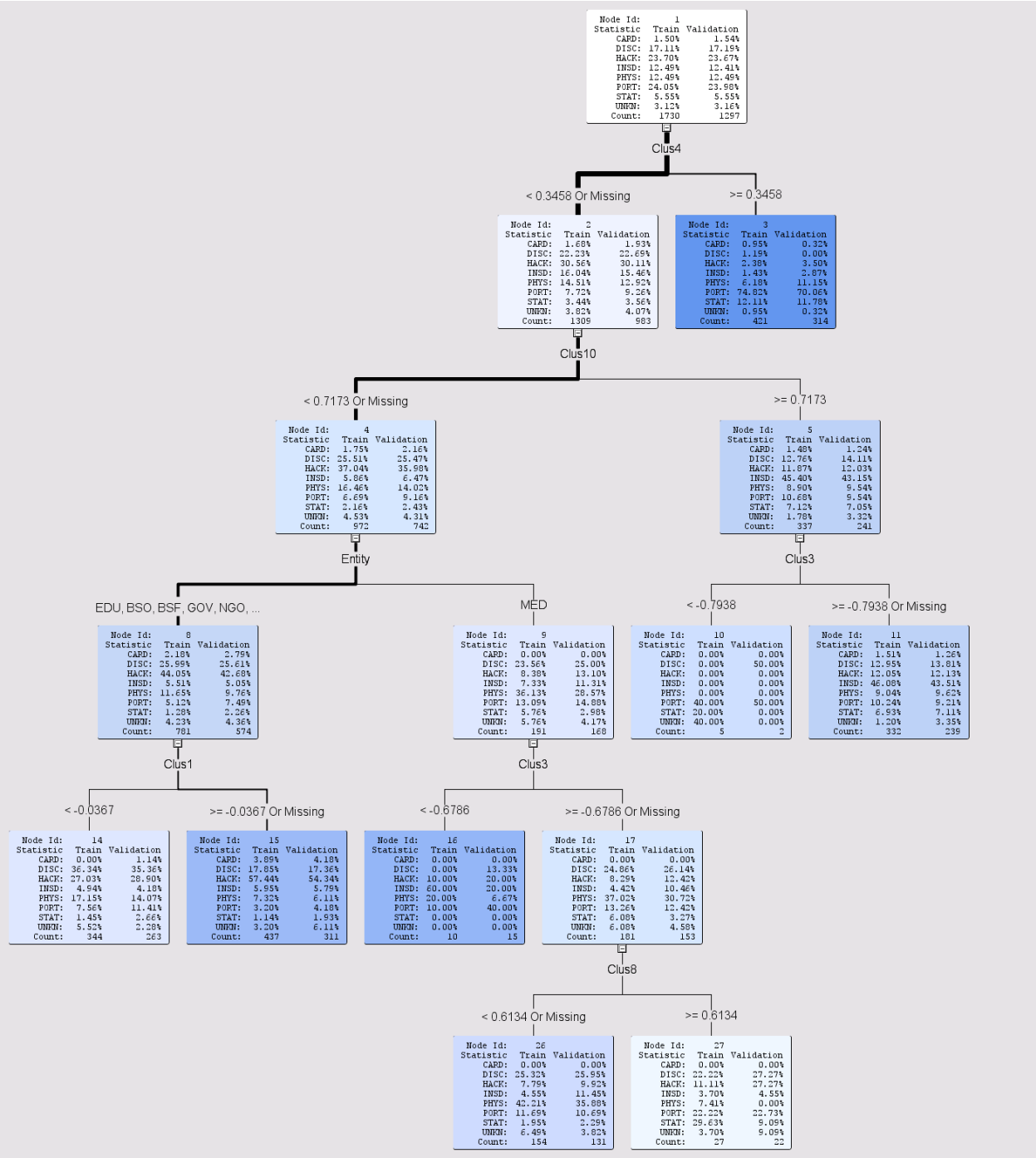


EXHIBIT 5: DECISION TREE GENERATED FROM FEATURES BUILT BY MULTI ASPECT WEIGHTING



Legend
Clus4 – +Stolen +Laptop +Contained
Clus10 – +Employee +Theft
Clus3 – +Birth +Dates
Clus1 – +Social +Security +Numbers
Clus8 - +Names +Addresses

