SANDIA REPORT
SAND2010-7392
Unlimited Release
Printed October 2010

A Toolkit for Detecting Technical Surprise

Michael W. Trahan, Mark C. Foehse

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd.
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2010-7392
Unlimited Release
Printed October 2010

A Toolkit for Detecting Technical Surprise

Michael W. Trahan
Emergent Threats Department

Mark C. Foehse
Proliferation Sciences Department

Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185-MS1207

Abstract

The detection of a scientific or technological surprise within a secretive country or institute is very difficult. The ability to detect such surprises would allow analysts to identify the capabilities that could be a military or economic threat to national security. Sandia's current approach utilizing ThreatView has been successful in revealing potential technological surprises. However, as data sets become larger, it becomes critical to use algorithms as filters along with the visualization environments.

Our two-year LDRD had two primary goals. First, we developed a tool, a Self-Organizing Map (SOM), to extend ThreatView and improve our understanding of the issues involved in working with textual data sets. Second, we developed a toolkit for detecting indicators of technical surprise in textual data sets. Our toolkit has been successfully used to perform technology assessments for the Science & Technology Intelligence (S&TI) program.

ACKNOWLEDGMENTS

This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.

CONTENTS

EQUATIONS

FIGURES

NOMENCLATURE

API	Application Programming Interface
BMU	Best Matching Unit
COTS	Commercial Off The Shelf
CSV	Comma Separated value
DOE	Department of Energy
DTT	Data Trace Tool
GUI	Graphical User Interface
HPC	High Performance Computing
LDRD	Laboratory Directed Research and Development
LSA	Latent Semantic Analysis
NER	Named Entity Recognition
NLTK	Natural Language Toolkit
S&TI	Science & Technology Intelligence
SNA	Social Network Analysis
SNL	Sandia National Laboratories
SOM	Self-Organizing Map
STANLEY	Sandia Text AnaLysis Extensible LibrarY
TF-ICF	Term Frequency-Inverse Corpus Frequency
TF-IDF	Term frequency-Inverse Document Frequency
VTK	Visualization ToolKit
WWW	World Wide Web

1. INTRODUCTION

The detection of a scientific or technological surprise within a secretive country or institute is very difficult. The ability to detect such surprises would allow analysts to identify the capabilities that could be a military or economic threat to our national security. Sandia's current approach utilizing ThreatView has been successful in revealing potential technological surprises. However, ThreatView has limitations.

ThreatView presents data visually, which allows analysts to identify trends, patterns, and relationships that otherwise are very difficult to detect. However, this detection is dependent upon the analyst: some analysts see the patterns; some analysts miss the patterns (false negatives); and still other analysts see patterns that are not real (false positives). In addition, ThreatView uses a single algorithm (LSA) to cluster the data set. There is no way to compare its results to an alternative clustering or to measure the quality of the clustering. We have addressed these limitations by developing a data mining toolkit, which can be used independently or as an extension to ThreatView.

As data sets become larger, it becomes critical to use algorithms as filters along with the visualization environments. Our toolkit provides a suite of algorithms to filter the data so that analysts are presented with less, but more relevant, data increasing the chance of detecting a scientific or technological surprise.

2. BUILDING A TOOL: SELF-ORGANIZING MAPS

Our first effort was to build a tool to extend ThreatView and improve our understanding of the issues involved in working with textual data sets. We chose to implement a Self-Organizing Map (SOM).

The self-organizing map (SOM) is a type of artificial neural network first described by Professor Teuvo Kohonen of the Helsinki University of Technology, Laboratory of Computer and Information Science, Neural Networks Research Centre, in the early 1980s. The SOM provides a way of representing multidimensional data in a two-dimensional space, while maintaining the data's topological relationships. SOMs are frequently used as visualization aids. They can make it easy for us to see relationships between vast amounts of multidimensional data. SOMs have been successfully used in many applications, including: speech recognition (Kohonen's original area of research); bibliographic classification; image browsing systems; medical diagnosis; seismic data interpretation; data compression; and, environmental modeling.

SOMs have many advantages. They are easy to understand (especially compared to most other neural network architectures). They work very well on a large number of problem classes and they are adaptive – they cannot be over-trained.

There are, however, some disadvantages to SOMs. It can be hard to get the "right" data: You must have a value for every dimension of every input vector. Every SOM is different and finds different similarities in the data. In the final map, every vector is surrounded by similar vectors; however, similar vectors are not always near each other. And, especially during training, SOMs are computationally expensive.

2.1. Data Pre-Processing

The data for this application is records of scientific and technical articles. The data is provided as a Microsoft Excel CSV-format file. Most of the fields consist of natural language text. This text must be pre-processed into a form (numeric) that is usable by the self-organizing map (SOM).

The pre-processor, called CSV2SOM, was written in the Python scripting language. It reads the records from the CSV-format data file and allows the user to generate a set of training and testing data for the SOM. The graphical user interface (GUI) is built with the wxPython toolkit (wxPython is a wrapper for the wxWidgets cross-platform GUI API, which is written in C++). The natural language text is processed using the Natural Language Toolkit (NLTK). See Figure 1, Figure 2, Figure 3, and Figure 4 for screen shots of the pre-processor.

NLTK provides the user with information about the data set. For each field, the NLTK parses the data to determine the number of empty records, the number of tokens, the number of unique words, the diversity score, the number of common words, and the number of unusual words.

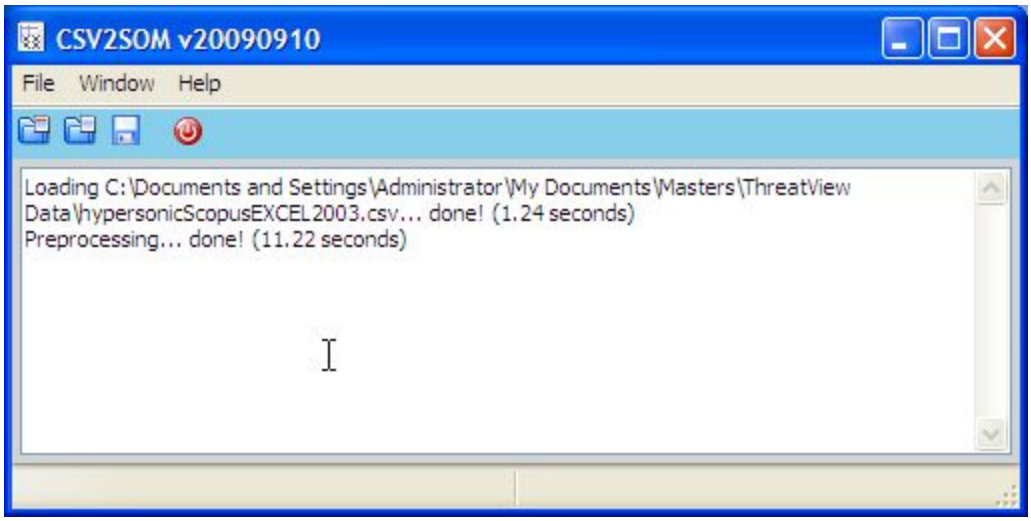

Figure 1. CSV2SOM – main window.

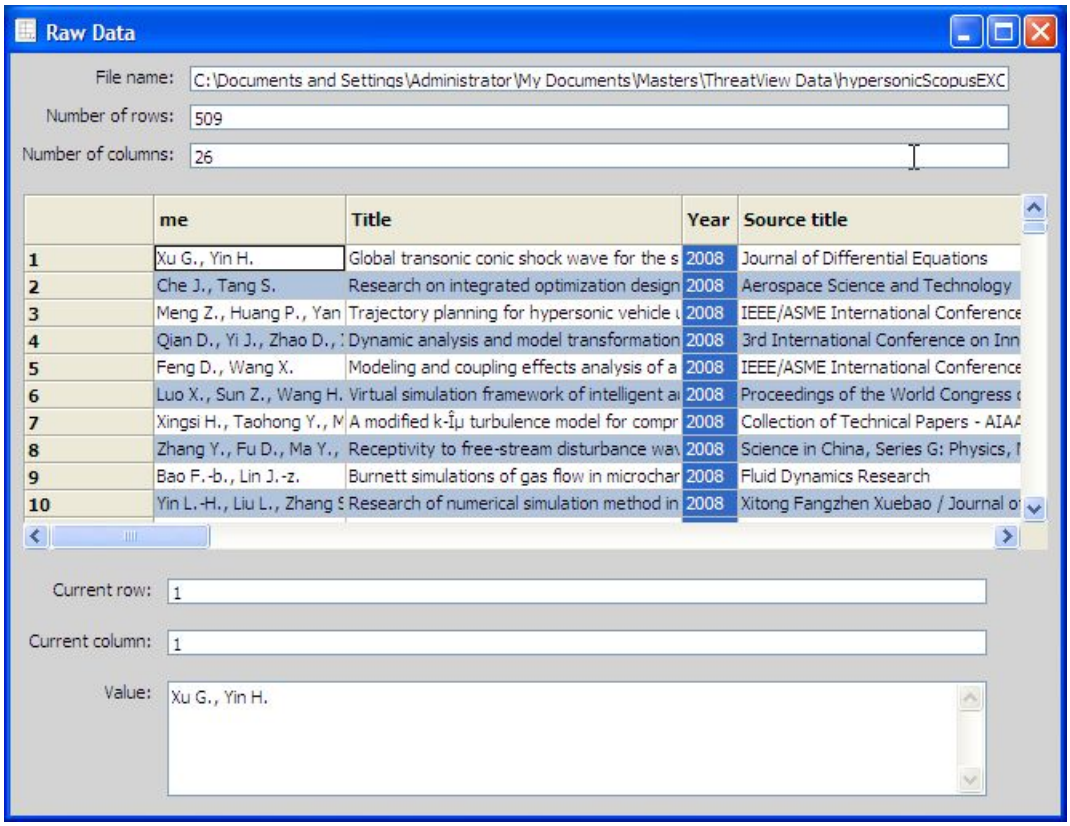

Figure 2. CSV2SOM – raw data window.

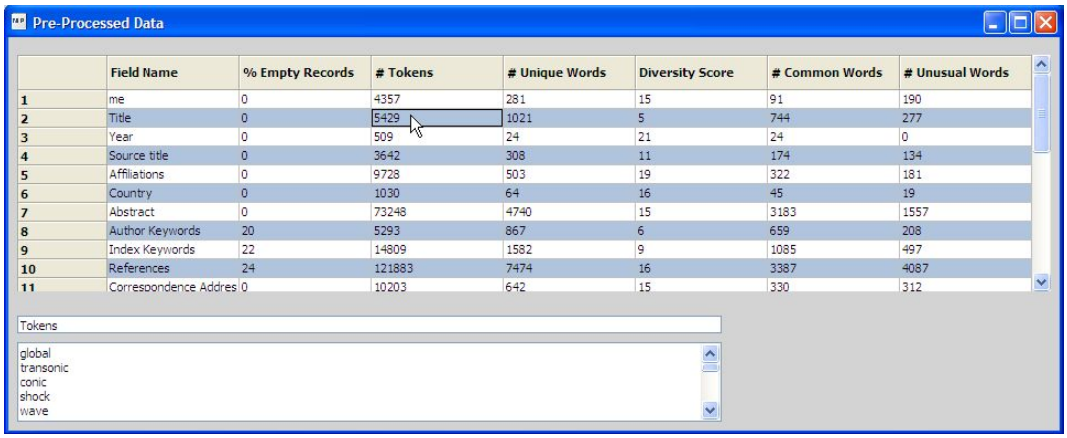

Figure 3. CSV2SOM – pre-processed data window.

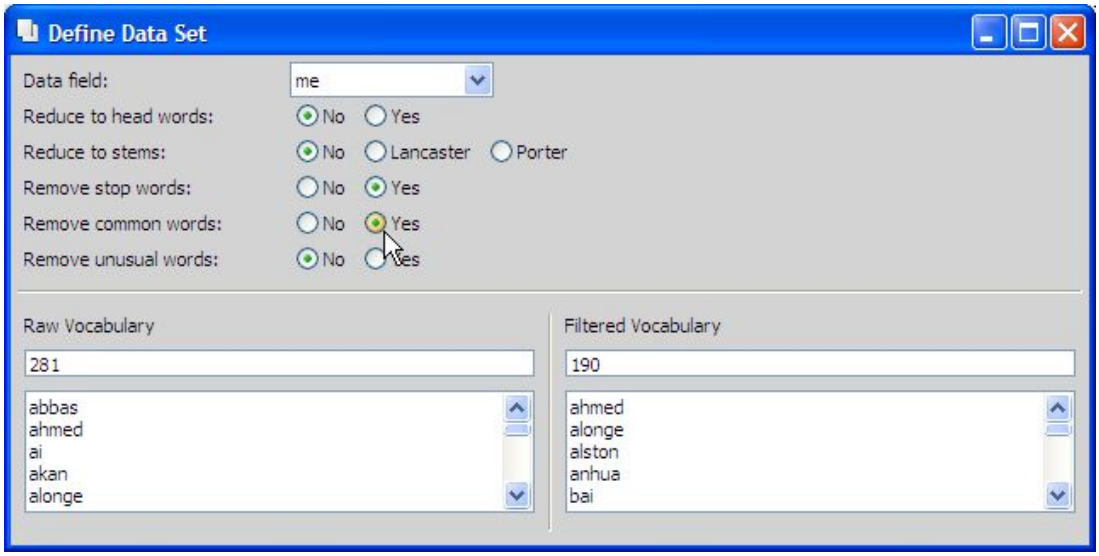

Figure 4. CSV2SOM – define data set window.

To convert the data to a form usable by the SOM, the NLTK allows the user to: reduce the words to their head words; reduce the words to their stems (using the Lancaster or Porter stemmers); remove stop words (of, the, etc.); remove common words; and/or remove unusual words. In addition, the user can choose to remove high frequency words and/or low frequency words. Finally, the user can specify the percentage of the data set to be used for training the SOM (empty records are ignored); the remainder of the data set is automatically generated for testing the SOM. In the training and testing data sets, each record is represented as a vector of dimension *n*, where each component represents the Term Frequency-Inverse Document Frequency (TF-IDF) of the associated word.

2.2. Training

For this application, we used the public-domain SOM_PAK software package. SOM_PAK is written in C++ and is provided by the Helsinki University of Technology, Laboratory of

Computer and Information Science, Neural Networks Research Centre. See Figure 5 and Figure 6 for a listing of typical SOM_PAK training commands.

```
# Map initialization
randinit -din ex.dat -cout ex.cod -xdim 12 -ydim 8 -topol hexa
      -neigh bubble -rand 123

# Map training (ordering phase)
vsom -din ex.dat -cin ex.cod -cout ex.cod -rlen 1000 -alpha 0.05
      -radius 10

# Map training (fine tuning)
vsom -din ex.dat -cin ex.cod -cout ex.cod -rlen 10000 -alpha 0.02
      -radius 3

# Evaluation of the quantization error
qerror -din ex.dat -cin ex.cod
```

Figure 5. SOM_PAK – typical commands for training a basic SOM.

```
vfind

      Give the number of trials:                    1000
      Give the input data file name:                ex.dat
      Give the input test file name:                ex.dat
      Give the output map file name:                ex.cod
      Give the topology type:                       hexa
      Give the topology type:                       bubble
      Give the topology type:                       12
      Give the y-dimension:                         8
      Give the training length of first part:       1000
      Give the training rate of first part:         0.05
      Give the radius in first part:                10
      Give the training length of second part:      10000
      Give the training rate of second part:        0.02
      Give the radius in second part:               3
```

Figure 6. SOM_PAK – typical commands for training an optimized SOM.

