Transformation Algorithms for Data Streams[1] [2]

Stephen G. Eick
SSS Research, Inc.
650 Warrenville Road,
Suite 100A
Lisle, IL 60532
eick@sss-research.com

John W. Lockwood, Ron Loui, James Moscola,
Chip Kastner, Andrew Levine, Mike Attig,
Applied Research Laboratory
Washington University in St Louis
1 Brookings Drive, Campus Box 1045,
St. Louis, MO 63130
lockwood@arl.wustl.edu

Doyle J. Weishar
SAIC Advanced Systems & Concepts Division
3811 N. Fairfax Drive Suite 850
Arlington, VA 22203
weishard@saic.com

Abstract—Next generation data processing systems must deal with very high data ingest rates and massive volumes of data. Such conditions are typically encountered in the Intelligence Community (IC) where analysts must search through huge volumes of data in order to gather evidence to support or refute their hypotheses. Their effort is made all the more difficult given that the data appears as unstructured text that is written in multiple languages using characters that have different encodings. Human Analysts have not been able to keep pace with reading the data and a large amount of data is discarded even though it might contain key information. The goal of our project is to assess the feasibility of incrementally replacing humans with automation in key areas of information processing. These areas include document ingest, content categorization, language translation, and context-and-temporally-based information retrieval.

Mathematical transformation algorithms, when implemented in rapidly reconfigurable hardware, offer the potential to continuously (re)process and (re)interpret extremely high volumes of multi-lingual, unstructured text data. These technologies can automatically elicit the semantics of streaming input data, organize the data by concept (regardless of language), and associate related concepts in order to parameterize models. To test that hypothesis, we are building an experimentation testbed that enables the *rapid implementation of semantic processing algorithms in hardware*. The system includes a high-performance *infrastructure* that includes hardware-a accelerated content processing platform; mass storage to hold training data, test data, and experiment scenarios; and tools for analysis and visualization of the data.

In our first use of the testbed, we performed an experiment where we implemented three transformation algorithms using FPX hardware platforms to perform semantic processing on document streams. Our platform uses Field-programmable Port Extender (FPX) modules developed at Washington University in Saint Louis [3].

This paper describes our approach to building the experimental hardware platform components, discusses the major features of the circuit designs, overviews our first experiment, and offers a detailed description of the results, which are promising.

TABLE OF CONTENTS

1. INTRODUCTION

In the Intelligence Community (IC) analysts must search through huge volumes of data in order to gather evidence to support or refute their hypotheses. Their effort is made all the more difficult given that the data appears as unstructured text that is written in multiple languages using characters that have different encodings. Data processing systems must deal with massive volumes of data and ingest content at very high rates. The problem is that existing approaches to analyze and process data have not keep pace with the increasing volumes of data. Large volumes of data are discarded even though they might contain useful information. We explored a new approach for analyzing and organizing intelligence data that provides for categorizing and translating the content of high speed data streams.

[1] 0-7803-8870-4/05/$20.00© 2005 IEEE
[2] IEEEAC paper #1633, Version 4, Updated December 5, 2004

Our approach uses *mathematical transformation algorithms* implemented in reconfigurable hardware to continuously (re)process and (re)interpret high volumes of multi-lingual unstructured text data. The system can automatically elicit the semantics of streaming input data, organize the data by concept (regardless of language), and associate concepts with similar concepts needed to parameterize text processing models. To evaluate the potential of this system, we are building an experimental testbed that enables *rapid implementation of data processing algorithms in hardware*. The system provides a high-performance *infrastructure* consisting of a hardware-accelerated content processing platform; mass storage device that holds training data and experimental scenarios; and tools for analysis and visualization of the data.

In our first use of the testbed, we performed an experiment where we implemented circuits to perform three transformation algorithms in hardware. Together, they provided statistical analysis on the semantic content in document streams flowing through a network. Our platform uses the Field-programmable Port Extender (FPX) modules developed at Washington University in Saint Louis. Each FPX contains two large FPGAs – called the Reconfigurable Application Device (RAD) and the other called the Network Interface Device (NID) [3]. Multiple RAD circuits implemented the semantic processing circuits to perform the "bag of words" style text analysis. NID circuits were used to route data through the system at a bandwidth of 2.4 Gigabit/second. We used this platform to implement our transforms and performed an experiment. The first part of this paper describes our approach to building the experimental hardware platform components including the major aspects of the circuit design and integration.

This remainder of this paper is devoted to an overview of our first experiment and a detailed description of the results, which are promising. We tested the performance of our transforms and hardware platform by analyzing postings to 12 Google groups. The postings were divided into seven known categories, four unknown categories, and a large "chaff" category that we treated as noise. We then compared the transforms on their ability to discover known categories when trained and on their ability to discover unknown categories without training in the presence of high and low noise levels. Our results are promising. For the known categories, all of the transforms were successful in identifying and organizing documents into known categories. For the unknown categories, two of the transforms algorithms successfully identified some unknown but pure categories. This second case is particularly important as tests the ability of our transforms self-organize historical data when new issues arise.

2. A New Approach to Streaming Data Processing

Innovative **mathematical transformation algorithms** implemented in software have shown promise for automatically understanding the content of documents [1, 2, 3, 4, 5, 6]. Similarities between content from multiple sources can be computed, but computation can be very expensive -- $O(n^3)$ in some cases, where n is the number of documents that contain information. The high computational time limits the amount of information that can be processed. To understand social indicators which predict hostility and conflict before it happens, massive amounts of data must be ingested in order to gain a ground truth and to determine how the information is relevant in multiple contexts.

Development of a testbed that can perform transform algorithms in high-speed, reconfigurable hardware enables several key innovations. First, it allows for the real-time management and processing of information. Live data can be processed in real-time as it arrives. Data can also be re-processed as new contexts emerge. Two methods for processing data flows are described below.

