Hardware Accelerated Algorithms for Semantic Processing of Document Streams,
by John W. Lockwood, Stephen G. Eick, Justin Mauger, John Byrnes, Ron Loui, Andrew Levine,
Doyle J. Weishar, Alan Ratner, IEEE Aerospace Conference, Big Sky, MT, March 4-11, 2006, Paper 10.0802

Hardware Accelerated Algorithms for Semantic Processing of Document Streams

Stephen G. Eick
SSS Research, Inc.
600 South Washington
Suite 100
Naperville, IL 60540
eick@sss-research.com

John W. Lockwood,
Ron Loui,
Andrew Levine
Washington University in St Louis
1 Brookings Drive,
Campus Box 1045,
St. Louis, MO 63130
lockwood@arl.wustl.edu

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

Alan Ratner
National Security Agency
9800 Savage Road
Fort Meade, MD 20755-6158
asratne@nsa.gov

John Byrnes
HNC Software, LLC
3661 Valley Centre Drive
San Diego, CA 92130
johnbyrnes@fairisaac.com

[1]*Abstract—There is a need within the intelligence communities to analyze massive streams of multilingual unstructured data. Mathematical transformation algorithms have proven effective at interpreting multilingual, unstructured data, but high computational requirements of such algorithms prevent their widespread use. The rate of computation can be vastly increased with Field Programmable Gate Array (FPGA) hardware.*

To experiment with this approach, we developed a system with FPGAs that ingests content over a network at high data rates. The system extracts basewords, counts words, scores documents, and discovers concepts on data that are carried in TCP/IP network flows as packets over a Gigabit Ethernet link or in cells transported over an OC48 link. These algorithms, as implemented in FPGA hardware, introduce certain constraints on the complexity and richness of the semantic processing algorithms.

To understand the implications of these constraints and to benchmark the performance of the system, we have performed a series of experiments processing multilingual documents. In these experiments, we compare techniques to generate basewords for our semantic concepts, score documents, and discover concepts across a variety of processing operational scenarios.

TABLE OF CONTENTS

[1] This research was sponsored by the Air Force Research Laboratory, Air Force Materiel Command, USAF, under Contract number MDA972-03-9-0001. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFRL or the U.S. Government.

1. INTRODUCTION

Within the intelligence community, there is a need to search through massive amounts of multilingual documents that are encoded using different character sets. It has been shown that computational linguistics and text processing approaches are useful in sorting through information sets, extracting relevant documents, and discovering new concepts. The problem remains that the computational complexity of the current generation of algorithms is such that document ingest rate is insufficient to keep up with the high rate of information flow.

To overcome this problem we have developed a system that uses FPGAs to accelerate concept discovery and classification algorithms. Circuits have been implemented as reconfigurable hardware modules that dramatically increase data ingest rates. Many types of data processing algorithms are well suited for acceleration in hardware, but some are not. Design optimizations were made for the implementation of the semantic processing hardware modules that determine how many words the algorithm considers, how hash collisions will occur, and how text in different character sets is processed.

Some text analysis algorithms are better suited for FPGA acceleration than others. Algorithms that perform "bag of words" processing, for example, are widely used and appropriate for many types of computational linguistics tasks. Our 2004 efforts centered on implementing one such algorithm with FPGAs [1]. To investigate the utility of hardware accelerated text analysis algorithms, we have developed a reconfigurable FPGA-based semantic processing system and a text analysis testbed. We have used our testbed to experiment with a variety of target problems involving concept classification, concept discovery, and language identification.

In the remainder of this paper we will describe our semantic processing system (Section 2), discuss design considerations (Section 3), review our algorithms (Section 4), describe the text analysis testbed (Section 5), display some of our experimental results (Section 8), and summarize (Section 9). The experiments involve language identification as well as classification of concepts and concept discovery within one language and across

multiple languages. To perform our test we downloaded postings from Google newsgroups and various forums. We also analyzed CMU's 20 Newsgroups corpus. Some results are quite promising. For language identification, our system has proven to be 99% accurate as well as extremely effective at concept classification over these corpora.