The SOM is created from a 2D lattice of "nodes", each of which is fully connected to the input layer. There are no lateral connections between the nodes. Each node has a position (an x, y coordinate) in the lattice and contains a vector of weights (also called a reference vector). If the input vector V has n dimensions (v_1, v_2, v_3, … v_n), then each node's weight vector W has n dimensions (w_1, w_2, w_3, … w_n).

The SOM's training process is a series of steps repeated over many iterations:

1. Initialize each node's weights to small, standardized, random values ($0 < w < 1$).

2. Choose a vector at random from the set of training data and present it to the map.

3. Examine every node to determine which node's weights are most like the input vector. One method is to calculate the Euclidean distance between each node's weight vector and the input vector. The winner is the node whose weight vector is the closest to the input vector. The winning node is commonly known as the Best Matching Unit (BMU).

$$dist = \sqrt{\sum_{i=1}^{n}(v_i - w_i)^2}$$

Where n is the dimension of the input and weight vectors, V (v_1, v_2, v_3, … v_n) is the input vector, and W (w_1, w_2, w_3, … w_n) is the node's weight vector.

Equation 1. Calculate the Distance Between an Input Vector and a Node's Weight Vector.

4. Calculate the nodes in the BMU's neighborhood. The neighborhood radius is typically initialized to the "radius" of the lattice and decays over time (typically with an exponential decay function). Any nodes found within this neighborhood radius are deemed to be inside the BMU's neighborhood.

$$\sigma(t) = \sigma_0 e^{\left(-\frac{t}{\lambda}\right)}$$

Where σ_0 is the width of the neighborhood at time t_0, λ denotes a time constant, and t is the current time step (iteration).

Equation 2. Calculate the BMU's Neighborhood Size.

5. Adjust the weights of the BMU and its neighbors to make them more like the input vector. The closer a node is to the BMU, the more its weights are altered.

$$\theta(t) = e^{\left(-\frac{dist^2}{2\sigma^2(t)}\right)}$$
$$W(t+1) = W(t) + \Theta(t)L(T)\left[\left(V(t)\right) - W(t)\right]$$

Where $dist$ is the distance between the input and weight vectors, σ is the neighborhood size, Θ represents the amount of influence a node's distance from the BMU has on its learning, t represents the time step, and L is the learning rate (which decreases with time).

Equation 3. Adjust the Weights of the BMU and Its Neighbors.

6. Adjust the learning rate. It decays over time, just like the neighborhood radius.

$$L(t) = L_0 e^{\left(-\frac{t}{\lambda}\right)}$$

Where λ denotes a time constant and t is the current time step (iteration).

Equation 4. Update the Learning Rate.

7. Repeat steps 2-6 for N iterations.

Unlike many other types of neural networks, the SOM does not need a target output to be specified. The SOM can learn to classify data without supervision. The area of the lattice which contains the node whose weight most closely matches the input vector is selectively optimized to more closely resemble the data for the input vector's class. After many iterations (from an initial distribution of random weights), the SOM converges into a map of stable zones. Each zone is effectively a feature classifier, so the SOM's graphical output is a type of feature map of the input space. After training, any new, previously unseen input vectors presented to the SOM will stimulate nodes in the zone with similar weight vectors.

2.3. Metrics

For our work with the SOM, we selected two metrics widely used in the SOM community: the average quantization error and the average topology preservation error.

The Average Quantization Error measures the average distance between each data vector and its best-matching unit (BMU). It evaluates the fitting of the SOM to the data. An optimal map is expected to yield the smallest average quantization error.

$$E_q = \frac{1}{N}\sum_{i=1}^{N}\|V_i - w_i\|$$

Equation 5. Calculate the SOM's Average Quantization Error.

Where N is the number of input vectors, V is an input vector, and w is the weight vector of the BMU for vector V.

The Average Topology Preservation Error measures the proportion of all data vectors for which the first and second best-matching units (BMUs) are not adjacent. The lower the topological error is, the better the SOM preserves the topology of the data.

$$E_t = \frac{1}{N}\sum_{i=1}^{N} u(V)$$

Where N is the number of input vectors and V is an input vector. $u(V_i) = 1$, if the first and second BMUs of V_i are not adjacent. Otherwise, $u(V_i) = 0$.

Equation 6. Calculate the SOM's Average Topology Preservation Error.

2.4. Visualization

SOM_PAK provides four main visualization programs. *Visual* plots the trajectory of the best-matching units (BMUs) over time. *Sammon* generates the Sammon mapping from n-dimensional input vectors to 2-dimensional points on a plane whereby the distances between the image vectors tend to approximate to Euclidean distances of the input vectors. *Planes* generates plots of a selected component plane (or all planes) imaging the values of the components using gray levels. *Umat* (see Figure 7) visualizes the distances between reference vectors of neighboring map units using gray levels; darker colors represent smaller distances. See Figure 8 for a listing of typical SOM_PAK visualization commands. The plots produced by SOM_PAK are static; the user cannot interact with them.

We have found the plots generated by *Umat* are useful with our typical data sets; the plots generated by *Visual*, *Sammon*, and *Planes* are not.

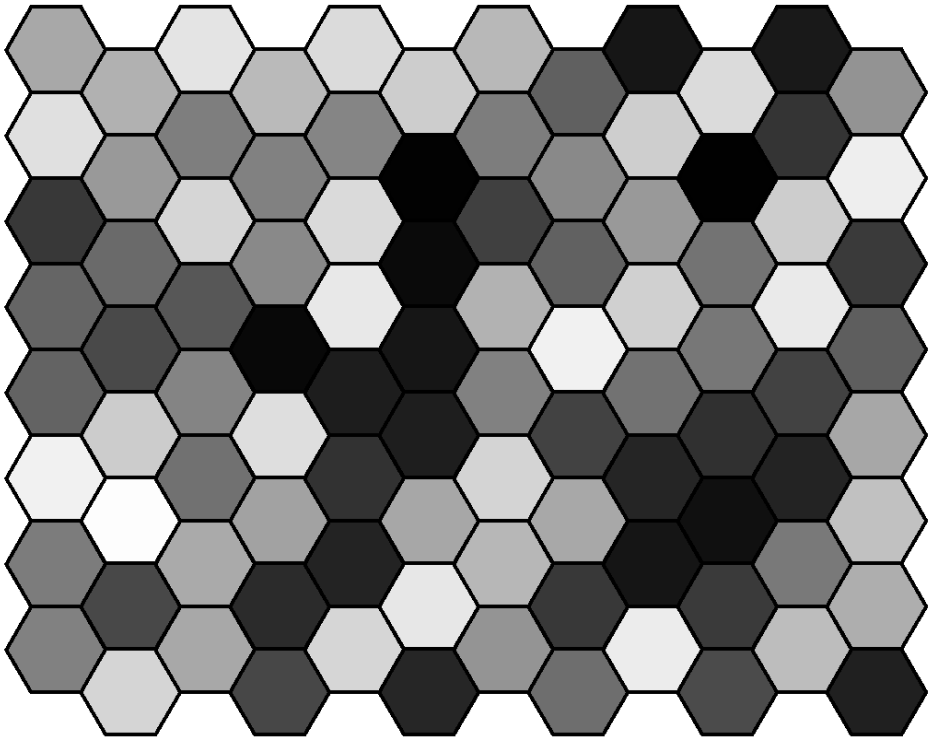

Figure 7. SOM_PAK – *Umat* plot.

```
# Map visualization
vcal -din ex_fts.dat -cin ex.cod -cout ex.cod
visual -din ex_ndy.dat -cin ex.cod -dout ex.nvs
visual -din ex_fdy.dat -cin ex.cod -dout ex.fvs
sammon -cin ex.cod -cout ex.sam -rlen 100
planes -cin ex.cod -plane -din ex.dat
umat -cin ex.cod -ps 1 > umat.eps
umat -cin ex.cod -average -ps 1 > umat_average.eps
umat -cin ex.cod -median -ps 1 > umat_median.eps
```

Figure 8. SOM_PAK – typical commands for visualizing a SOM.

3. BUILDING A TOOLKIT

Our second effort was to build a toolkit to perform technology assessments in support of Sandia's Science and Technology Intelligence (S&TI) program. Our goals were to extend our capabilities beyond ThreatView by: 1) being able to analyze and visualize both structured and unstructured data sets; 2) being able to analyze and visualize larger data sets; and 3) being able to use alternative algorithms (non-LSA) and visualizations (non-landscape). We investigated tools developed internally at Sandia National Laboratories and Oak Ridge National Laboratory, commercial-off-the-shelf (COTS) tools, and open source tools.

3.1. Sandia-Developed Tools

Sandia has developed a number of tools which are well suited to our technology surprise toolkit. Some of the tools are based on the STANLEY (Sandia Text AnaLysis Extensible LibrarY) text analysis engine (developed by Cognitive and Exploratory Systems as part of the Cognitive Grand Challenge), while other tools are based on the Titan toolkit (developed by Visualization and Data Services as part of the Network Grand Challenge).

3.1.1 Stanley-Based Tools

The STANLEY (Sandia Text AnaLysis Extensible LibrarY) text analysis engine uses a Log-Entropy algorithm (see Equation 7) to calculate term dominance and a cosine similarity algorithm to compare documents (see Equation 8).

$$LogEntropy = \log(1 + f_{ij}) \times \left(1 + \left(\frac{\sum_j (p_{ij} \log(p_{ij}))}{\log n}\right)\right)$$

Where $p_{ij} = \frac{f_{ij}}{g_i}$ and f_{ij} is the frequency of term i in document j and g_i is the number of documents in which term i occurs.

Equation 7. Log-Entropy.

$$\cos\theta = \frac{\sum_{i=1}^{m} a_i b_i}{\sqrt{\sum_{i=1}^{m} a_i^2}\sqrt{\sum_{i=1}^{m} b_i^2}}$$

Where m is the number of terms and a_i and b_i are the log-entropy scores of term i in documents a and b respectively.

Equation 8. Cosine Similarity.

Cognitive Science and Applications has developed five tools based on STANLEY. We are currently using two of these tools (Data Trace Tool and Cognitive Spider) and are continuing evaluations of another three tools (Archive Assistant, Navigator, and Sounding Board).

Another STANLEY-based tool we use is LDRDView, the predecessor to ThreatView, which was developed by Visualization and Data Services.

3.1.1.1 Data Trace Tool

When performing a technology assessment, one of the problems we encounter is tracking the search: how and why did an analyst reach a particular website or document? In the past, the search was tracked manually: a tedious, error-prone process. The Data Trace Tool (DTT) automates this process. As an analyst performs a web search, DTT (see Figure 9) automatically creates a graphical map of the websites the analyst accesses. Analysts are able to annotate the map to highlight particularly interesting sites, to add explanatory notes, to draw connections between nodes to show relationships, etc. The map can be edited, trimmed, and customized for inclusion in presentations and documents. In addition, maps can be shared allowing analysts to continue or build on previous work.

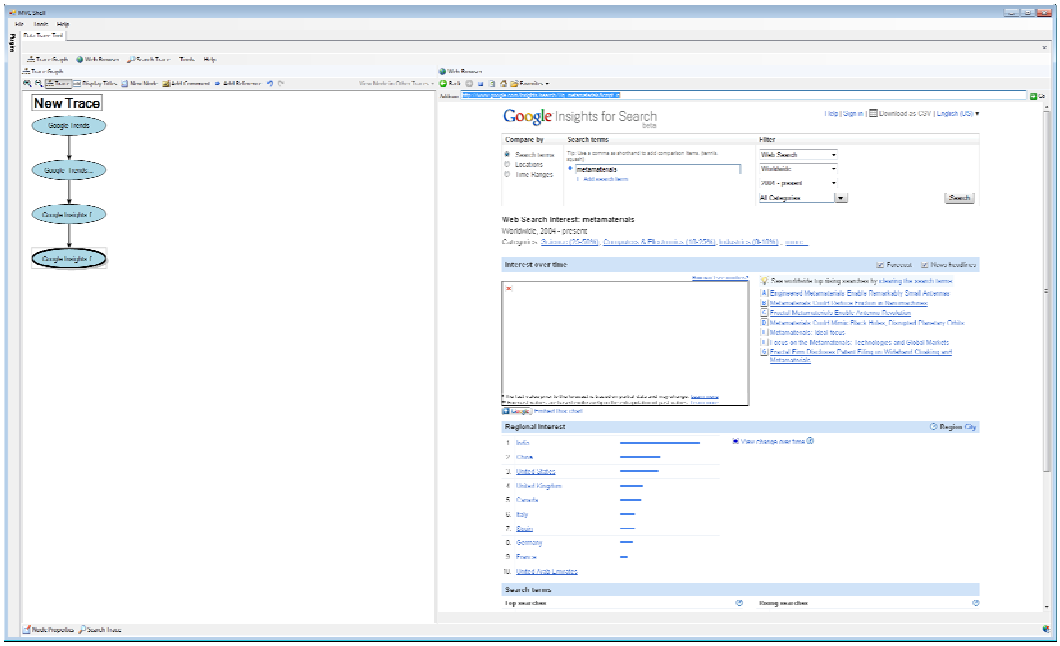

Figure 9. Data Trace Tool – main window.

3.1.1.2 Cognitive Spider

Another common problem we encounter when performing a technology assessment is extending a document set: if an analyst has a set of documents related to a particular technology, how can he/she find similar documents? In the past, the analyst would use a search engine to manually search for similar documents: another tedious process. The Cognitive Spider automates this process.

The Cognitive Spider builds a STANLEY text model of the existing unstructured-text documents and uses this model to direct a crawl to websites which contain documents of interest. The resulting set of documents can be automatically prioritized based on their similarity to the original documents. All interesting pages and documents (PDFs, Microsoft Office documents, etc.) are automatically stored on the local hard drive for later analysis.

The Cognitive Spider provides an extensive set of reports which document its crawls. It provides tables of overall statistics information (# pages found; # unique pages: # unique and interesting pages), the top 100 pages, the top 100 pages by top-level domain (e.g., .com, .edu, .gov, etc.), etc. It provides graphs of interesting versus non-interesting pages, host distribution, top-level host distribution, average interest level by host and by top-level domain, and the distribution of queries by domain. The Cognitive Spider shows what search terms were used and how often each term was used. The Cognitive Spider also provides error reporting including graphs of error types and distribution of errors across hosts and tables of hosts which are missing.