EXHIBIT 6: CLUSTER SEGMENTS GENERATED FROM FEATURES BUILT BY MULTI-ASPECT WEIGHTING

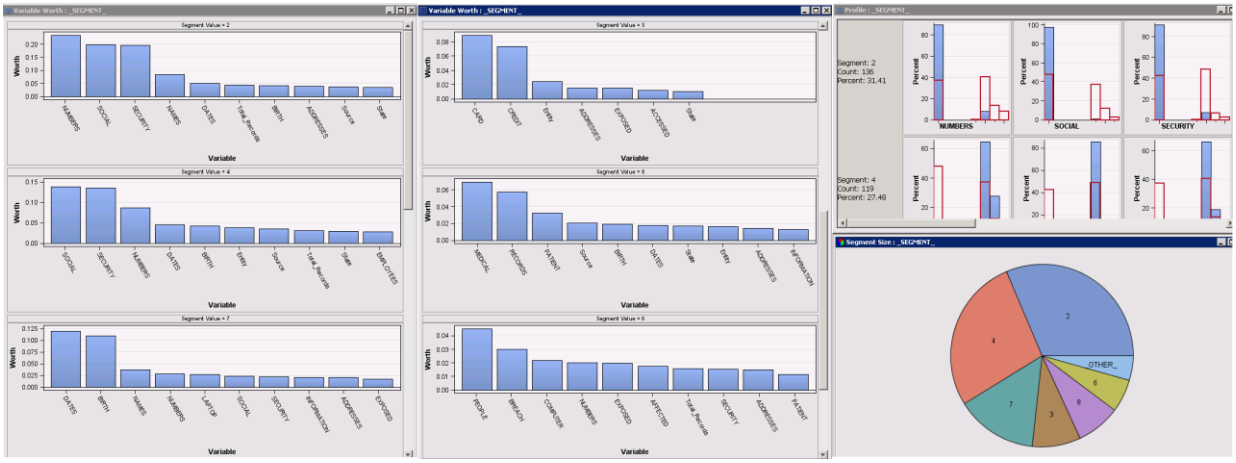
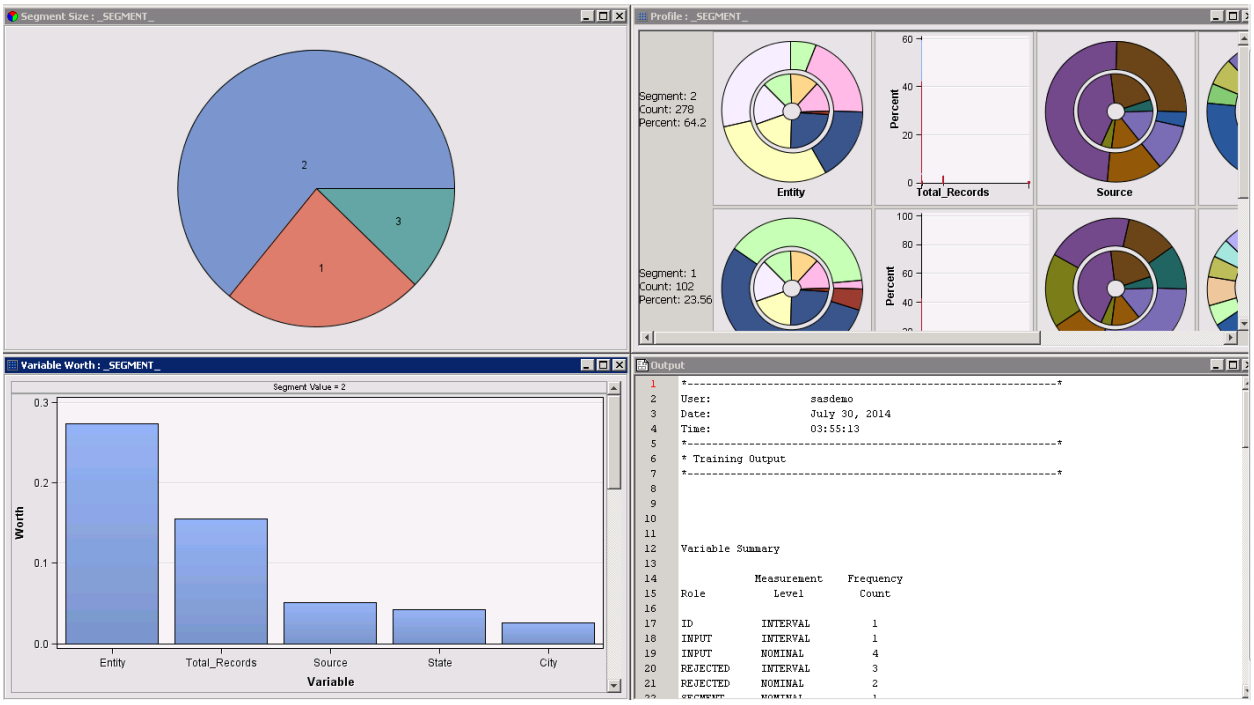


EXHIBIT 7: CLUSTER SEGMENTS GENERATED FROM ORIGINAL DATASET



REFERENCES

Cimiano, P., & Staab, S. (2004, December). Learning by Googling. ACM SIGKDD Explorations Newsletter, Volume 6 Issue 2, pp. 24-33.

Eberhardt, P. (2010). The SAS® Hash Object: It's Time To .find() Your Way Around. SAS Global Forum (p. 151). SAS Institute Inc.

Guyon, I., & Elisseeff, A. (2007). An Introduction to Feature Extraction. In I. Guyon, & A. Elisseeff, Feature Extraction (pp. 1-25). New York City: Springer Berlin Heidelberg.

Han, J. (2012). Data Mining: Concepts and Techniques (3rd edition). Waltham, MA: Morgan Kaufmann Publishers.

Manning, C., & Raghavan, P. (2008). Introduction to Information Retrieval. Cambridge: Cambridge University Press.

Paik, J. H. (2013). A Novel TF-IDF Weighting Scheme for Effective Ranking. ACM SIGIR Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, 343-352.

Salton, G., & Buckley, C. (1988). Term-Weighting Approaches in Automatic Text Retrieval. Information Processing & Management Vol.24 No. 5, 513-523.

SAS. (2012). Getting Started with Enterprise Miner 12.1. Cary, North Carolina, USA: SAS Institute Inc.

SAS. (2012). Getting Started with SAS Text Miner 12.1. Cary, North Carolina, USA: SAS Institute Inc.

SAS. (2014, August 13). Using the Hash Object. Retrieved from SAS(R) 9.2 Language Reference: Concepts, Second Edition: http://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/viewer.htm#a002585310.htm

Singhal, A., Buckley, C., & Mitra, M. (1996). Pivoted Document Length Normalization. ACM SIGIR Proceedings of the 19th annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 21-29.

Warner-Freeman, J. K. (2011). I cut my processing time by 90% using hash tables - You can do it too! NESUG (p. BB16). NESUG.

Whissell, J., & Clarke, C. (2011). Improving document clustering using Okapi BM25 feature weighting. Information Retrieval Volume 14 Issue 5, 466-487.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Ila Nitin Gokarn
Ingokarn.2011@sis.smu.edu.sg