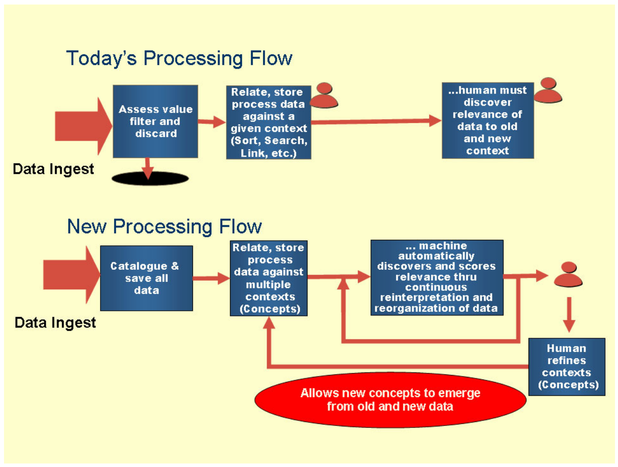
Today's Processing Flow
Data Ingest
Assess value filter and discard
Relate, store process data against a given context (Sort, Search, Link, etc.)
...human must discover relevance of data to old and new context
New Processing Flow
Data Ingest
Catalogue & save all data
Relate, store process data against multiple contexts (Concepts)
... machine automatically discovers and scores relevance thru continuous reinterpretation and reorganization of data
Human refines contexts (Concepts)
Allows new concepts to emerge from old and new data

Figure 1. A New Approach to Information Management

"Track before Detect" Information Management

Today's processing flow involves the detection of a context, then the tracking of information that relates to that concept. Specifically, the "detect and track" scheme can be described as:

1. Filter all information being gathered and save only the information that is germane to the immediate problem.
2. Relate atomic facts in the data, including the names of people, places, things, and the relationships between them.
3. Process the results with queries built to uncover "gold nuggets" of useful information.

4. Track and report changes in the status of the information.

As shown in Figure 1, the new processing flow **inverts** the mechanism used by today's flow. Using the new flow, the process becomes:

1. Catalog and save all information being gathered based upon atomic facts in the data.
2. Populate many parallel hypotheses (concepts) with the facts and data.
3. Use a machine to detect when a hypothesis has sufficient supporting facts (relevance) to warrant evaluation and reinterpretation of data.
4. Aggregate evidence for the hypotheses, and report any changes that emerge.
5. Adapt and refine concepts when indicated that a concept is either too broad or too specific.

Future information processing systems must provide a large amount of cognitive support to the analyst about all the data that has been used in all on-going or past analyses. In other words, the system needs to be able to keep track of the original data, the relationships of the data to other information, and the reason why it was considered important at the time. Additionally, since no analyst could possibly have "seen" *all* the data collected by the system, the system also needs to supply the means by which analysts can search for related information and discover new associations and patterns in the vast amounts of data.

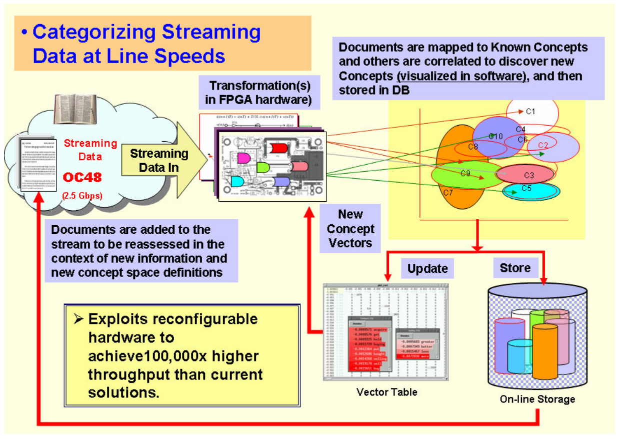
• Categorizing Streaming Data at Line Speeds
Documents are mapped to Known Concepts and others are correlated to discover new Concepts (visualized in software), and then stored in DB
Transformation(s) in FPGA hardware)
Streaming Data
OC48
(2.5 Gbps)
Streaming Data In
C1
C10
C4
C6
C2
C8
C9
C3
C5
C7
Documents are added to the stream to be reassessed in the context of new information and new concept space definitions
New Concept Vectors
Update
Store
Exploits reconfigurable hardware to achieve100,000x higher throughput than current solutions.
Vector Table
On-line Storage

Figure 2. Real-time concept-based streaming data processing

A "*concept-based*" storage and retrieval approach enables "track before detect" by reducing the intellectual impedance mismatch between analysts and machines. Developing special-purpose computing machines to continuously (re)process and (re)interpret extremely large volumes of unstructured multi-lingual text data makes this possible. And, the ability to process massive amounts of information quickly is the basis of our second innovation:

Stream data in *real-time* into machines that *self-organize concepts* from input semantics

The basis for this innovation is that as data are streamed into the system, they are transformed by high-speed semantic processing circuits into points in multi-dimensional space, as shown in Figure 2 (above). Documents containing similar information – that is, those that relate to similar concepts, will tend to cluster into similar regions of the multi-dimensional space. A cluster region, defined as a set of points within a set distance from the centroid of the cluster, defines a concept. This concept, in turn, is used as a basis to store and receive documents. Clustering self-organizes the data.

By attaching the concept storage space to a distributed data network, automated servers can rapidly retrieve data that had been collected from multiple sources and score how their position in the multiple-dimensional concept space relates to a new hypothesis. Scoring circuits, also implemented as high-speed computing machines in reconfigurable hardware, can rapidly scan through vast amounts of data to determine what information is relevant to a new concept.