3.1.1.3 LDRDView

One of biggest problems when performing technology assessments is visualizing and exploring a data set. LDRDView (see Figure 10) is a software tool for visualizing a collection of documents, exploring relationships between them, and examining the content of individual documents. By evaluating document content and assigning coordinates to each document based on its similarity to other documents, LDRDView graphically displays a corpus of documents as a landscape of hills and valleys or as a graph of nodes and links. A suite of data analysis tools facilitates in-depth exploration of the corpus as a whole and the content of each individual document. LDRDView was developed in response to the need for better R&D program management tools that could provide insight into the large LDRD project portfolio. The program integrates the SNL-patented VxInsight visualization software and the STANLEY (Sandia Text AnaLysis Extensible LibrarY) text analysis engine into the visualization process. Both VxInsight and STANLEY were developed in support of previous LDRD projects. Although largely replaced by its successor, ThreatView, LDRDView is still a useful tool.

3.1.2 Titan-Based Tools

The Visualization ToolKit (VTK) by Kitware, Inc. is an open source, freely available software system for 3D computer graphics, image processing, and visualization used by thousands of researchers and developers around the world. Visualization and Data has developed Titan, an information visualization library built as an extension to VTK. Titan supports the ingestion, processing, and visualization of informatics data.

Visualization and Data has developed three tools based on Titan: P1 (NetView), P2, and P3. At this time, we are only using P2. Interactive System Simulation & Analysis and Information Engineering have developed ThreatView, which is also Titan-based.

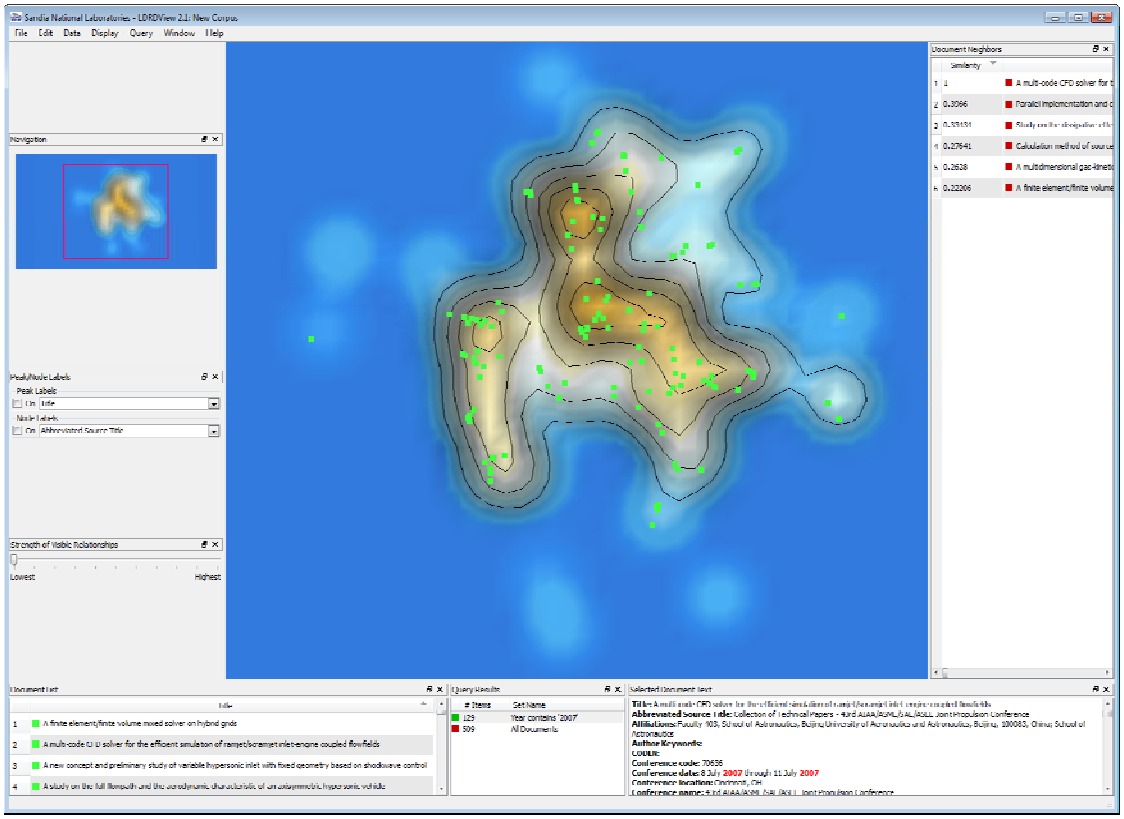

Figure 10. LDRDView – main window.

3.1.2.1 P2

P2, or Prototype 2, is the second prototype developed by the Networks Grand Challenge LDRD. It was designed to facilitate analysts working with large volumes of text documents through the use of named entity recognition (NER) and the graphing of entity-to-document relationships. In addition, documents in the corpus are clustered — grouped together — so the user can see how documents are related based upon topical content. The P2 user interface (UI) is divided into six views, as shown in Figure 11.

The Document Text view is in the center of the window (see Figure 12). It is the main reading view and contains the current document. If the document is a web page (HTML), then any web links in the document are active — the user need only select the link to visit the targeted web page.

The Document Clusters view is in the upper left-hand corner of the window (see Figure 13). When a set of documents is ingested and analyzed by P2, a clustering algorithm groups like documents together based upon similarity of content. Each category is named using the most common topics found within the documents in that category.

The Corpus Maps view in the lower left-hand corner of the window contains a graph of the document clusters in the Document Clusters view. The Corpus Map view uses two different user-selectable graph layouts. The tree-ring graph layout (see Figure 14) places documents

around the circumference of a ring, color codes sections of the ring's circumference based upon topic and then links similar documents to one another using arcs that transit the interior of the ring. The "force-directed" graph layout (see Figure 15) displays the same information in a traditional network graph.

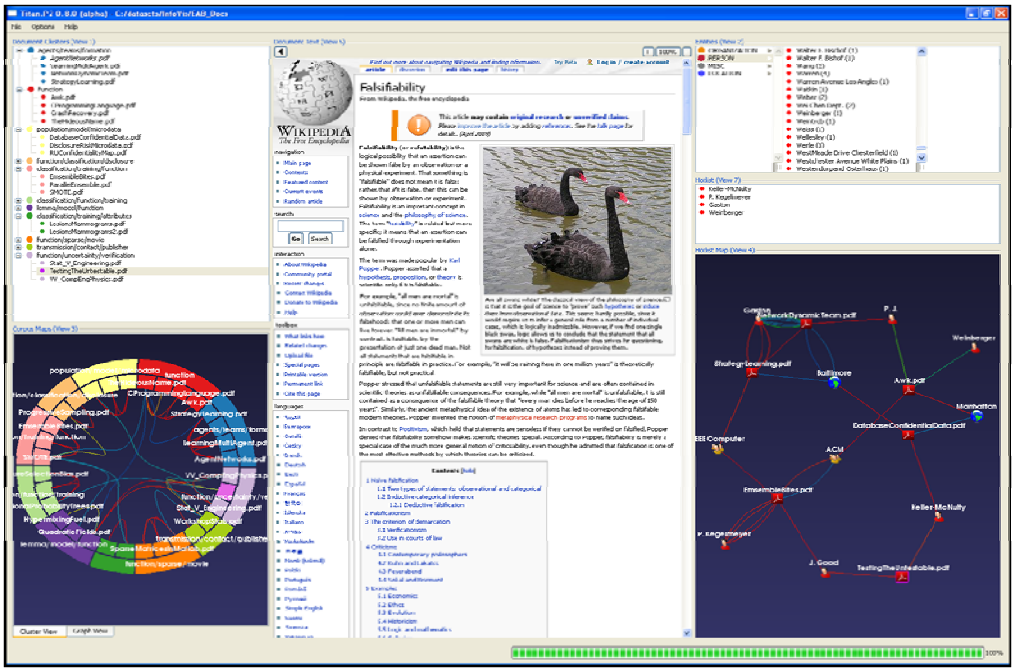

Figure 11. P2 – the main window.

Figure 12. P2 – the Document Text view.

Figure 13. P2 – the Document Clusters view.

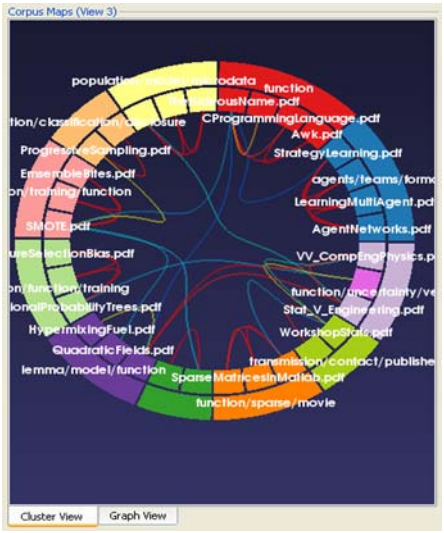

Figure 14. P2 – the Corpus Map window (tree-ring layout).

Figure 15. P2 – the Corpus Map window ("force-directed" graph layout).

The Entities view (see Figure 16) in the upper right-hand corner of the window displays the results of the named entity recognition (NER) process. When a corpus of documents is opened, a NER process identifies people, places, and things and then lists the recognized entities in the Entities view.

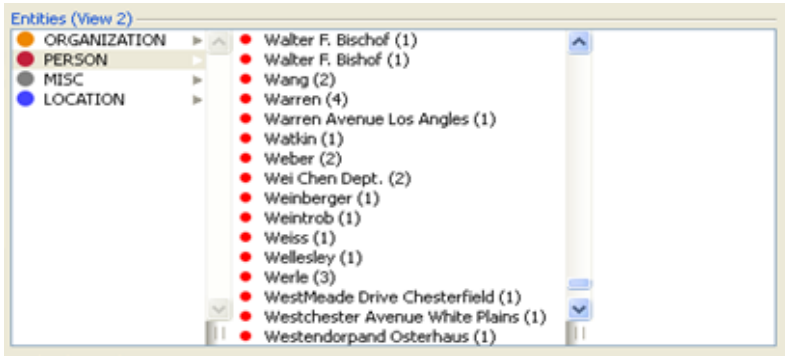

Figure 16. P2 – the Entities view.

The Hotlist view (see Figure 17) in the middle right-hand portion of the window is used to create a list of named entities for graphing in the Hotlist Map view (see Figure 18). To create the list, a user selects an entity of interest in the Entities view and drags the entity into the Hotlist view. As soon as the user has dragged two or more entities into the Hotlist view, P2 creates a graph in the Hotlist Map view depicting how the entities are related to one another, that is, in which document or documents do the selected entities appear?

Figure 17. P2 – the Hotlist view.

As stated above, the Hotlist Map view displays relationships between entities and documents. Text analytics have not advanced to the point where a computer algorithm can reliably identify relationships between people (e.g., Bill is Mary's husband) by "reading" a text. It takes a human to determine those types of relationships. The graph in the Hotlist Map view facilitates a user's finding and vetting relationships by informing the user where to look (i.e., which documents contain references to the entities of interest). This relationship discovery is not limited to people-to-people relationships; the user can explore the relationships between any selected entities: people, organizations, places, etc.

Figure 18. P2 – the Hotlist Map view of entity-to-document relations.

3.1.2.2 ThreatView

ThreatView (see Figure 19) is a software tool for visualizing a collection of documents, exploring relationships between them, and examining the content of individual documents. By evaluating document content and assigning coordinates to each document based on its similarity to other documents, ThreatView graphically displays a corpus of documents as a landscape of hills and valleys or as a graph of nodes and links. A suite of data analysis tools facilitates in-depth exploration of the corpus as a whole and the content of each individual document. ThreatView is the successor to LDRDView and has many new capabilities including the ability to handle larger data sets.

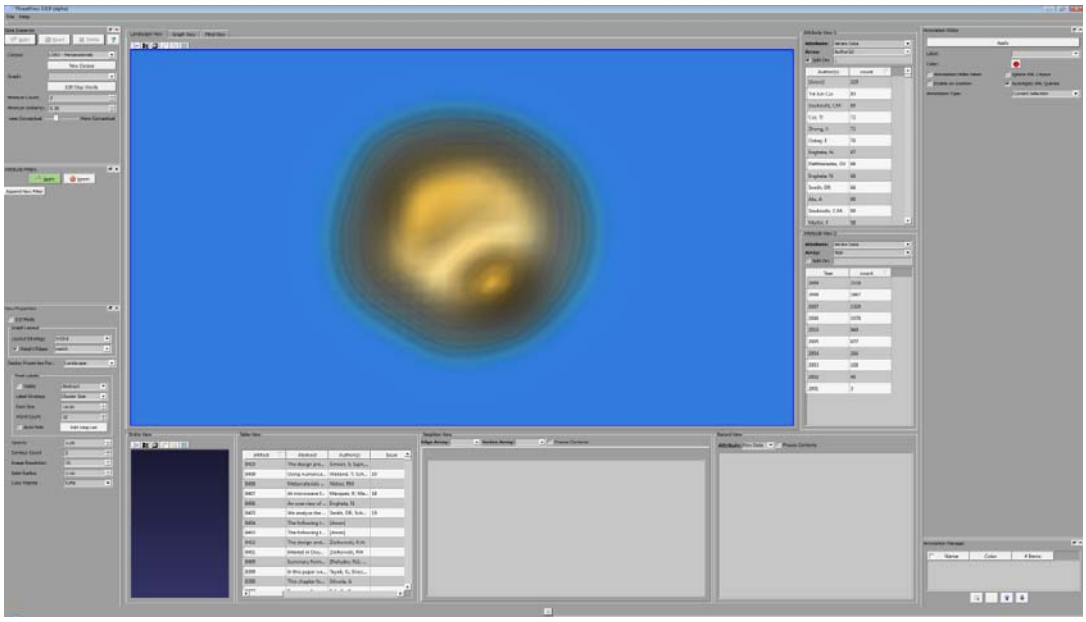

Figure 19. ThreatView – main window.

ThreatView uses the Latent Semantic Analysis (LSA) algorithm to analyze relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms.

3.2. Oak Ridge-Developed Tools

Oak Ridge National Laboratory's Applied Software Engineering Research Group has developed a knowledge discovery engine called Piranha. Piranha uses an advanced, intelligent-agent architecture to cluster large unstructured data sets with incremental clustering handled using a threshold-based solution. Using Piranha, an analyst can find similar documents, find or eliminate duplicate documents, create representative samples of a set of documents, etc. Piranha can be used for single-term word analysis or multi-term phrase analysis.