2. FPGA-BASED SEMANTIC PROCESSING

We have built a system with reconfigurable hardware that enables the real-time analysis of multilingual data transmitted over a high-speed network. The system elicits the semantics of unstructured text that appears in documents passing through the network by performing hardware accelerated Latent Semantic Indexing (LSI) on the content of each network traffic flow.

The system performs the data processing functions using reconfigurable hardware. Specifically, each content processing module is implemented using Field Programmable Gate Array (FPGA) logic and tables stored in a combination of on-chip and off-chip memory [1].

The function of each module is determined by bitfiles that configure FPGAs on the Field-programmable Port Extender (FPX) platform [2]. Data is processed in a pipeline as it streams through a stack of FPX modules in a GVS-1000 chassis. [3].

Our system first reads Internet Protocol (IP) packets from a Gigabit Ethernet or OC-48 line card. The first module in the system reconstructs the original sequences of bytes sent over each Transmission Control Protocol (TCP) flow [4]. Additional modules, like HAIL, have also been developed to determine the language and encoding used to transmit text in each TCP/IP flow by performing an n-gram analysis of extracted bytes [5].

3. HARDWARE AND SOFTWARE DESIGN CONSIDERATIONS

Our approach to accelerating algorithms using FPGAs involves the mapping of computations onto FPGA circuits which can perform the data processing at multi-Gigabit/second line rates. For semantic processing, we have found that there are three primary compute-bound operations. These operations are: (1) tokenization and calculation of a document vector; (2) mapping of a document vector to its associated concept; and (3) scoring of document vectors against concept vectors to identify the nearest match. Each of these operations is well-suited for acceleration and parallelization using FPGAs.

The steps required to develop encodings for FPGA circuits are different than those needed to develop software code. The resulting hardware circuits can be blindingly fast when the algorithm space is constrained in certain ways. However, these constraints impose limits on how our semantic processing algorithms operate. This section will describe the flow of data through the hardware and also demonstrate the cooperative tradeoffs between hardware and software, and discuss how we have overcome these limitations.

3.1 Relationship to Latent Semantic Indexing (LSI)

LSI uses a feature vector to represent a document. In our system, a document consists of the textual data that appears within a TCP/IP flow. We have implemented circuits that reconstruct the text from within TCP/IP flows and compute a document vector that describes the occurrence of words within the data. This document vector is scored against multiple concept vectors to determine which topic the text most closely relates to. Scoring can be performed in a number of ways. For each underlying method, every word can be weighted to determine its importance to a particular concept.

The hardware data processing is performed using a series of hardware modules. The baseword module determines (tokenizes) words from a flow and passes the individual words to the count module. The count module takes the individual feature counts and builds the feature vector for a flow. When the flow ends, the feature vector is passed to the score module where it is scored against a set of known concept vectors.

3.1.1 Tokenizing and Calculating Baseword Vector

Our baseword module determines the set of words in a flow. A word, or token, is a sequence of bytes followed by one or more separating characters. In English, the most common separating characters are spaces. Our tokenizer parses text in multiple character encodings to allow for processing of foreign language documents.

The determination of a word is performed though a circuit that has two primary functions. First, the circuit translates characters to normalize the case. Lower case is translated to upper case. Multiple types of whitespace characters are translated into a unified space character. Second, the circuit implements multiple computational engines that tokenize the resulting byte stream using multiple encodings. Each engine is programmed with a range of bytes that is considered acceptable for a particular language encoding. Numerous character set encodings are used on the Internet today, including ASCII, UTF-16, UTF-8, ISO 8859 pages and Windows code pages. Some of these encodings overlap. For the ASCII engine, the valid byte range is 0x41 to 0x5a and 0x61 to 0x7a. This corresponds to 'A' to 'Z' and 'a' to 'z'. Characters outside the acceptable byte ranges are considered whitespace delimiters and will stop words from building. Words are assumed to be strings of characters that have at least three valid characters. As an example of how the engines operate on words, consider the following sequence:

the quick brown fox

The ASCII engine will output the following:

THE QUICK BROWN FOX

This same set of words would result from an input sequence that has the following combination of letters, symbols, and numbers:

9_the_88+quick3brown<>fox

The word-building engines find valid runs of acceptable bytes. This is the first step in determining important features for a vector. Our system processes words that have a length ranging from 3 to 16 characters. Byte sequences longer than 16 characters are truncated to a word length of 16. For example, the word "antidisestablishmentarianism" would be treated as "ANTIDISESTABLISH" and the remainder, "mentarianism," would be discarded. Other mechanisms of hashing input strings can be implemented by reconfiguring the FPGA.

The method of building words and the limitations of word size are two examples of how the hardware design becomes a factor for the algorithms used to train the system. These factors frame the starting conditions for learning algorithms.

3.2 Mapping Words to Semantic Concepts

Even if 16-byte English words were to be the only input considered, there would still be a large number of possible patterns − equal to 26^{16} = 4.3e+22. Practical systems have a finite amount of memory: therefore it is not possible to associate a unique semantic meaning to all possible combinations of 16-byte sequences. In order to implement a system that uses a reasonable amount of memory, a hash can be used to map words into a smaller range of values which the hardware can process. We implemented a hash function in hardware that can be computed quickly and can effectively process multiple character sets. Our hash produces a 20-bit number as an output. The example given below shows how our system hashes a 5-byte string (40 bit) into a smaller, 20 bit number.

H("HELLO") = 0x2c563 (101,603)

With the output of 20 bits, there is a possibility of 1M (1,048,576) individual representations of acceptable words. The hash used to translate words to numbers is a factor taken into consideration by the learning algorithms.

3.3 Word Mapping Table

Our system performs dimensionality reduction in two steps. As described above, the first part of our system hashes each word to one of 1M values. The second part of our dimensionality reduction occurs via a Word Mapping Table (WMT). The WMT provides a programmable mechanism for mapping each of the 1M words to a smaller subset of up to 4,000 features. To perform this mapping quickly, we implemented the WMT with high-speed Static RAM (SRAM). The first 19 bits of the hash provide a memory address for a 36-bit wide memory bank implemented in Zero-Bus Turnaround Static RAM (ZBT-SRAM). The 20th bit is used to indicate which half of the 36-bit result stores the value. Learning algorithms are used to determine how hash values map into entries in this table. An example use of the WMT is shown in Figure 1.

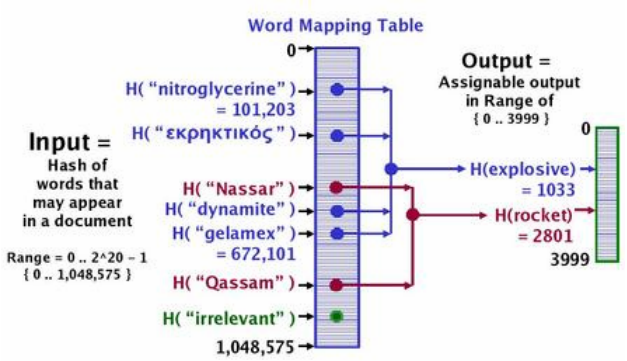

Figure 1. Workings of a Word Mapping Table

3.4 Document Vectors

To track the state of words that appear in all of the simultaneously active TCP/IP flows, counts are maintained to track the occurrence of words in each feature. The numeric range of these counts, times the number of dimensions, determines how much memory is needed to track the state of each flow. In our system, we allow the counts to have a numeric range of 0 to 15. This range is represented with a 4-bit unsigned integer that occupies one half of a byte in memory. To store counts for all 4000 dimensions, we use 4 bits * 4000 dimensions = 16,000 bits (2000 bytes) of memory. We use 256 Mbytes (2^{28}) of low-cost Synchronous DRAM (SDRAM) to track the words that appear in each of 131,072 (2^{17}) simultaneous flows by allocating 2048 (2^{11}) bytes to store each individual flow's state and a feature vector. Of the 2048 bytes, we use 48 bytes to store flow state (source IP address, destination IP address, source port, destination port, and other metadata that describes the traffic flow).