The methodology described above allows us to not only organize data by concept, but also to continuously look through data and associate it with existing or newly formed hypotheses. Since hypotheses can be thought of as being comprised of combinations of concepts, we can realize yet another innovation. That is:

<u>Continuously reprocess</u>* and reinterpret data to update and score in real-time *<u>thousands of hypotheses</u>

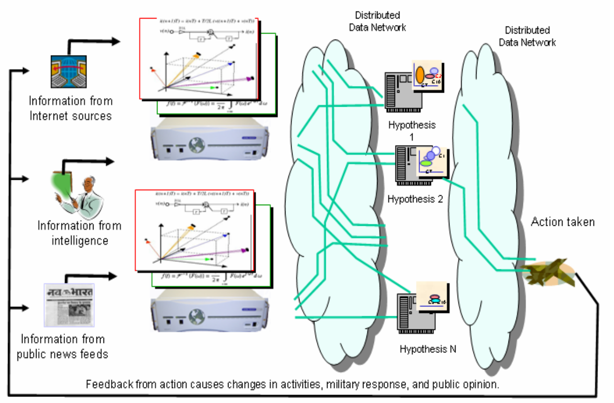
Information from Internet sources
Information from intelligence
Information from public news feeds
Distributed Data Network
Distributed Data Network
Hypothesis 1
Hypothesis 2
Hypothesis N
Action taken
Feedback from action causes changes in activities, military response, and public opinion.

Figure 3. Real-time hypothesis processing

This final innovation will assist analysts to obtain evidence of when social indicators emerge. These data can come from various sources that include the Internet, intelligence gathering agencies, and public news feeds, as shown in Figure 3. The text is instantly processed for semantic meaning, related contents are grouped together, and multiple hypothesis servers process the data in parallel to find new concepts and topics that emerge. Because data are constantly being processed, hypothesis servers continually aggregate weak information in support for, or rejection of, both new and ongoing concepts. Once a hypothesis gains sufficient support, action can be taken.

3. TESTBED EXPERIMENTS

The overall approach for conducting experiments is shown in Figure 4. A blend of high-speed network devices and reconfigurable hardware is used to rapidly ingest and process data, while software is used to control and manage clusters. Data is received over a network as text or HTML documents carried over standard Transmission Control Protocol / Internet Protocol (TCP/IP) packets. A TCP processor decodes the packets that contain the document in one or more TCP/IP input flows. Every word (baseword) in the document is analyzed for its semantic meaning. All of the words in each document are then counted to determine their frequencies of occurrence. A document vector is then generated which characterizes the content of the document. This document vector is then scored against a set of vectors that represent known or emerging concepts. Thresholds are used to determine if content can be classified as an existing concept or if a new cluster should be formed.

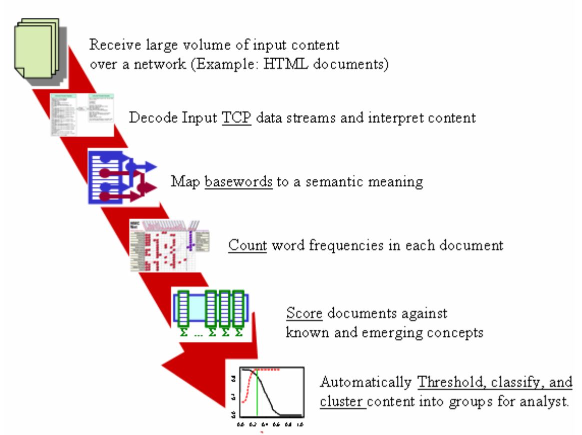
Receive large volume of input content over a network (Example: HTML documents)
Decode Input TCP data streams and interpret content
Map basewords to a semantic meaning
Count word frequencies in each document
Score documents against known and emerging concepts
Automatically Threshold, classify, and cluster content into groups for analyst.

Figure 4. Testbed Enables Real-Time, Semantic Classification of Content

The testbed enables computationally intensive semantic processing functions to be performed in real-time. Field Programmable Gate Arrays (FPGAs) are used to perform hardware-accelerated processing of the data at all of the levels described above. By using FPGAs, all parts of the system can be dynamically reconfigured to perform new algorithms for data processing, content classification, and/or concept clustering. Massive volumes of real data can be streamed through the system. Measurements can be made of the system's precision, recall, throughput, and latency.

The testbed provides a modular environment where different hardware and software components can be used to process data. New software can be used by downloading programs into the compute servers in the testbed. New hardware circuits can be tested by dynamically reconfiguring one or more of the FPGAs in the testbed. A variety of different mechanisms, implemented in software and/or hardware, can be used to scan, process, count, score, and cluster documents. Well-defined XML interfaces provide a common interface between computational modules so that most of the infrastructure can be reused in different experiments.

4. DESIGN OF THE EXPERIMENT HARDWARE PLATFORM

Our first experimental hardware platform has been prototyped and uses reconfigurable hardware to rapidly process content in FPGA hardware. This system uses Field-programmable Port Extender (FPX) modules developed at Washington University in Saint Louis, to perform several layers of data processing functions in hardware. Multiple FPX modules have been integrated into a Global Velocity GVS-1000 chassis. A photograph of an FPX module and the GVS-1000 chassis is shown below.

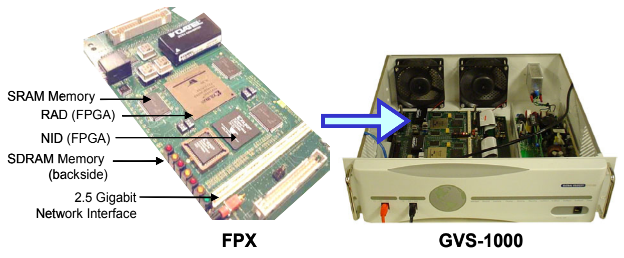

Figure 5. Field-Programmable Port Extender (FPX) Modules Mount in GVS-1000 System below Gigabit Ethernet or OC48 Line Cards

Each FPX contains two large FPGAs – called the Reconfigurable Application Device (RAD) and the Network Interface Device (NID). In addition to the FPGAs, each FPX cards includes two parallel banks of SRAM and two parallel banks of SDRAM. Reconfigurable hardware modules are deployed using logic within the vast resources of a Xilinx Virtex XCV2000E FPGA circuit that implement the RAD.