Term Frequency-Inverse Document Frequency (TF-IDF) is a commonly used weight for text mining (see Equation 9 and Equation 10). TF-IDF is difficult to parallelize because the IDF requires the entire document set to be known in advance. Piranha uses a patented Term Frequency-Inverse Corpus Frequency (TF-ICF) algorithm (see Equation 9 and Equation 11), which allows documents to be processed in parallel. TF-ICF differs from TF-IDF in that a known, representative corpus is used to determine the inverse corpus frequency (instead of using the actual document set being processed).

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

Where $n_{i,j}$ is the number of occurrences of term t_i in document d_j and the denominator is the sum of the occurrences of all terms in document d_j.

Equation 9. Term Frequency.

$$W_{i,j} = \log_2\left(tf_{i,j} + 1\right) * \log_2\left(\frac{N}{n}\right)$$

Where $tf_{i,j}$ is the term frequency of term t_i in document d_j (see Equation 9), N is the number of documents in the corpus, and n is the number of documents in the corpus containing term t_i.

Equation 10. Term Frequency-Inverse Document Frequency.

$$W_{ij} = \log_2\left(tf_{ij} + 1\right) * \log_2\left(\frac{C+1}{c+1}\right)$$

Where $tf_{i,j}$ is the term frequency of term t_i in document d_j (see Equation 9), C is the number of documents in the known corpus, and c is the number of documents in the know corpus containing term t_i.

Equation 11. Term Frequency-Inverse Corpus Frequency.

Figure 20 shows a Cluster Document Nodes view of 10 news articles related to Amphetamines. The center node represents the top of the tree. The leaves of the tree represent the documents. The remaining nodes represent levels of the tree with threshold values of document similarity. Notice that although most documents are unrelated, two sets of two documents are closely related.

Piranha provides an alternative method of clustering and visualizing data sets and has the ability to run in parallel. It can cluster larger data sets quickly, but it can only visualize 8,192 documents at a time.

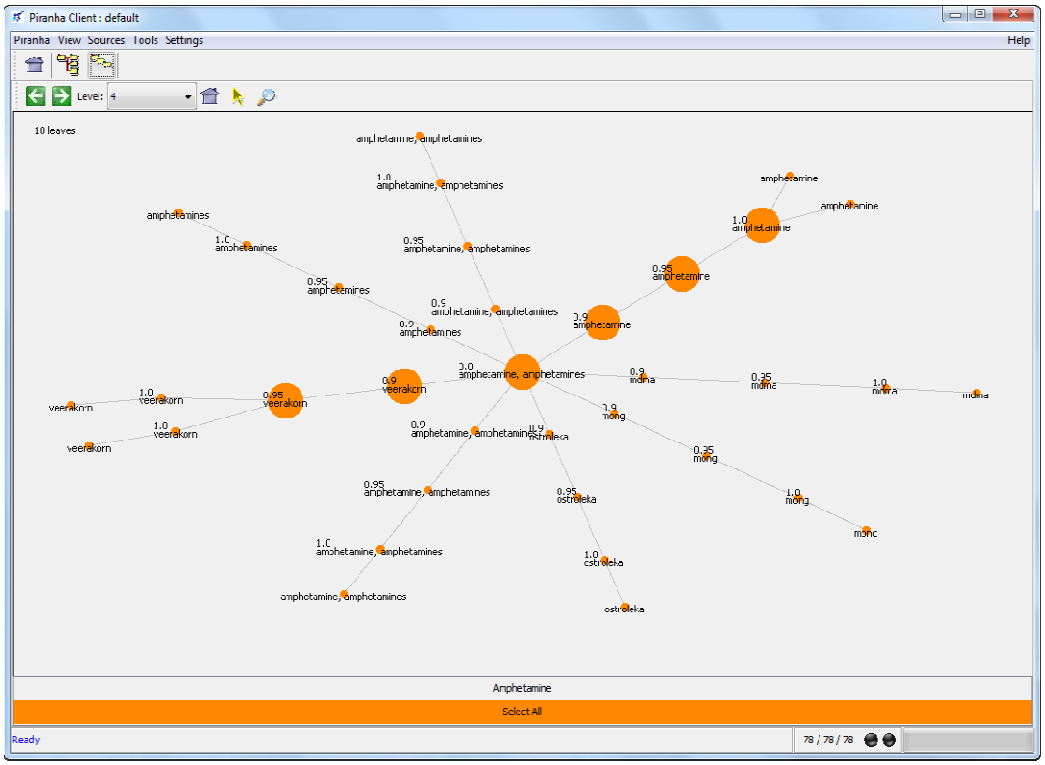

Figure 20. Piranha – plot of clustered documents.

3.3. COTS (Commercial Off The Shelf) Tools

We have found several COTS tools which are well suited to our technology surprise toolkit.

3.3.1 COTS Analysis and/or Visualization Tools

We have found several COTS tools which provide additional analysis and/or visualization capabilities: dtSearch Desktop, Analyst's Notebook, Text Chart, Google Trends, and Google Insights for Search.

3.3.1.1 dtSearch Desktop

dtSearch Desktop (see Figure 21) from dtSearch Inc. (www.dtsearch.com) is a "desktop" file search tool. It is extremely useful as well as easy to use. In addition, it offers far more sophisticated search capabilities than Microsoft Explorer or the built-in Windows search utility or web tools like Google or Bing.

Imagine you need to write a technology assessment. Using your favorite browser, you use Google to search for web pages on your particular topic of interest. You visit multiple pages of interest from the Google search results page and save a copy of each relevant page to your local hard drive. Along the way, you find relevant files (PDFs, Microsoft PowerPoint files, Microsoft Word files, etc.) and save them to your hard drive as well. After a long day of searching and

scanning web pages, you have amassed a collection of several hundred documents. Now you are ready to begin your actual in-depth topic research.

How will you conduct your research? Will you simply read all the documents, one after another, or will you try to work your way through the documents in some systematic fashion, perhaps using the same search terms you used when retrieving the documents from the WWW using Google? This is where dtSearch comes in — in fact, this is where it absolutely excels.

To use dtSearch, a user creates an index of the files and folders of interest (see Figure 22). To do this, a user "points" dtSearch to multiple files or folders and instructs dtSearch to index the contents. dtSearch then performs a full-text index of the contents of all the files and folders specified — and it does it very quickly.

Once the index has been created, the user can now search the index for documents of interest (see Figure 23). Using, perhaps, the same terms the user entered into Google, the user can now search for documents containing those terms on their local hard drive (recall the user saved a copy of all the web pages and files they discovered via Google). dtSearch uses a simple Boolean search grammar (as does Google, Bing, Yahoo, etc.), but its search grammar is more flexible. After the user enters the terms of interest, dtSearch returns a list of the documents found. Depending upon how the dtSearch options are set, the first document amongst those found is opened by dtSearch and all the search terms found within the document are highlighted in yellow (see Figure 24). The user can read (and advance) through the file search term by search term using their mouse or the space bar on the computer keyboard. The user can create sophisticated Boolean searches using standard operators AND and OR as well as the proximity operator with "w/x." For example, term A within 5 words of term B would be written as "A w/5 B" (see Figure 25). Again, when the user executes this search the results are highlighted within all matching documents (see Figure 26).

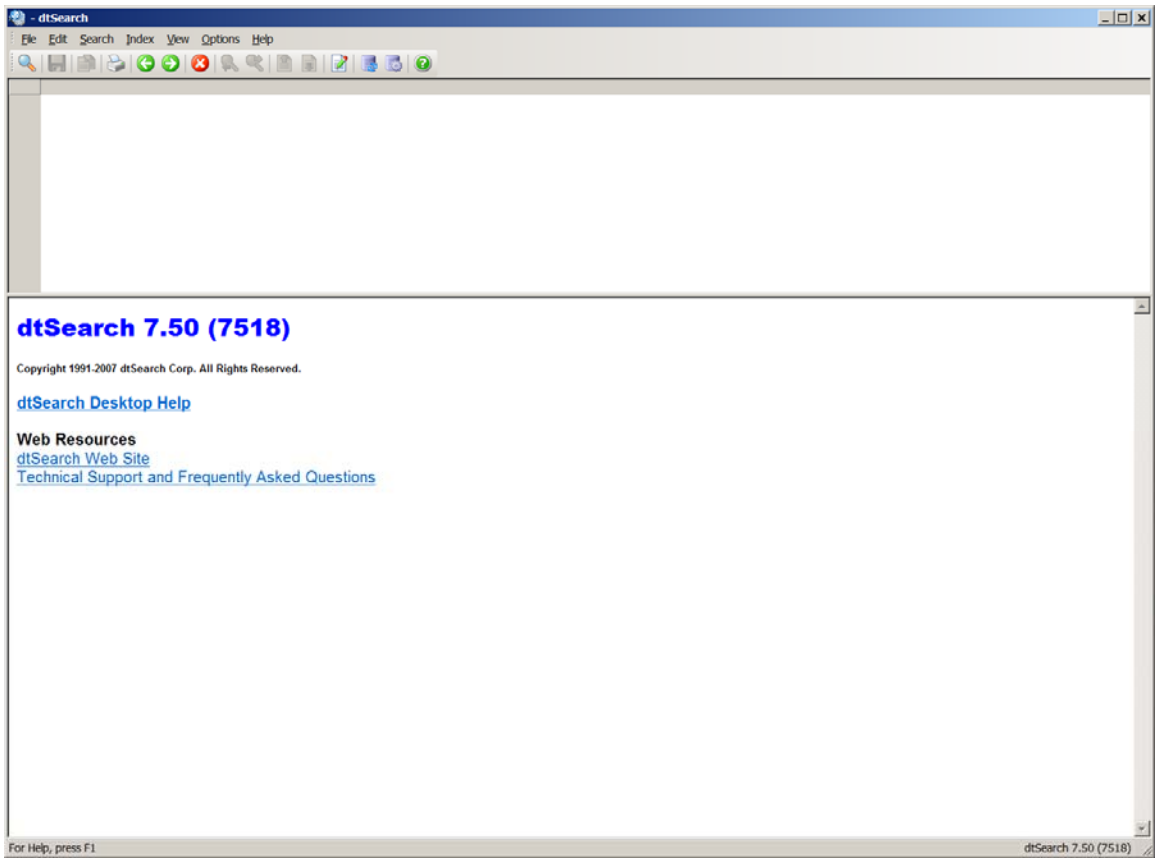

Figure 21. dtSearch – start-up window.

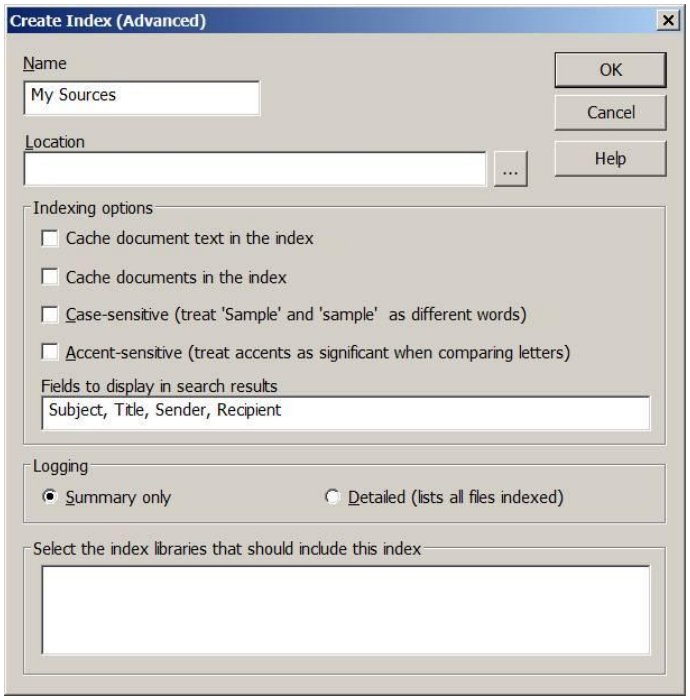

Figure 22. dtSearch – creating an index.

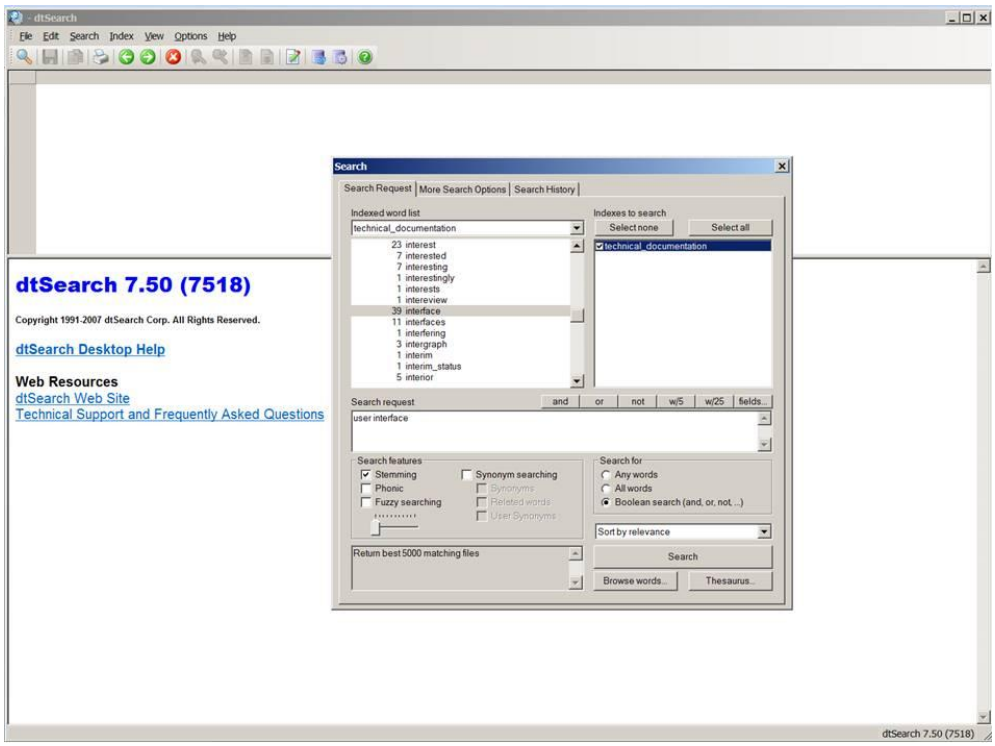

Figure 23. dtSearch – a simple search.