As values are extracted from the WMT, they are passed to the count module to build a document vector. A list of features is received by the count module from the baseword module. Each occurrence of a feature causes a counter for that feature to be incremented. Once a flow completes, a vector of 4000 elements in 2048 bytes that contains counts of all the features is passed to the score module. Saturating arithmetic is used so that if any 4-bit entry overflows (i.e., takes a value greater than 15), the resulting value is set to 15.

3.5 Scoring and Concept Vectors

The scoring module compares vectors that count the

features in TCP/IP flows to vectors that represent concepts. A set of concept vectors that is used to configure the score module is referred to as a Score Table (ST). The concept vectors have the same dimension as the document vectors (4000 elements), but a concept vector assigns a weight to each feature that is representative of the concept. The total score is computed as the sum of the products of the features and weights, as shown in Figure 2.

$$Score_i = \sum_{k=0}^{3999} v_k \bullet cv_{ik}$$

Figure 2. Equation for scoring a document vector *v* against the concept vector cv_i

In our system, two versions (bitfiles) of the score module have been created. One design compares input vectors to each of 15 concept vectors represented with 8-bit resolution. The other bitfile compares input vectors to each of 30 concept vectors represented with 4-bit resolution.

FPGAs can perform numerical processing very quickly by using parallel hardware circuits. To be efficient in amount of space used by these circuits, we limit the complexity of the math. The use of limited-precision integers was a factor considered for the learning algorithms. The scoring of a vector against the concepts in the system is performed using a hardware-implemented circuit that computes a dot product between the document vector and the concept vector. Once all the scores are computed, the results are passed out of the system.

3.6 Score Vectors

The output of the scoring module is called a *score vector*. It consists of the 15 (or 30) scores, along with the sum of the entries of the document vector and the dot product of the document vector with itself. The score vector, along with the document vector, is passed from hardware to software for further processing.

3.7 Classification Algorithms

The method for classification of text is dependent on the learning algorithms used by the system. If a Bayesian method was used in the design of the WMT and ST, then the classification processing would take that into consideration. Likewise, if a spherical distance metric was the criterion for classification, then a different mathematical transform would be performed. Two algorithms were developed for the AFE system, each has different classification metrics.

4. REVIEW OF ALGORITHMS

We experimented with two types of algorithms. The first algorithm was developed by Washington University and the second by the HNC division of Fair Isaac.

4.1 The Washington University (WU) Algorithm

The WU method generates a word mapping table (*WMT*) using an algorithm with the following properties and capabilities:

(1) Stems word using knowledge of a language's morphology;
(2) Performs synonym extension using a thesaurus such as Wordnet;
(3) Considers word-for-word translation using a dictionary;
(4) Benefits from, but does not require, word frequency statistics from training data
(5) Accepts input with explicit keywords
(6) Is effective on data with little or no training sets;
(7) Is robust to idiosyncrasies of the training set;
(8) Is fast (using one or two passes through the data set).

The method is based on principles in information retrieval, augmented with heuristics from information theory. The basic idea is to satisfy the selection of information-bearing features of the training set and to ignore the possibility that two words might be correlated.

Given that a training set from a corpus can have a large number of possible words--10,000+ distinct words for a single language is common--the feature set extraction must be learned. The basic idea is to allocate words to buckets (or features) in three distinct classes. First, reserve a fixed number of "first class" features, one bucket per word, for the words with the highest frequency. This is done after first removing "stop" words, which are defined as words that appear with too high a frequency in the training set. The stop word list can be augmented with background information, so that some words are anchored on this list whether they are present in training documents or not. Stop words are allocated either to a single bucket whose count remains zero or to a single bucket that is presumed to easily saturate its count.