The RAD circuits on the FPX have been used to implement the *TCP processor, Baseword module, Count module, Score module,* and a *Report* module. All of these circuits have all been implemented as modular hardware components that make use of parallel finite state machines seas of combinatorial logic. High-speed network interfaces allow the FPX to communicate other modules in the system and with other hardware and software outside of the system using standard Internet Protocol (IP) datagrams. The detailed configuration of a GVS1000 system that performs semantic processing of TCP/IP traffic passing over a network is shown in Figure 6. For this circuit, five FPX cards were operated in parallel in order to process content quickly.

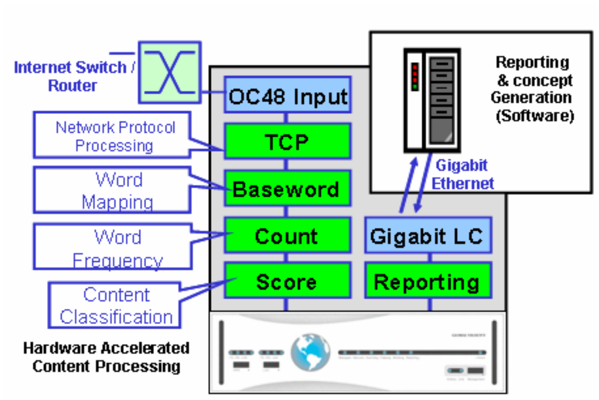

Figure 6. Stacks of FPX Modules fit into a GVS-1000

The FPX cards have connectors that allow them to be stacked on top of each other. By stacking cards, data can be processed as streams that flow through multiple modules. Each module transforms the data in a way that the output from one card provides the input for next card below. A photograph that shows how five FPX cards were stacked to implement the semantic processing system is shown in Figure 7.

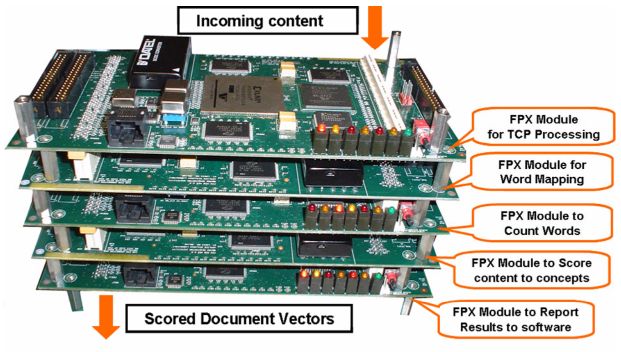
Incoming content
FPX Module for TCP Processing
FPX Module for Word Mapping
FPX Module to Count Words
FPX Module to Score content to concepts
FPX Module to Report Results to software
Scored Document Vectors

Figure 7. FPX Modules stacked to process stream with multiple modules

TCP/IP Processing

Over 85% of all traffic on the Internet today uses the TCP/IP protocol. TCP is a stream-oriented protocol providing guaranteed delivery and ordered byte flow services. A vast number of end systems communicate over the Internet. Traffic is concentrated to flow over a relatively small number of routers which forward traffic through the core of the Internet. Currently, Internet backbones operate over communication links ranging in speed from OC-3 (155 Mbps) to OC-768 (40 Gbps) rates.

Processing of TCP data flows at a location in the middle of the network is generally considered to be extremely difficult. Along the way, packets can be dropped, duplicated and re-ordered. Packet sequences observed within the interior of the network can be different from packets received and processed at the connection endpoints [9].

The TCP processor which is used in this testbed enables complex network services to operate at gigabit speeds by processing TCP stream data directly in hardware. The TCP processor tracks up to 8 million bidirectional TCP flows on an OC-48 (2.5 Gbps) network link. Network data packets are annotated with additional control signals which provide information about which data bytes correspond to the IP header, the TCP header, and the TCP payload section. There are also signals that indicate which TCP data bytes are parts of a consistent stream of data and which bytes should be ignored because they are retransmissions. Signals which indicate the Start of Flow (SOF) and End of Flow (EOF) are included along with a unique flow identifier so that the client can independently manage per-flow context [9].

Word Mapping

The current hardware design provides for the representation of documents and document-prototypes as high-dimension feature vectors. A Word Mapping Table (WMT) is used to map input from a universe of 1 million words into one of 4000 dimensions. The WMT maps the hash of an input word to one of 1 million entries in a lookup table. Each entry in the lookup table can then remap the word into one of the 4000 dimensions based on the semantics of the word. Three approaches have thus far been explored to perform automatic, semi-automatic, and automatic selection of features used to create the WMT.

Dictionary-based Word Mapping

In the first approach, knowledge about the language itself was used to populate the WMT. Words in a dictionary were first assigned to a large number of dimensions, and then dimensions were collapsed wherever a synonymous relationship was noted. The dictionary of words was generated using a hybrid of tools, scripts, and (in some cases) human input. Automated scripts were developed that applied stemming rules and searched words for common prefix and suffix extensions to quickly populate entries in the WMT with groups of words that had similar meaning.

In one experiment, over 27,000 words, including proper nouns, were collapsed to 6,414 dimensions. Further reduction was used to map the table into 4000 dimensions. Words which were not in the dictionary were hashed to any one of the 4000 dimensions.