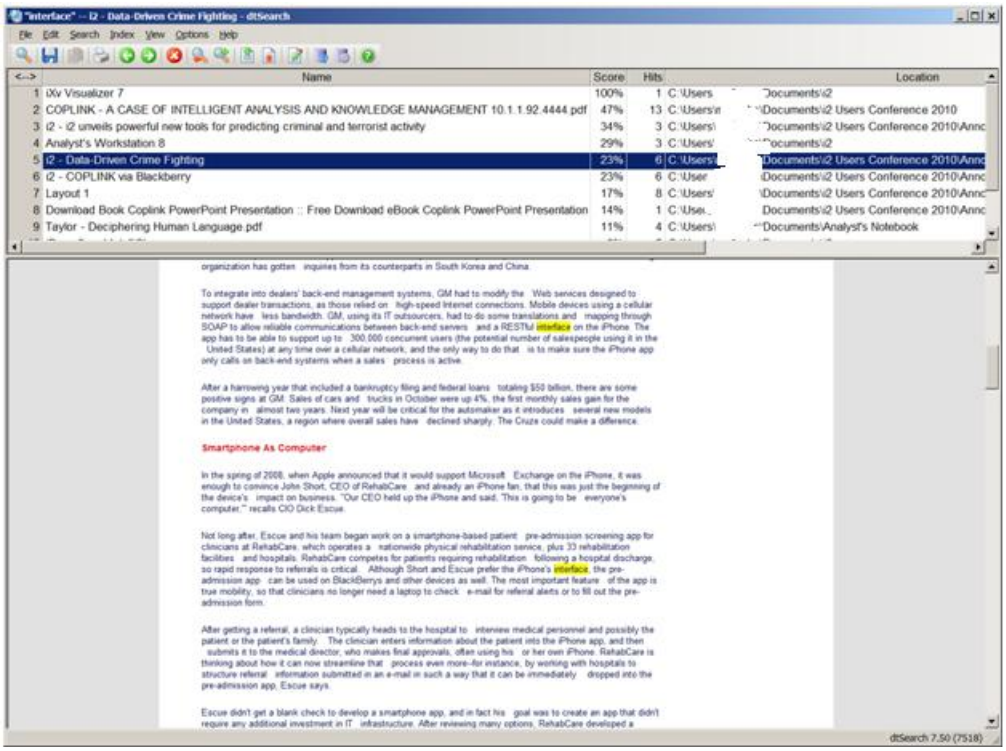

Figure 24. dtSearch – search terms highlighted in context.

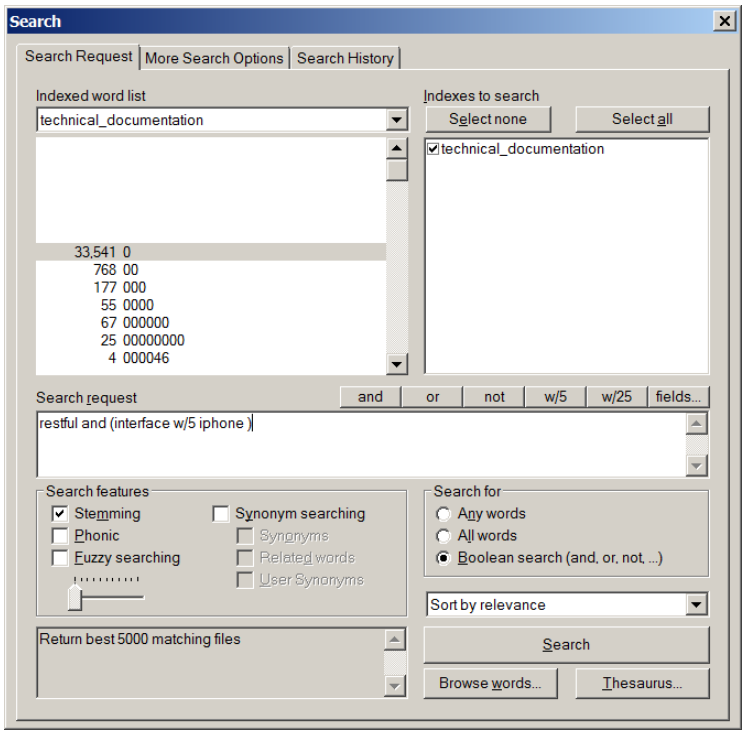

Figure 25. dtSearch – a complex search.

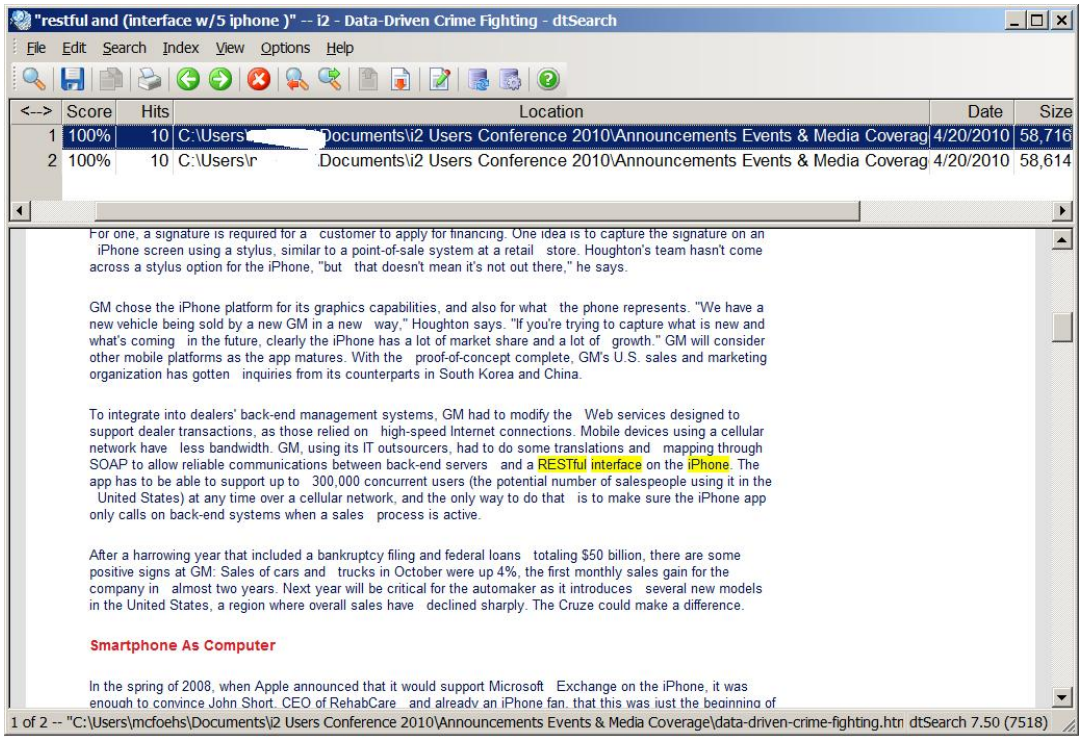

Figure 26. dtSearch – results of a complex query.

3.3.1.2 Analyst's Notebook and TextChart

For many situations, the best way to visualize information is a graph. A company's organizational chart, a map of the U.S. highway system, a supply chain, and connections between people are often depicted as graphs. Analyst's Notebook and TextChart, offered by i2 Inc. (http://www.i2group.com), provide the ability to create graphs and visualize the relationships between entities in the graph. This allows the user both to understand the graph as well as to deduce interesting new facts about the graph.

Analyst's Notebook is the de facto standard software application within the U.S. intelligence and law enforcement communities for building graphs. It facilitates the creation of graphs that depict relationships between entities as well as relationships in time. The user can choose from alternate graph construction schemes so that information can be represented in its most natural manner. For example, if the relationships are simply associations, the user can choose a corresponding graph layout (see Figure 27). If the relationships are predominantly temporal, the user can choose an alternate graph layout that emphasizes the significance of time (see Figure 28). The user can "highlight" certain types of information through conditional formatting (color, boldness of lines, etc.). This visual emphasis of certain details within the graph helps users to understand the contents of the graph.

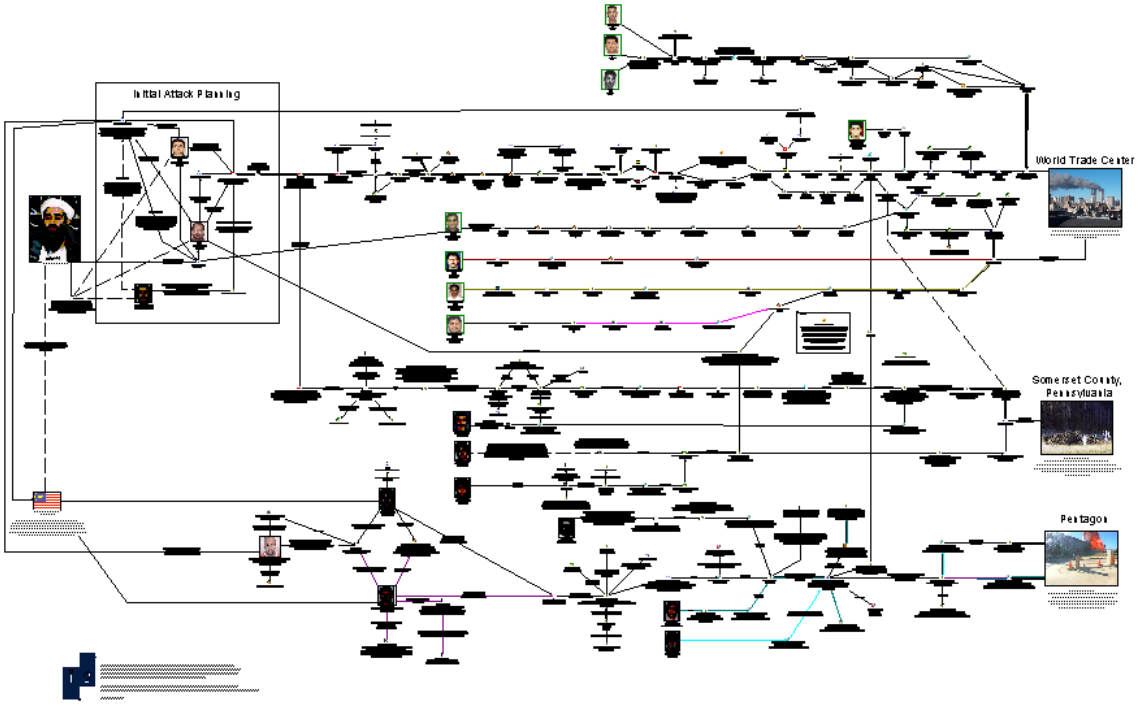

Figure 27. Analyst's Notebook – a graph showing relationships between Osama Bin Laden and the 9/11 attackers.

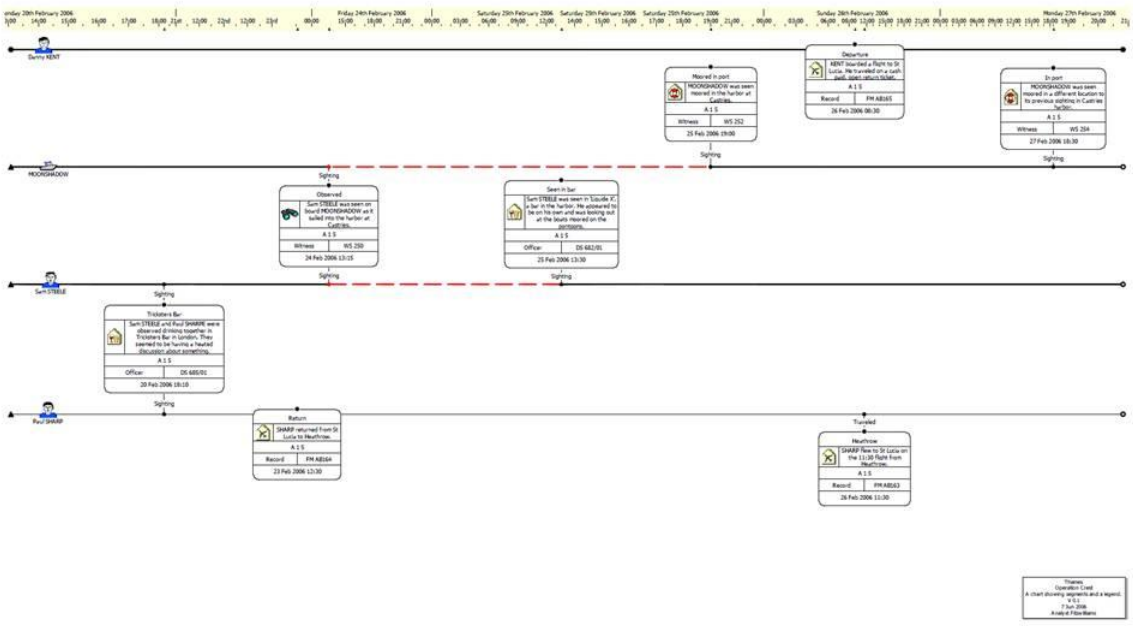

Figure 28. Analyst's Notebook – a theme line showing events ordered by time.

Analyst's Notebook has a Social Network Analysis (SNA) capability. SNA uses analytic graph methods to describe a graph and its nodes. For example, who are the most highly connected people in a graph? Who controls the flow of information from one part of a graph to another? Who seems to be "central" to the graph? SNA attempts to understand the roles of individuals within a social network.

Analyst's Notebook has the ability to integrate with ArcServer by ESRI (Environmental Systems Research Institute, Inc. the leading purveyor of GIS (Geographical Information System) software in the U.S.). ArcServer is a GIS data service platform; it serves GIS information to other software applications. The integration of Analyst's Notebook and ArcServer means that a user can include geo-coding information in a graph and then, because the information is geo-coded, display the graph on a map — the user can see where in the world events took place. Similarly, the user can select a region on a map and then see what data (people, places, and events) are related to that physical location.

The primary weakness of Analyst's Notebook is its reliance on visualization for analysis. The single exception to this is SNA. Given that exception, the user must *see* the graph in order to discern the characteristics of the graph. The analytic process of Analyst's Notebook is all about visualizing graphs. There exists a rich set of discrete mathematical algorithms that can be used to both describe and discern the characteristics of a graph. At this time, Analyst's Notebook does not incorporate this type of analysis.

TextChart provides a (nearly) drag-and-drop interface that allows users to create link charts (graphs) from unstructured text documents. Users can create graph nodes or graph edges.

TextChart adds attributes to both links and nodes so users can ascertain the source of the information at a later date.

A user creates graphs in TextChart by opening text documents within the application, selecting the text of interest like "Terry Nichols," dragging the selected text over an icon (a male icon in this case), and then dragging the "union" of the text and icon into the graph window (see Figure 29). This process creates a graph node. A similar process is used for creating links.