For "second class" features, multiple words are allocated as a single feature, permitting an intentional bucket collision. The proportion of second class to first class features can be specified by the user, and an iterative search is used to find the collision rate that best matches the target proportion. It is important to allocate second class words so that colliding words all belong to the same category. “Third class” features are allocated just to complete the number of buckets at 4000, and are allocated with an even higher collision rate.

To reflect information theoretic ideas, the strategy for allocation is altered in the following way: if a word that would have been given a first class bucket occurs in too large a fraction of the categories, then it is "*demoted*" and allocated a second class bucket instead. Similarly, second class buckets can be demoted to third class. Moreover, a word that would have been removed for having too high a

frequency is actually assigned to a first class bucket if it occurs in a very small number of categories.

This is a three-level discrete version of what could be an algorithm with continuously varying collision rates and more than three classes of buckets. Moreover, it is clear that a word occurring in one category is not the same as a word occurring in two categories, but the method only distinguishes between "*few*" and "*many*". This coarse discretization and the use of frequency as the primary criterion for judging words are deliberate attempts to introduce robustness into the feature set and avoid overtraining. Buckets are weighted by inverse term frequency (ITF) and by document frequency (DF).

4.2 The HNC / Fair Isaac Algorithm

Information retrieval applications often represent documents by term frequency vectors, typically resulting in a high-dimensional space. As described in the previous section, dimensionality reduction can be used to allow for efficient implementation. The reduction is carried out by partitioning the terms of the original vocabulary (the *raw words*) into a smaller number of *basewords*. One familiar example from information retrieval is the identification of words having the same root, "run", such as "running", "runs" and "runner". Synonyms may also be identified in this manner. The partition is referred to in Section 3.3 as the *word mapping table* (*WMT*), as it is represented in hardware as a map from raw words (more exactly, hash values of raw words) to basewords.

A possible technique for creating a WMT is to use thesauri and stemming tables, as suggested by the preceding discussion. A drawback of such an approach is that the linguistic resources may not be readily available to map the languages we would like to process. Another drawback with the method is that much of the text which is informative for classification may be jargon, names, slang, and other non-standard usage that can not be readily mapped. In order to have a dynamic system that continually learns language usage in a fluid environment, we turn to data-driven techniques for this class of algorithms.

Via a WMT, we create the features that will be used to predict the concept C to which each document belongs. The information each feature w possesses is described by the probability $P(C \mid w)$. *Association-grounded semantics* (*AGS*) assigns to each data object w a "meaning" represented by the probability distribution $P(C \mid w)$ over all contexts C. We can then partition the w objects so as to minimize the amount of information lost due to the reduction of the feature space. Details of this technique are given in [6], [7], and [8].

The document is represented as a feature vector in the lower dimensional feature space induced by the WMT. Currently, we are focusing on centroid-based techniques, like the Rocchio algorithm [9], although a number of linear models (including naïve Bayes and linear support vector machines) could easily be loaded into the same hardware. Given a set of document vectors known to belong to a given category C, we create a *concept vector* which represents C by forming the sum of the normalized document vectors using a weighting scheme such as inverse document frequency (IDF). We can then classify a given unlabeled document vector d as belonging to the category represented by that concept vector lying closest to d under one of a number of possible distance measures. We have experimented primarily with using cosine and dot product as similarity measures. When the concept vectors have the same scale, these metrics are equivalent to Euclidean distance.

We anticipate deploying our system in an environment in which most of the documents seen are correctly classified as "none of the above". When using the cosine distance metric, one can place a threshold on the maximum possible angle away from each concept vector that will be considered for classification within that concept. This amounts to augmenting our centroid model with a radius for each centroid. An alternative is to look at the ratio between the largest and second largest dot products and require a minimum gain in order for a document to be classified. Documents which are distant from all centroids will have low ratios and be rejected. This technique has a second advantage in that it abstains from attempting to classify a document that is very close to multiple concepts, which is important when we are especially sensitive to false positive classification. Details for these techniques are presented in [1] and [6].