To understand the desired content of the Word Mapping Table as shown in Figure 8 below, consider how two dimensions in the table could be populated with words that represent the meanings of *explosives* and *rockets*. There are several specific names for explosives, such as “nitroglycerine”, “gelamex”, “dynamite”, and their equivalents in Arabic, Greek, and other languages of interest. Each of these input words will have a hash applied that maps it to any one of a million locations. For example, given some hash function, H, it would be possible that H(“nitroglycerine”)=101,203 and that H(“gelamex”)=672,101. Each of the possible resulting hash values is used to index an entry in the Word Mapping Table. The million-entry WMT is then populated with pointers that remap the input word to a common baseword. For example, if the baseword for explosive has been mapped to dimension number 1033, then:

WMT(H(“nitroglycerine”))=WMT(101,203)=1033, and
WMT(H(“gelamex”))=WMT(672,101)=1033

Likewise, if rockets are mapped to dimension 2801, then:

WMT(H("Nassar"))=2801, and
WMT(H(“Qassam”))=2801

Thus processing of different words with similar meanings is made possible. A graphical view of the function of the word mapping table is shown below. The Word Mapping Table itself is described to the system in a file with a XML format.

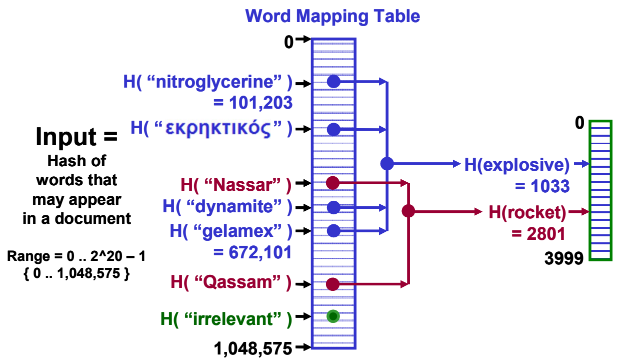

Figure 8. Word Mapping Table (WMT) Allows Hash of Each of 1 Million Input Words to be Mapped into any Output Value in a 4000-Dimensional Vector

Word Mapping with Pair-wise Differentiation
The second method, explored by Fair Isaac, starts with an *m-document* training set partitioned into *k* known categories. The most powerful discriminators for pairwise differentiation are used to generate data in the known categories.

This method bypassed entirely the question of what words were in the lexicon and what words were not. The hardware design represented all words using a million-wide hash. Each of the million buckets is mapped to one of the unit vectors in the 4000-dimensional space. This mapping could be specified by the feature-selection algorithm or be reprogrammed as needed.

There are downsides to the algorithm. Even with one million unique words, is not possible to encode all of the unique four-letter sequences that can occur with a 26-letter alphabet. If a lexicon were to have few collisions due to nuisance strings, or quasi-words, extracted from binary documents (binary noise), significant collisions could occur. The mapping from the million-wide representation to the 4000-dimensional representation would preserve any ambiguity about what string actually hashed to the original million-wide bucket.

To avoid populating the table with quasi-words that have little value, computation is performed to determine the actual words that contribute most to the classification and clustering decisions. This computation requires significant computation and the algorithm’s running time is inherently nondeterministic. An unfortunate side effect of this optimization is that most words unseen in the training set are not assigned representation in the 4000-dimension feature vector for a document. Thus, clustering and category drift could only be based on those hash values that had proved useful in separating the training set's categories.

Word Mapping with Information Retrieval
The third method was based on ideas from information retrieval (IR). Documents in each training set category were studied for the frequency of occurrence of words. The most used words, as defined as those that occur more than 0.1% of the time, are excluded from the table because they typically represent common nouns, verbs, and prepositions that add little meaning to the document. Excluding those, the next most frequent 100 and next most frequent 500 words are assigned to dimensions in the WMT.

Use of duplicated dimensions is avoided for the most frequent 100 words. Likewise, use of duplicate dimensions is minimized for the next most frequent 500 frequent words. 2000 of the dimensions are reserved for words that were not seen in the training set (unknown terms). A background inverse term frequency (ITF) and inverse document frequency are used (IDF) for weighting dimensions in dot products between document vectors.

Implementation of the Word Mapping Circuit
A block diagram of the word mapping circuit, as it was implemented in FPGA logic is shown below. The circuit reads data input from the TCP/IP wrappers. The word parser includes multiple modules to process text in different languages. With English ASCII text, for example, words are processed as groups of 8-bit characters separated by white space. With Arabic and Greek text, however, the circuit uses modules that process 16 bits at a time to decode sequences of UTF-16 characters. For languages that have a notion of upper and lower case characters, the circuit can normalize the case so that both variations of the same word are treated the

same. The policy can be altered by reconfiguration of the FPGA hardware.

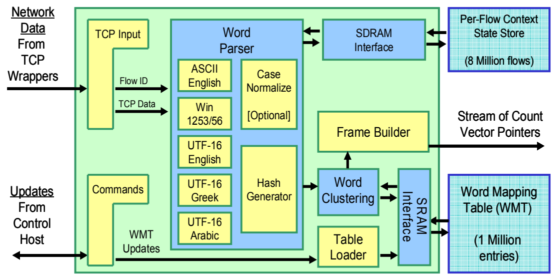

Figure 9. Block diagram of the word mapping circuit, as it was implemented in FGPA logic

Once a word has been identified, a hash is computed using the *word clustering* module. This module accesses the WMT, which itself is implemented in off-chip Static Random Access Memory (SRAM). The values in the WMT each return a value in the range of {0..3999} and are used to identify the root meaning of the word.

It is not always the case that an input packet will contain an entire word. It is possible that a word will be split between packets when the data stream is segmented for transmission over a TCP/IP network. The circuit supports identification of strings that cross packet boundaries. Synchronous Dynamic Random Access Memory (SDRAM) is used to store the state of each traffic flow's context. Words segmented across multiple, interleaved packets can still be identified in a TCP/IP steam by tracking the last K bytes of each stream in the large, off-chip SDRAM. The existing system tracks the last K=32 bytes of data per stream.