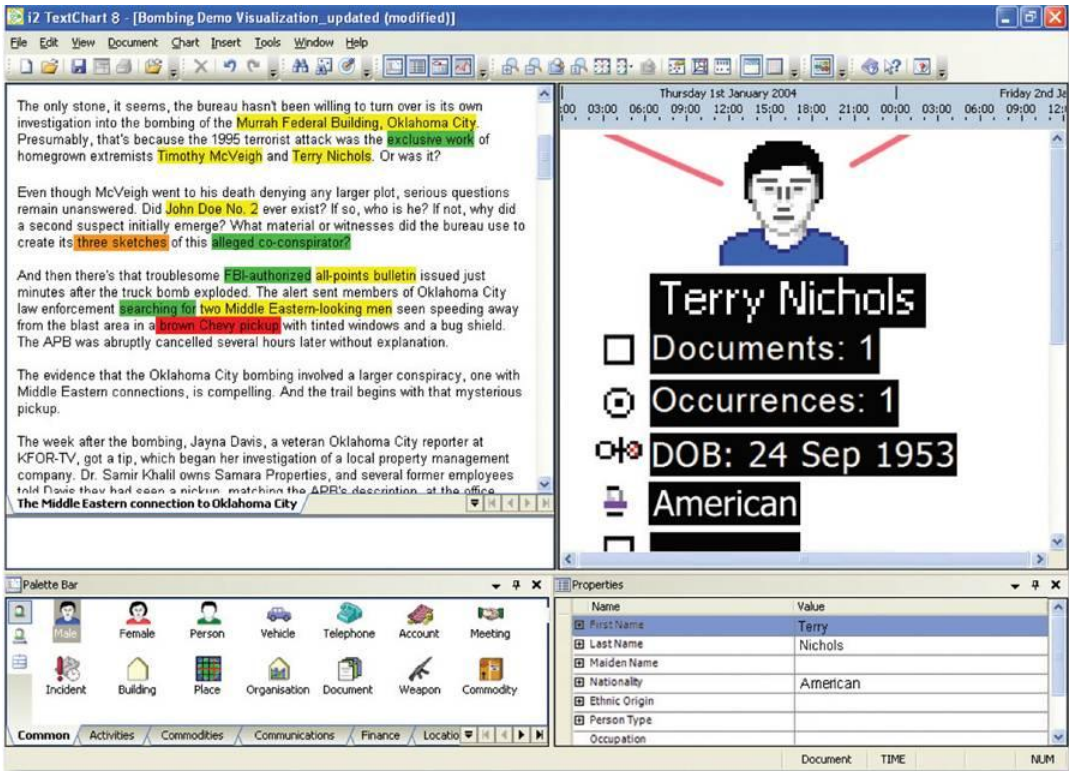

Figure 29. TextChart – text document window.

TextChart has a feature called "AutoMark." AutoMark provides named entity recognition (NER) as a backend service. NER software attempts to identify and categorize the people, places, and things within unstructured text documents. Humans do this very easily, but slowly. For software, this is a difficult task. Instead of the user manually reading the entire text document and identifying all the entities of interest, the NER software identifies the entities for the user; the user need only choose the entities of interest. This significantly speeds up the process of creating graphs. Since TextChart graphs can be imported into Analyst's Notebook (but not vice versa), TextChart is best used to construct an initial graph with analysis being performed later within Analyst's Notebook.

3.3.1.3 Google Trends

Google Trends (http://www.google.com/trends) provides a variety of useful information. For a given set of search terms, Google Trends shows: plots over time of Search Volume and News Reference Volume; links to significant news articles related to the topic(s); the top 10, the top 10

Cities, and the top 10 Languages being searched. Google Trends bases its results on data going back to January 2004. Results can be filtered by restricting the time frame (All Years, Last 30 Days, Last 12 Months, a particular year, or a particular month and year). Results can be further filtered by restricting the region (All Regions or any one of 237 countries). In addition, Google Trends results can be exported as a CSV file. See Figure 30 for an example of the results provided by Google Trends when the search term is "metamaterials."

3.3.1.4 Google Insights for Search

Google Insights for Search (http://www.google.com/insights/search/#) is similar to Google Trends, but it provides additional information and options. For a given set of search terms, Google Insights for Search shows: Categories (e.g., Science, Computers & Electronics, Industries, etc.); Interest Over Time (with a one year forecast); links to significant news articles related to the topic(s); top 10 Regional Interest; an animated world map showing the change of Regional Interest over time; the top 10 Searches; and the top 10 Rising Searches. Google Insights for Search bases its results going back to January 2004. Results can be filtered in a number of ways including: Compare by Search Terms, Locations, or Time Ranges; Web, Image, News, or Product Search; Worldwide or any one of 237 countries; 2004 to present, last 7 days, last 30 days, last 90 days, last year, or a specific date range; All Categories, any one of 27 major categories (e.g., Business, Computers & Electronics, Industries, Internet, News & Current Events, Photo & Video, Reference, Science, etc.), or any one of a large number of subcategories (e.g., for Science the subcategories include: Astronomy, Biological Sciences, Anatomy, Chemistry, Ecology, Geology, Mathematics, Physics, Scientific Equipment, and Scientific Institutions). In addition, Google Insights for Search results can be exported as a CSV file. See Figure 31 for an example of the results provided by Google Insights for Search when the search term is "metamaterials."

Figure 30. Google Trends – “metamaterials.”

Figure 31. Google Insights for Search – "metamaterials."

3.1.2 COTS Support Tools

We have found several COTS tools which provide tracking and/or reporting capabilities: Beyond Compare, Camtasia Studio, MindManager, MindView, and SnagIt. These tools are useful for managing data sets, capturing screen images and videos, and planning and recording technology assessments.

3.1.2.1 Beyond Compare

Beyond Compare from Scooter Software (http://www.scootersoftware.com) is a file and folder comparison utility. It allows the user to compare the content of files of many different file types as well as folder contents and hierarchies. Quick comparisons are based on file sizes and modification times. Thorough comparisons are based on byte-by-byte comparisons. Data files, executables, binary data, and images have dedicated viewers. Text files can be viewed with specific comparison rules for documents, source code, and HTML. Folders can be synchronized and files can be merged. Beyond Compare simplifies the management of the data files used in our technology assessments.

The Beyond Compare home view (see Figure 32) allows the user to choose what to view in addition to specifying options that control how files or folders are viewed.

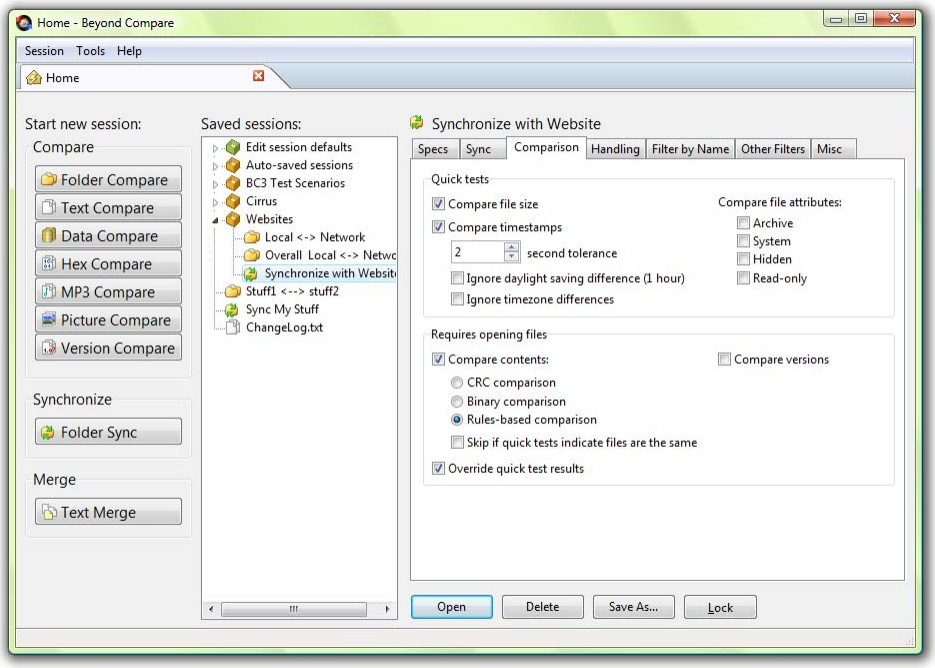

Figure 32. Beyond Compare – home view.

When viewing folders, the user may choose between multiple viewing options to immediately see all folder contents or the filtered folder contents (see Figure 33). For example, the user can choose to see: all files, only the files that are the same, only the files that are different, only the files that are newer in the left-hand folder, only the files that are newer in the right hand folder,

only the files that are in the left-hand folder but not in the right-hand folder (“orphans”), only the orphans that are in the right-hand folder but not in the left-hand folder, or combinations thereof.

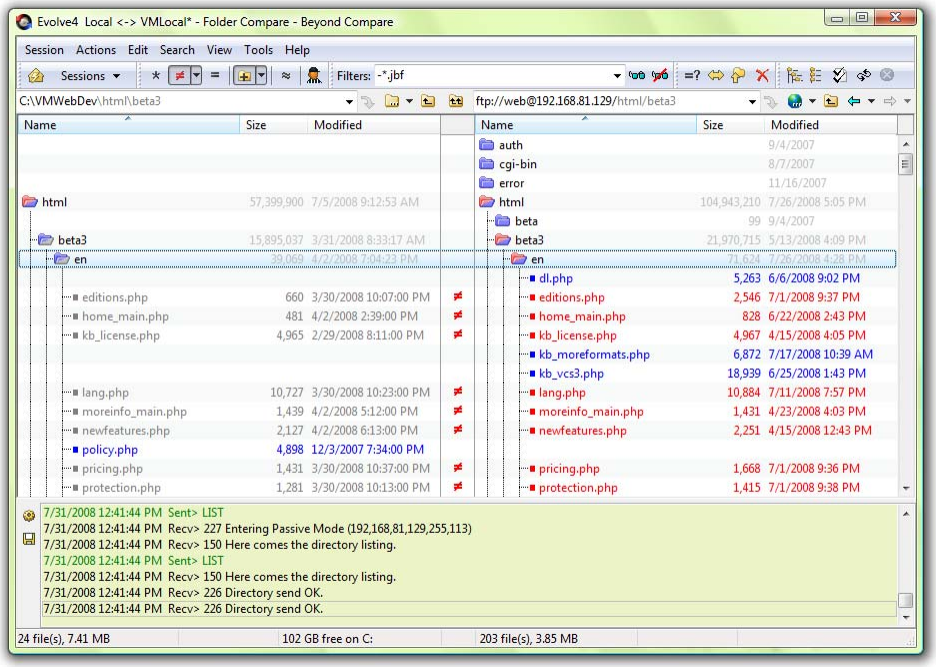

Figure 33. Beyond Compare – comparing folder contents.

When comparing text files, Beyond Compare will align the file contents line-by-line (see Figure 34). This is an incredibly useful option for anyone who has had to manage or merge multiple versions of a file. Beyond Compare goes beyond simple content comparison — it supports file editing — allowing the user to move or copy lines from one file to another, updating the display of file contents in real time. Color is used to distinguish file differences on a line-by-line basis.

Folders may be synchronized using Beyond Compare (see Figure 35). For example, a folder on your local hard drive can be kept in sync with a folder on a server. Beyond Compare makes the folder differences readily apparent and provides multiple options for bringing the folder contents into sync.

In addition to text files and folder contents, Beyond Compare has specialized viewers for binary files (see Figure 36), data files (see Figure 37), image files (see Figure 38), and others.

Sandia National Laboratories has a site license for Beyond Compare. Installation is simple and does not require Microsoft Windows Administrator rights.

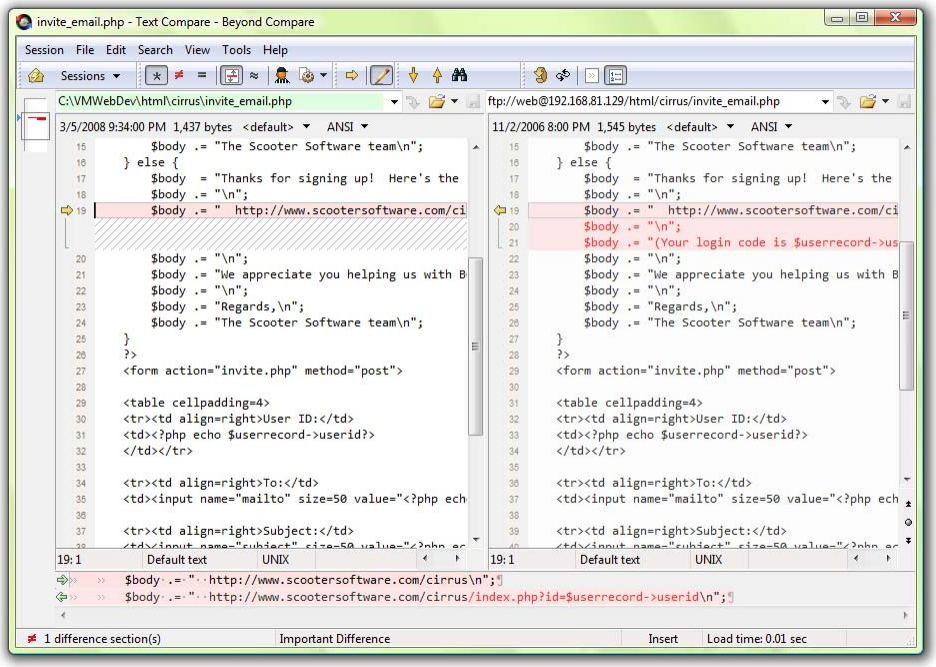

Figure 34. Beyond Compare – text file comparison.

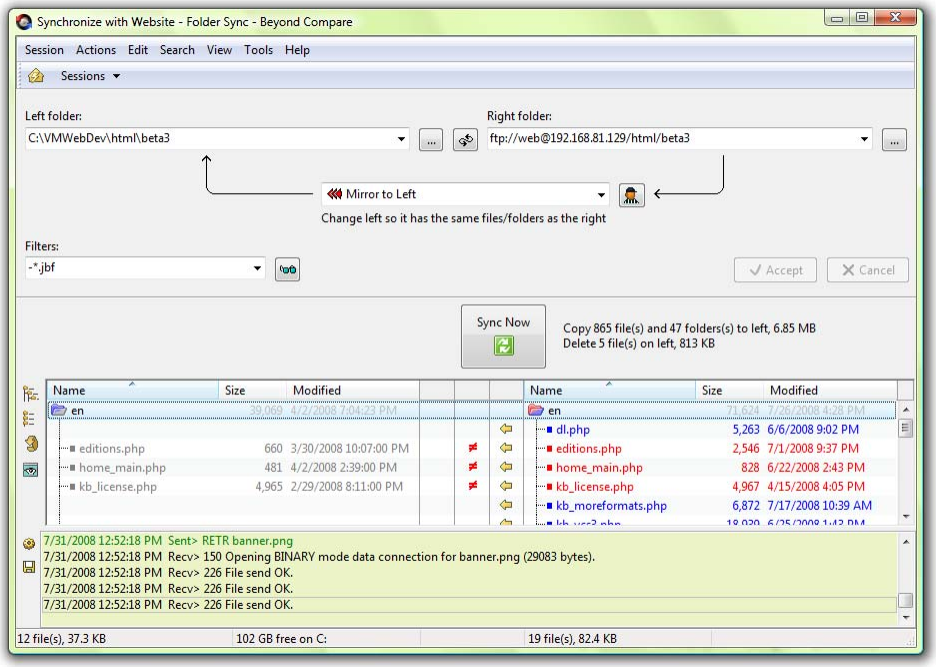

Figure 35. Beyond Compare – synchronizing folders.