Word Frequency

Word frequencies are counted for every active flow in the system for each dimension. For each traffic flow, the count circuit computes the sum of the basewords that occur in each dimension of the document vector. To support 4000-dimensional vectors that are used by the baseword circuit, the count circuit maintains the state of 4000 parallel counters.

A challenge for determining the word frequency for data that appears on a TCP/IP network is that packets from different flows can be interleaved as they pass through the network. Count arrays resulting from flows that are interleaved will produce count array packets that are interleaved. Data can appear when it arrives to the count circuit from a TCP/IP network. Interleaved packets generate interleaved basewords. Thus, basewords resulting from the content in the flows can be interleaved.

As with the baseword circuit, the count circuit uses off-chip SDRAM to maintain multiple contexts that track the state of each flow. The count circuit supports 524,288 (512k) flows that each requires storage of 4000 counters. Each counter is represented with a 4-bit (1/2-byte) value. A total of 512K*2K = 1 Gbyte of memory is used to store the state of all flows.

Scoring

Once a flow has ended, the resulting *document vector* is compared against a set of *concept vectors*. In general, the FPGA could be programmed to compute a score using any desired Support Vector Machine (SVM) function. For the circuit implemented in the existing testbed, a dot product is computed of the *document vector* against a set stored *concept vectors.*

Coefficients can be dynamically loaded into the score table. This *score table* can be formatted in one of two ways. One FPGA circuit was implemented that supports 4-bit coefficients for a table of 30 concepts. The other FPGA circuit that was implemented supports 8-bit coefficients for a table that supports 15 concepts. Both circuits operate on the 4000-dimension vector.

A high throughput of processing is achieved because the computation of the multiplication occurs in parallel for each concept. Count values that define the incoming *document vector* arrive at a rate of 8 elements per clock cycle. For the scoring circuit that supports 30 concepts, the system performs 8*30=240 parallel multiplications per clock cycle. Over the duration of the 4000-element vector, the FPGA performs a total of 4000*30=120,000 multiplications per document. Given that the system can score 150,000 documents per second, the parallel hardware performs 120,000*150,000 = 18 Billion multiplications per second.

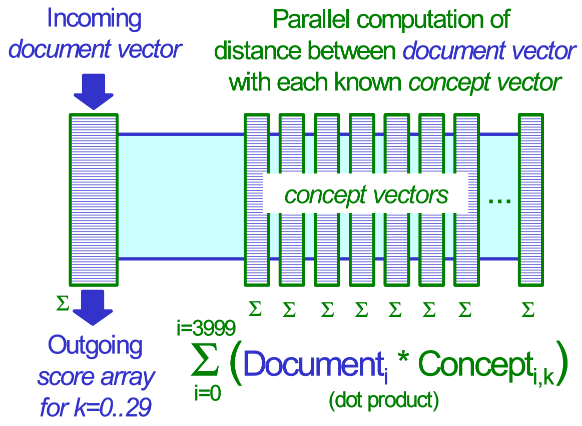
Incoming
document vector
Parallel computation of
distance between document vector
with each known concept vector
concept vectors
...
Σ
Σ Σ Σ Σ Σ Σ Σ Σ
Outgoing
score array
for k=0..29
$\sum_{i=0}^{i=3999} (\text{Document}_i * \text{Concept}_{i,k})$
(dot product)

Figure 10. Hardware circuits accelerate the classification of documents by computing the least-squares distance (dot product) between an input vector and representative content. At times of peak processing, the circuit performs 18 Billion multiplications per second.

5. Description of the Experiment

A set of newsgroups were selected for analysis. Scripts were written to automatically download content from Google's newsgroup area. A corpus was collected that included 2000 messages from the Internet newsgroups:

- alt.sports.baseball.stl_cardinals
- comp.ai.neural_nets
- comp.programming.threads
- humanities.musics.composers.wagner
- misc.consumers.frugal_living
- misc.writing.moderated
- rec.equestrian
- rec.martial_arts.moderated
- sci.archeology.moderated
- sci.logic
- soc.libraries.talk

The data was human analyzed for content and off-topic messages were removed. Headers to the files were stripped so that the machine learning process would not have access to the newsgroup identifier.

The training sets were generated through random selection to avoid training only on data that happened to consist of a few large threads. Chaff was downloaded from another newsgroup *talk.origins*. That newsgroup has a large amount of off-topic content and flame messages and threads that were largely non-cohesive.

Tests were run on the corpus with different amounts of training data and chaff. The first set of experiments used no chaff and a training set which consisted of {1%, 33%, and 50%} of the files. The second set of tests added random data so that 10% of the total files were chaff. The experiment was again run using {1%, 33%, and 50%} of the data for training. The third set of tests added more random data so that that 90% of the files in the corpus were chaff. The experiment was run again using {33% and 50%} of the data for training

Mathematical Transformation Algorithms
In the experiment, we evaluated the three mathematical transform algorithms in terms of their ability to detect known categories when provided with training data on these categories and on their ability to discover unknown categories without training. The first case is intended to model the situation where the transform organizes material related to a known context. In many ways it parallels the traditional form of analysis where information related to a known concept is tracked and organized. The second case investigates if transforms are able to self organize and detect new concepts without the benefit of training material.

The three algorithms we investigated were:

- K-Means, a standard statistical clustering technique.
- AGS –Associated Grounded Semantics as described in the related paper by John Byrnes of Fair Isaac that appears in this conference.
- Order – ratio of top two order statistics for the document score vector

Due to the proprietary nature of these algorithms, we are unable to provide detailed descriptions of them in this paper.

Two Parts of the Experiment
There were to parts of the experiment. In the part each of the algorithms was trained using sample data from seven groups of the eleven groups. The algorithms were then applied to the remainder the corpus that contained postings from the seven selected groups and posting from the four other groups. Then experimental runs were done where the amount of "chaff," e.g. postings from *talk.origins*, was varied from 0% to 90%. The optimal performance for these runs would be if the transform correctly labeled every posting from the seven groups with the appropriate label and labeled posting from the five groups that it not trained on as "chaff."

The second part of the experiment was intended to test the transform's ability to self organize. Without the benefit of any training data, the transforms were compared on their ability to discover clusters in the corpus. The ideal performance in this case would be if the transform identified 12 cluster corresponding to the 12 groups.

6. Analysis of Results

We analyzed the results from the first part of the experiment in two ways. First, we compared the three transform algorithms using a confusion matrix. This visual display, showing in Figure 11, compares the three algorithms on their ability to classify posting transforms. The figure shows the results for the run with 90% chaff where 33% of the corpus was used in the training set. This is the most difficult case. Figure 11 shows all three of the confusion matrices. The ideal performance would be if each matrix were diagonal for the first seven categories corresponding to the training data and if all of the documents for the remaining five groups assigned to the last category. The AGS algorithm approaches ideal performance. It correctly classifies essentially all of the postings for the groups that it was trained on and correctly assigns most of the other postings to the chaff group. Its precision was 95% meaning that it out of 20 documents assigned to a category 19 were in the correct category and recall rate was 62% meaning that it found 62% of the labeled documents.

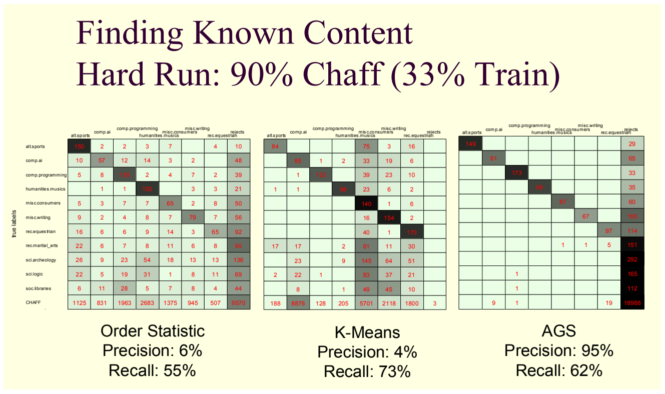

Figure 11 Detecting Known Content.

Figure 12 shows the ROC curves for the AGS run described above. ROC curves show the ratio type 1 to type 2 errors. These ROC curves are properly shaped, indicated that the algorithm worked remarkably well.

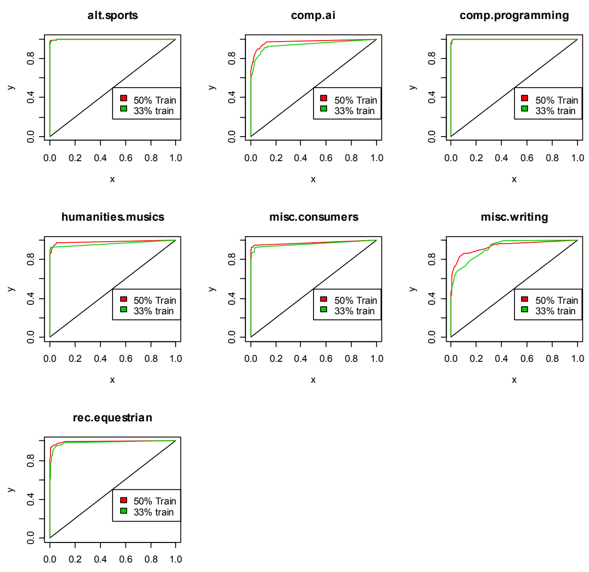

Figure 12 ROC Curves for AGS.

For the second part of the experiment that involved the ability to self-organize without out training, we use an extension of the confusion matrix to show the performance of the algorithms. Figure 13 shows the results from a sample run where the algorithm identified thirty clusters. Each of the thirty clusters is shown in descending order as the rows in the matrix display. The columns show the number of documents from that particular group in the cluster and the horizontal bar charts along the right show the same information graphically. The bars in the bar chart are stacked and color encoded to show the number of postings from each of the 12 groups. Thus an ideal cluster would consist of documents from a single group and would be represented as a "pure" color in the bar chart. As the figure shows, the algorithm was only somewhat successful at discovering clusters. It discovered several clusters that were cleanly from a single category but also identified several other clusters that were composed of a mixture of documents from various groups.

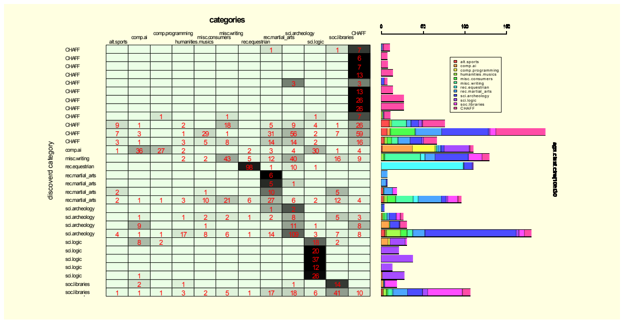

Figure 13 Detecting Unknown Content.

7. Discussion and Summary

We have described a fundamentally new approach for analyzing massive amounts of data in heterogeneous information streams that that has the potential to overcome many of the problems with current approaches. The new idea, enabled by hardware-accelerated computational processing engines, is to process and re-process real-time streams of information that have been transformed into self-organized concepts. To explore this idea we have developed an experimental testbed that is performs semantic computations on documents at very high rates. We applied our testbed to analyze postings to twelve Google groups using chaff and various subsets of our corpus as training data. We explored three different mathematical transformation algorithms.