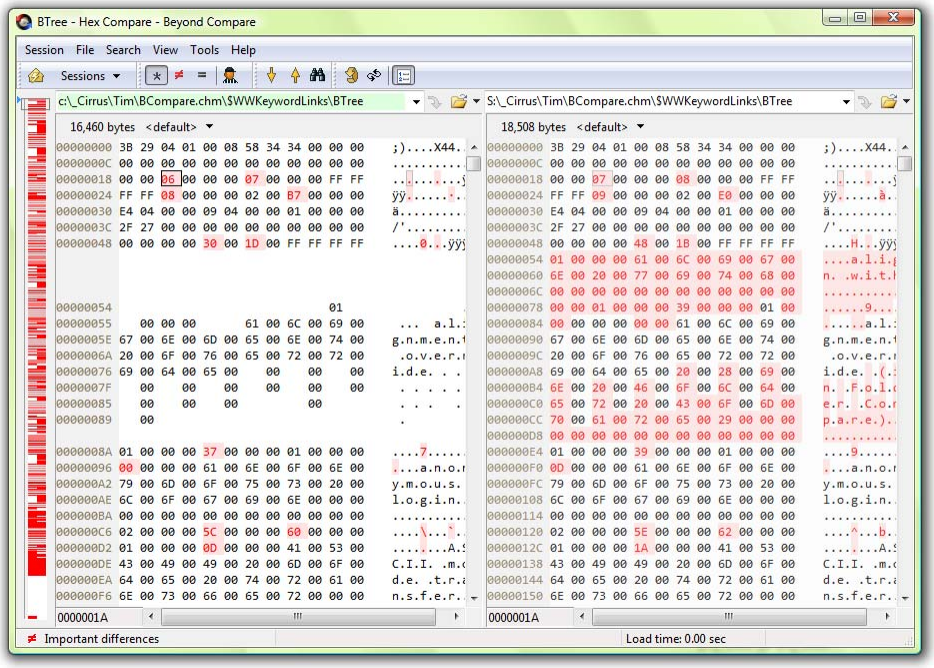

Figure 36. Beyond Compare – comparing binary files (the data is displayed in hexadecimal format).

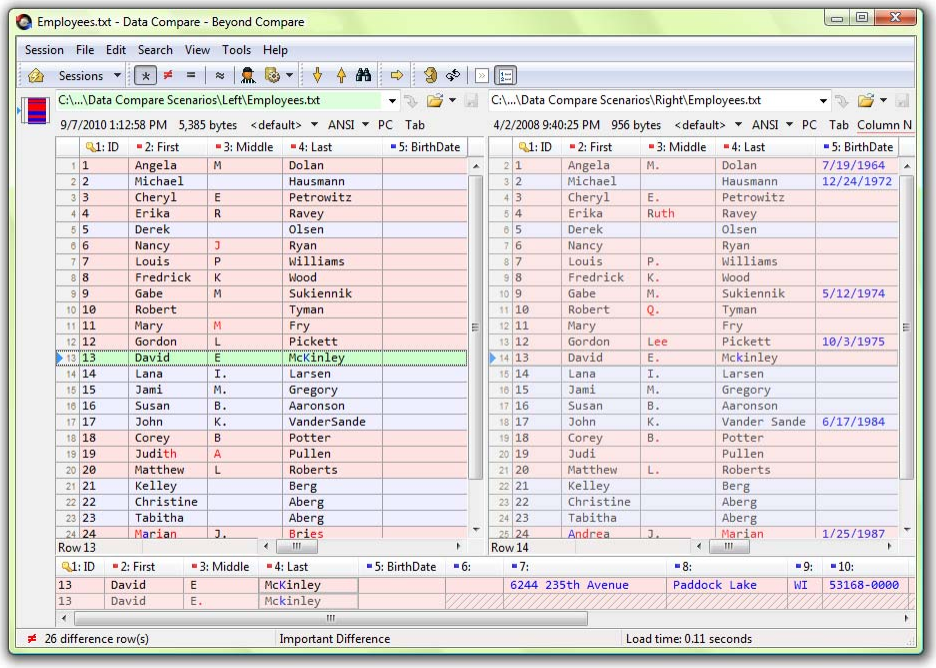

Figure 37. Beyond Compare – comparing data files.

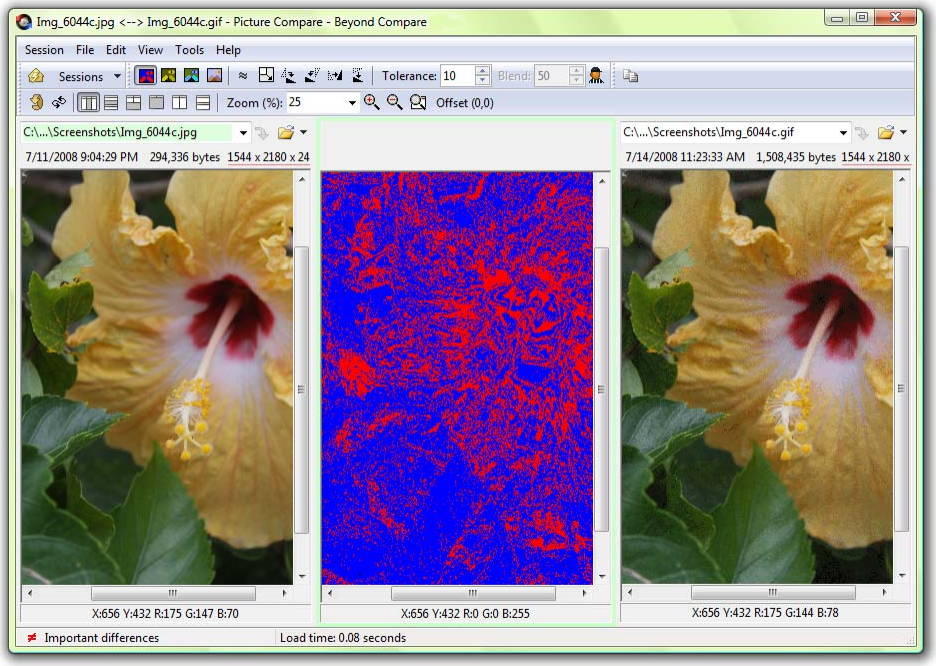

Figure 38. Beyond Compare – comparing image files.

3.1.2.2 Camtasia Studio

Camtasia Studio (http://www.techsmith.com/camtasia.asp) by TechSmith Corporation is a screen recorder and editor (see Figure 39). Camtasia Studio can capture the full screen, a window, or a region. It includes a Microsoft PowerPoint plug-in which allows presentations to be recorded. Camtasia Studio has TechSmith's exclusive SmartFocus technology, which tracks where the action happens during recording. During editing, the video can be zoomed in on the action. The editor allows the addition of callouts (to direct attention, link to a web page, or jump elsewhere in the video), cursor effects (to highlight cursor movements and mouse clicks, title slides, transitions, captions, quizzes, etc. It is also possible to add voice-over, music, animations, digital video clips, photos, etc. Camtasia Studio provides the ability to record videos of our technology assessments for use in reports and presentations and to create training materials.

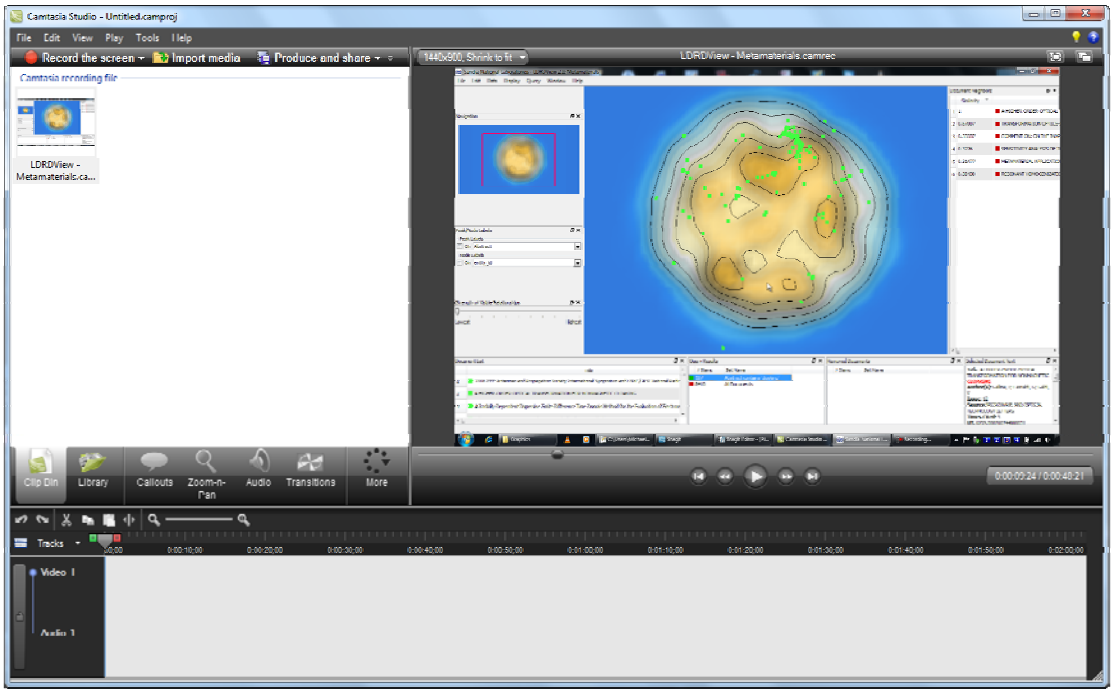

Figure 39. Camtasia Studio – edit window.

3.1.2.3 MindManager

MindManager (http://www.mindjet.com/products/mindmanager-9-win/overview) by Mindjet (see Figure 40) is a tool for generating visual information maps (mind maps). It provides a variety of maps (freeform, process, project, event, org chart, timeline, etc.) which can include hyperlinks, attachments, notes, images, etc. The map can be customized with icons, tags, shapes, callouts, colors, etc. Maps can be exported to Microsoft Word or Microsoft PowerPoint, as a PDF, as an image, or as a web page. MindManager is useful for brainstorming technology keywords and concepts, planning a technology assessment, and recording the flow of a technology assessment.

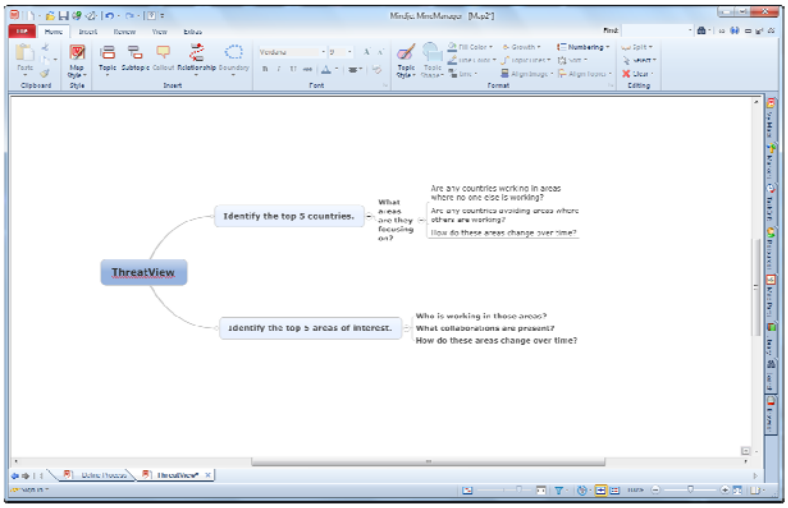

Figure 40. MindManager – main window.

3.1.2.4 MindView

MindView (http://www.matchware.com/en/products/mindview/default.htm) by MatchWare (see Figure 41) is a mind mapping tool which allows users to brainstorm and visualize ideas quickly and easily. It is fully integrated with Microsoft Office. It offers six map views (mind map, left right, top down, outline, timeline, and Gantt) which can include notes, attachments, hyperlinks, images, etc. Maps can be customized with icons, colors, images, etc. and exported in a variety of formats (Microsoft Word, Microsoft PowerPoint, Microsoft Excel, Microsoft Outlook, HTML, XML, or images). Like MindManager, MindView is useful for brainstorming technology keywords and concepts, planning a technology assessment, and recording the flow of a technology assessment.

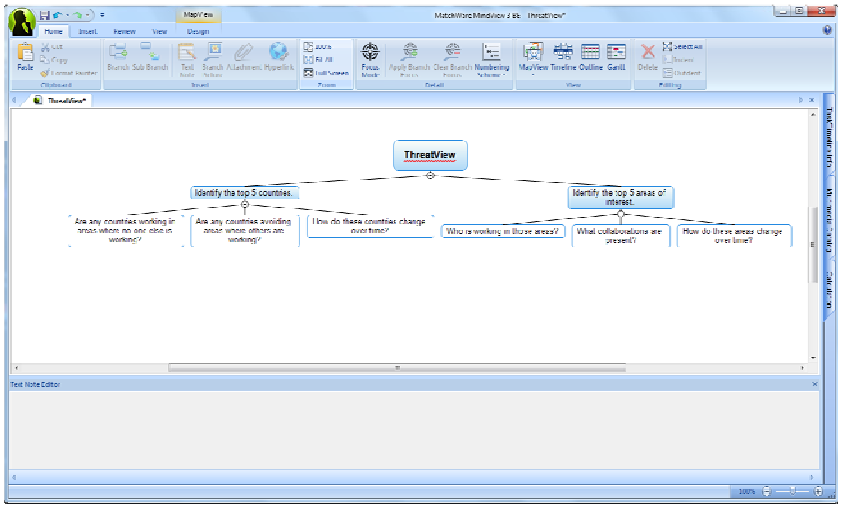

Figure 41. MindView – main window.

3.1.2.5 SnagIt

SnagIt (http://www.techsmith.com/screen-capture.asp) by TechSmith Corporation is a tool for generating screen captures (see Figure 42) and editing them (see Figure 43). SnagIt has many options for capturing all or part of the screen. The most useful capture options for our technology assessments are the window, scrolling window, and menu capture. The scrolling window option will automatically scroll the selected window and merge the multiple screen shots into a single continuous image. Screen captures can be output in a variety of graphic formats or exported directly to Microsoft Excel, Microsoft Word, and Microsoft PowerPoint. The editor allows images to be annotated, customized (resized, cropped, border and edge effects, etc.) and selectively blurred (to hide identifying information like accounts, etc.). SnagIt provides the ability to include relevant graphic images in our technology assessment reports and presentations.

Figure 42. SnagIt – main window.