Our results are extremely promising. When training data was available, the algorithms successfully classified postings to the correct Google groups with remarkable precision. Without training data, the algorithms successfully identified some, but not all, of the clusters. Our results support overall hypothesis that self-organizing transformation have potential to enable a totally new and fundamentally better approach for analyzing information streams.

8. References

[1] Cristianini, N. and Shawe-Taylor, J., "An Introduction to Support Vector Machines and other kernel-base learning methods", Cambridge University Press, 2000.

[2] Domingos, P., "Mining High-Speed Data Streams", *Sixth ACM SIGKDD international conference on Knowledge Discovery and Data mining (KDD),* Boston, MA, Aug. 20-23, 2000, pp. 71-80.

[3] Lockwood, J., "Evolvable Internet Hardware Platforms", *NASA/DoD Workshop on Evolvable Hardware (EHW'01)*, Long Beach, CA, July 12-14, 2001, pp. 271-279.

[4] Hulten, G., Spencer, L. and Domingos, P., "Mining

Time-Changing Data Streams", *Sixth ACM SIGKDD international conference on Knowledge Discovery and Data mining (KDD),* San Francisco, CA, Aug. 26-29, 2001, pp. 97-106.

[5] Vigna, G., Robertson, W., Kher, V. and Kemmerer R., "A Stateful Intrusion Detection System for World-Wide Web Servers", *Proceedings of the Annual Computer Security Applications Conference (ACSAC 2003),* pages 34–43, Las Vegas, NV, December 2003.

[6] Schuehler, D. and Lockwood, J., "A Modular System for FPGA-based TCP Flow Processing in High-Speed Networks", by David Schuehler, John Lockwood*; 14th International Conference on Field Programmable Logic and Applications (FPL),* Springer LNCS 3203, Antwerp, Belgium, August 2004, pp. 301-310.

[7] Madhusudan, B. and Lockwood, J., "Design of a System for Real-Time Worm Detection", by *12th Annual Proceedings of IEEE Hot Interconnects (HotI-12);* Stanford, CA, August, 2004, pp. 77-83.

[8] Dharmapurikar, S., Krishnamurthy, P., Sproull, T., and Lockwood, J., "Deep Packet Inspection using Parallel Bloom Filters", *IEEE Micro*, Vol. 24, No. 1, Jan 2004, pp. 52-61.

[9] Schuehler, D., Moscola, J., and Lockwood, J., "Architecture for a Hardware-Based, TCP/IP Content-Processing System", *IEEE Micro*, Vol. 24, No. 1, Jan 2004, pp. 62-69.

[10] Wang, Y., Hodges, J. and Tang, B., "Classification of Web Documents Using a Naïve Bayes Method", *IEEE International Conference on Tools with Artificial Intelligence*, Sacramento, CA, USA, November 3-5, 2003, pp 560.

[11] Fall, C. J. and Benzineb K., "Literature survey: Issues to be considered in the automatic classification of patents", World Intellectual Property Organization, Oct. 29, 2002.

9. BIOGRAPHIES

Stephen G. Eick*, a fellow of the American Statistical Association, has received 26 patents, and has won many awards for his technology including the Bell Lab's President's award and the 2000 Computer-world Smithsonian award for key technologies that change the way people live and work. His educational background includes a B.A from Kalamazoo College (1980), M.A. from the University of Wisconsin at Madison (1981), and his Ph.D. in Statistics from the University of Minnesota (1985). Eick's research focuses on information visualization: building rich visual interfaces to help users understand complex information sets. Eick is particularly interested in visualizing network information and in visualizations that are related to data mining. In his experience the most interesting visualizations are motivated by real problems.*

***John W. Lockwood** designs and implements networking systems in reconfigurable hardware. Lockwood and his research group developed the Field programmable Port Extender (FPX) to enable rapid prototype of extensible network modules in Field Programmable Gate Array (FPGA) technology. He is an assistant professor in the Department of Computer Science and Engineering at Washington University in Saint Louis. He has published over 50 papers in journals and technical conferences that describe technologies for providing extensible network services in wireless LANs and in high-speed networks. Professor Lockwood has served as the principle investigator on grants from the National Science Foundation, Xilinx, Altera, Nortel Networks, Rockwell Collins, and Boeing. He has worked in industry for AT&T Bell Laboratories, IBM, Science Applications International Corporation (SAIC), and the National Center for Supercomputing Applications (NCSA). He served as a co-founder of Global Velocity, a networking startup company focused on high-speed data security. Dr. Lockwood earned his MS, BS, and PhD degrees from the Department of Electrical and Computer Engineering at the University of Illinois. He is a member of IEEE, ACM, Tau Beta Pi, and Eta Kappa Nu.*

***Ron Loui** is an Associate Professor in Computer Science and Engineering. He is the author of over seventy articles in leading technical journals over the past two decades including AI Journal, Cognitive Science, Computational Intelligence, Journal of Philosophy, Journal of Symbolic Logic, MIT Encyclopedia on Cognitive Science, AI and Law, Theory and Decision, CACM, and ACM Computing Surveys. He was a Stanford Sloan Fellow in 1988 and has performed research for Digital Equipment and Rockwell, and has consulted for Xerox and McDonnel-Douglas. Professor Loui received his undergraduate degree at Harvard with high honors in Applied Mathematics: Decision and Control, 1982. He received a joint Computer Science and Philosophy doctoral degree from the University of Rochester, after a CS MS, in 1987. He is a graduate of Punahou School in Honolulu, Hawaii.*

***James Moscola, Chip Kastner, Andrew Levine, Mike Attig** are graduate students that work at Washington University in St. Louis.*

***Doyle Weishar** is the Vice President of Advanced Technology in the Advanced Systems and Concepts Division of SAIC Corporation. He has over 20 years of experience in the advanced research and development of strategic information systems. A former DARPA Program Manager, Dr. Weishar holds an M.S. in Computer Science and a Ph.D. Information Technology.*