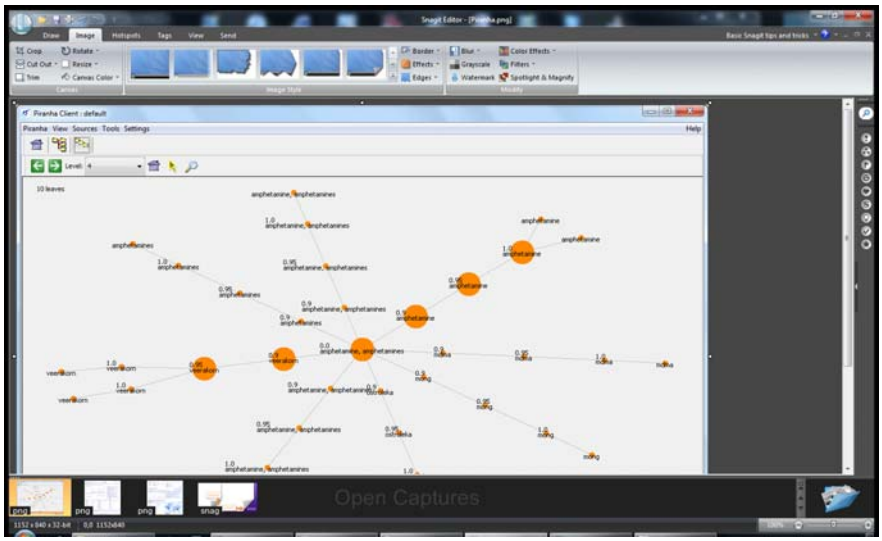

Figure 43. SnagIt – editor window.

3.4. Open Source Tools

There are several Open Source data mining tools available. We have found three such tools which are well suited to our toolkit: Gephi, KNIME, and ORA.

3.4.1 Gephi

Gephi (www.gephi.org) describes itself as "Like Photoshop but for data." It provides an interactive visualization and exploration platform focused on network graphs. It provides real-time visualization of networks with up to 50K nodes and 500K edges. Gephi includes an extensive set of metrics and algorithms for layout, dynamic network analysis, cartography, clustering and hierarchical graphs, and dynamic filtering. It is modular, well documented, and has a well-designed user interface. It also extensible though plug-ins.

3.4.2 KNIME

KNIME (www.knime.org), pronounced *naim*, is a data exploration package developed by the Chair for Bioinformatics and Information Mining at the University of Konstanz, Germany. It allows the user to create data flows (pipelines), selectively execute analysis steps, and explore the results interactively. KNIME provides over 400 processing nodes for data I/O (from both files and databases), data preprocessing and cleaning, data manipulation and statistics, modeling, analysis, and data mining (association rules, Bayes, clustering, rule induction, neural networks, decision trees, multidimensional scaling, principal component analysis, etc.). It includes a number of interactive visualizations (bar chart, histogram, pie chart, XY chart, box plot, parallel coordinates, scatter plot, etc.) and can be executed in parallel on multi-core systems. KNIME is modular, easily extensible, well documented, and has a well-designed user interface.

3.4.3 ORA

ORA (Organizational Risk Analysis) is a sophisticated graph analysis tool with many uses including social network analysis, geospatial network analysis, community finding, path finding and others. It was developed by the Computational Analysis of Social and Organizational Systems (CASOS) program at Carnegie-Mellon University (http://www.casos.cs.cmu.edu/). The software is free to download and use.

ORA uses the term meta-network to describe a grouping or collection of networks. A network is a set of relationships between people, places, or things. Networks and meta-networks can be represented in many different data formats; ORA has the ability to input and output these different data formats.

The ORA user interface is divided into three panels (see Figure 44). The left panel displays a list of the meta-networks that have been loaded into ORA. When a meta-network is selected, the right panel displays information about it and the bottom panel displays the results of any analytic processes run on it. The results shown in the bottom panel include network visualizations (see **Error! Reference source not found.**Figure 45).

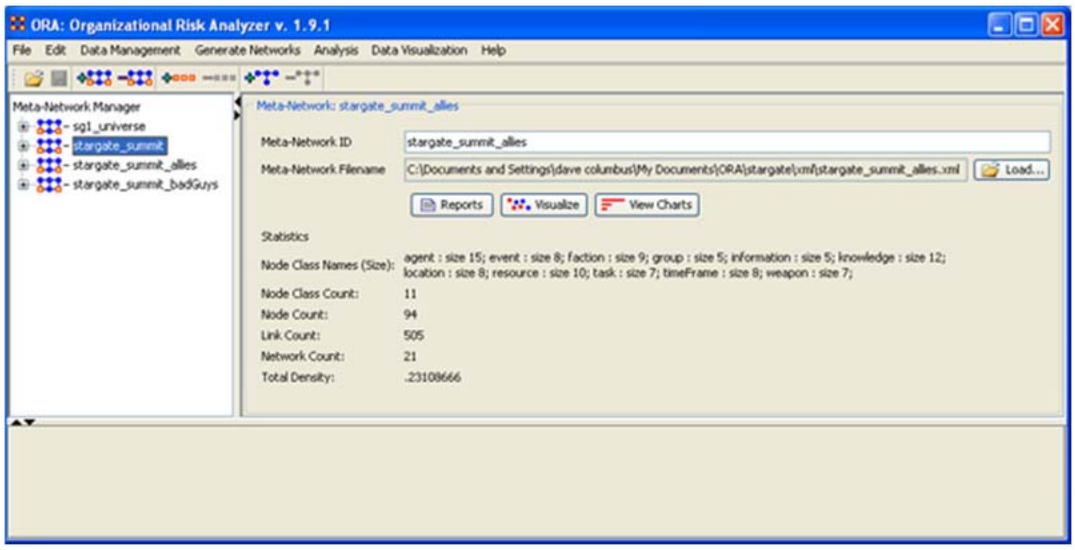

Figure 44. ORA – main window.

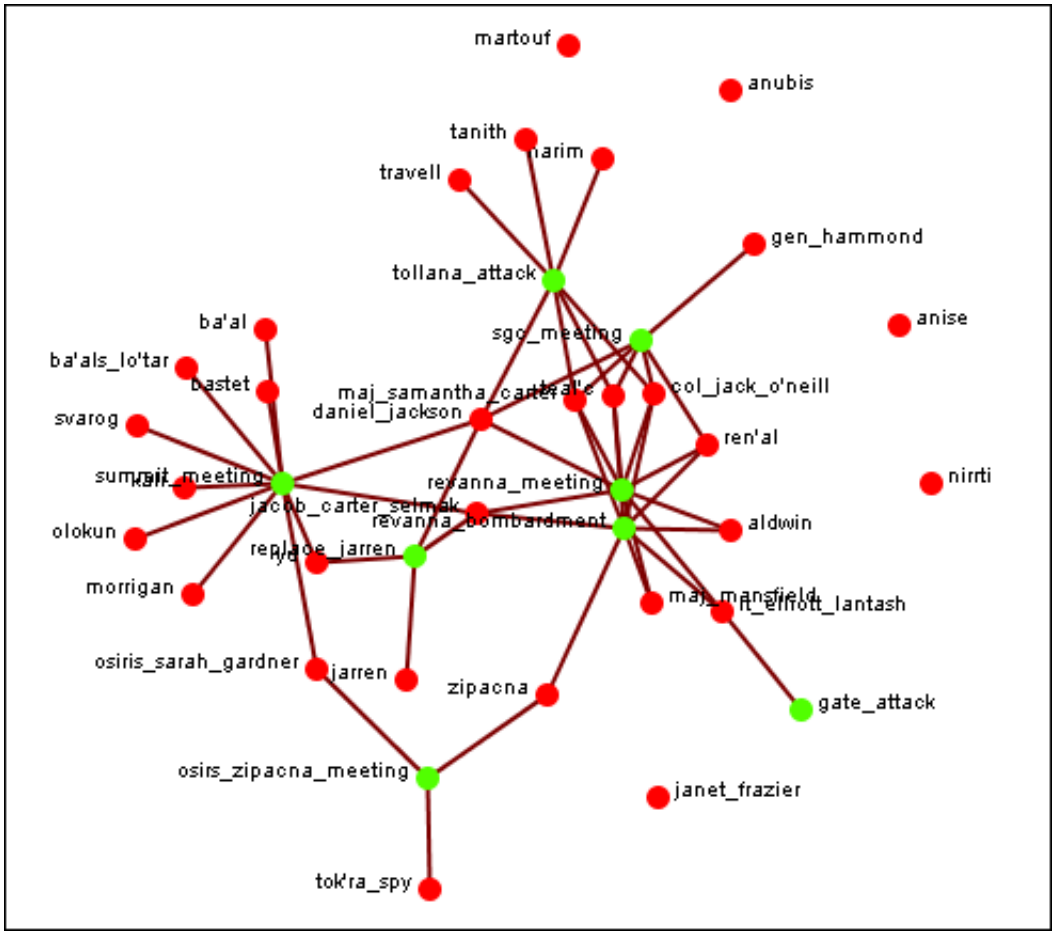

Figure 45. ORA – a network visualization.

In addition to visualizing a network, the user can analyze networks. For example, Figure 46 shows the results of running Newman's community finding algorithm on a network. Note how the communities in the resultant graph are color-coded. Some ORA analyses do not generate graphs, but instead generate numerical data (e.g., social network analysis returns summary statistics such as degree, betweeness and centrality).

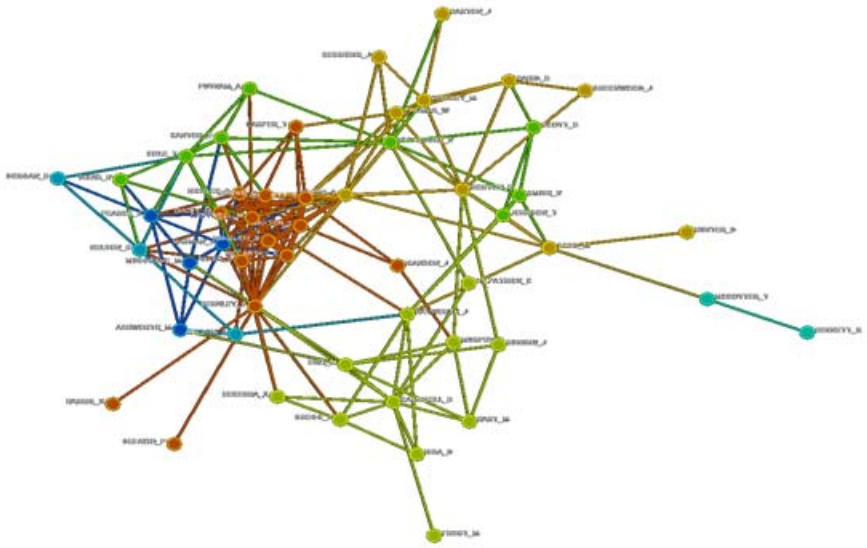

Figure 46. ORA – results of Newman's community finding algorithm.

ORA is a very capable tool that requires an investment in time on the part of the user, as evidenced by the 555 page user's guide. However, some useful analyses can be done without extensive investments of time or training.

4. FUTURE WORK

Our toolkit is being actively used to support the Science & Technology Intelligence (S&TI) program at Sandia. We have identified two major areas for future work.

First, we plan to incorporate new tools into our toolkit. We have acquired and are installing several tools from ESRI (Environmental Systems Research Institute, Inc.): ArcGIS Workstation, ArcGIS Extensions, ArcMap, ArcInfo, and Spatial Analyst. These tools will provide advanced geographic capabilities. We have also acquired and are installing two tools from SAS: Enterprise Miner and Text Miner. These tools will provide data mining with advanced linguistic capabilities. In addition, we plan to investigate Starlight (Future Point Systems/PNNL), CPLEX (IBM/OPL Studio), and THEMAT (Los Alamos National Laboratories).

Second, we plan to extend the existing Sandia-developed tools and to develop new tools. We plan to extend the Data Trace Tool and Cognitive Spider to better support the S&TI programmatic requirements. We plan to extend ThreatView to support alternative algorithms (non- LDA) and visualizations (non-landscape) and to parallelize it for improved performance. We plan to develop new tools to run on Sandia's HPC (High Performance Computing) resources. This will allow us to analyze and visualize larger data sets much faster than current tools.

5. CONCLUSIONS

The main goal of our two-year LDRD was to develop a toolkit for detecting indicators of technical surprise in textual data sets. We have succeeded in this effort. Our toolkit incorporates tools developed at Sandia National Laboratories and Oak Ridge National Laboratory, by COTS (Commercial Off The Shelf) vendors, and by Open Source developers. We are no longer limited to structured text; we can also analyze and visualize unstructured text. We are no longer limited to a single way to analyze and visualize data.

Although ThreatView remains a key tool in our toolkit, it is no longer our only tool. Our ability to perform technology assessments in support of the Science & Technology Intelligence (S&TI) effort is enhanced by the nineteen tools in our toolkit. These tools allow us to acquire data sets (Data Trace Tool and Cognitive Spider), to mine and visualize data sets in multiple formats (ThreatView/LDRDView, P2, Piranha, dtSearch, Analyst's Notebook, Text Chart, Google Trends, Google Insights for Search, Gephi, KNIME, ORA), and to record our process and results (Beyond Compare, Camtasia Studio, MindManager, MindView, and SnagIt).

We failed to find a Swiss Army Knife for our toolkit: no single tool does everything. By using multiple tools, each of which analyzes and visualizes the data sets in a different way, we are able to provide timely, relevant technology assessments with a high degree of confidence in our results.

6. REFERENCES

1. Kohonen, Teuvo. *Self-Organizing Maps*. 3. Springer-Verlag, 2001. Print.

2. Lensu, Anssi. "Similar Document Detection using Self-organizing Maps." Knowledge-Based Intelligent Information Engineering Systems, 1999. Third International Conference (1999): 174-177.

3. Roussinov, Dmitri G. "A Scalable Self-organizing Map Algorithm for Textual Classification: A Neural Network Approach to Thesaurus Generation." Communication and Cognition in Artificial Intelligence Journal 15(1998): 81-111.

4. Schatzmann, J. *Using Self-Organizing Maps to Visualise Clusters and Trends in Multidimensional Datasets* BEng thesis, Imperial College, June 19, 2003.

5. Vesanto, Juha. "Clustering of the Self-Organizing Map." IEEE TRANSACTIONS ON NEURAL NETWORKS VOL. 11, NO. 3(2000): 586-600.

6. Zhang, Jin. *Visualization for Information Retrieval*. Springer-Verlag, 2008. Print.

DISTRIBUTION

1	MS0359	D. Chavez, LDRD Office	1911
1	MS0549	R. Neiser, Jr.	5918
1	MS0621	A. Shrouf	5631
1	MS0899	Technical Library	9536 (electronic)
1	MS1188	E. Parker	6354
1	MS1188	J. Wagner	1432
1	MS1204	F. Mendenhall	5932
1	MS1207	M. Rightley	1207
1	MS1207	M. Trahan	5928
1	MS1209	G. Laughlin	5920
1	MS1209	N. Rahal	5925
1	MS1217	R. Contreras	5925
1	MS1218	M. Foehse	5925


Sandia National Laboratories