In addition to classifying documents into known concepts, we would also like to discover concepts in the document stream. A number of techniques are available for pursuing this goal; we have focused initially on clustering, in the sense of discovering a partition on a set of documents for which the equivalence classes tend to correspond to the concepts we are attempting to discover. We have experimented with both K-means clustering and an AGS method in which we attempt to maximize the mutual information between document clusters and the WMT features. This is the same technique used by HNC to derive the WMT but applied to documents distributed over word groups rather than to words distributed over document groups. This intuition motivates an additional possible application of AGS. Given the original raw word space and the original document set, we can simultaneously generate word and document partitions by attempting to maximize the mutual information of the joint distribution of the cluster spaces. Details are presented in [6].

5. EXPERIMENTAL TESTBED

Our approach to building the testbed was to first develop a software simulation of the system, next to design and implement hardware that performed the same functions,

and then finally to verify that the software and hardware systems behaved identically. We developed software simulators for each FPGA circuit:

- *simBase* – baseword circuit simulator
- *simCount* – count circuit simulator
- *simScore* – score circuit simulator

A feature of this approach is modularity. It enables us to prototype new circuits in software before incurring the engineering costs to implement them on FPGAs. Also, during hardware debugging, the outputs from hardware and software may be compared at each step in the process to isolate errors.

5.1 *Train* and *Test* Document Sets

To perform an experiment, documents from a corpus are organized in a directory and divided into two groups. First, labeled *training* documents are used to generate the WMT and the ST, which contains up to thirty prototype concept vectors. These two tables are used to process data by our semantic processing software simulator or hardware bitfiles.

Second, documents in the *test* group are passed through the system and classified or clustered. Although documents in this group are labeled, the label is not exposed to the semantic processor and is used to evaluate the system and algorithms.

5.2 *Signal, Interference,* and *Noise* Document Types

Besides a concept label, the second attribute of every document in our test corpus is a type from the set {*Signal*, *Interference*, *Noise*} chosen to be analogous to Radio Frequency (RF) communications. Documents of type *Signal* exhibit Concepts Of Interest (COI). Documents of type *Interference* exhibit Concepts Not Of Interest (CNOI) analogous to co-channel interference in the RF environment. Finally, documents of type *Noise* are analogous to RF static caused by lightning strikes. In the context of searching for concepts, anything that is not linguistic textual communications constitutes Noise. This includes voice, video, images, executables and machine-to-machine handshaking (such as Keep-Alive messages).

5.3 XML File Specifications

We specify each experiment using an XML file called *config.xml.* This file describes the experiment, identifies each of the labeled concepts, and lists each document in the training and testing groups. The label and type for each document is also captured in *config.xml.* In addition, the input and output formats for each step in the experiment are captured in xml documents.

6. Corpus Collection and Cleaning

For the experiments described below, there are three distinct sources of documents:

(1) Messages posted to online forums**;**
(2) Messages posted to public online newsgroups;
(3) Documents translated from Arabic into English and Urdu;
(4) The 20 Newsgroups corpus.

These sources were mixed in order to create a specific test problem, but it is convenient to describe them individually before describing their combinations.

The forum messages were found in Turkish, Spanish, and English, and covered topics generally classified as astrology, beauty, marriage, cooking, etc. The actual messages in each topic were obtained by following one or two threads within a forum with the topic's title and merging those threads. For example, Turkish beauty might have consisted of several threads, pertaining respectively to hair, diet, and cosmetics. It is especially important to note that some topics are quite parallel across languages, e.g., astrology postings from speakers of English tend to be similar to astrology postings from speakers of Spanish. On the other hand, some topics are much less parallel. An extreme example is marriage, where the threads of an English language forum were quite different from the threads of a Spanish language forum.

The postings to public newsgroups were all in English, and included newsgroups such as logic, martial arts, neural nets, and talk.origins. The last group, talk.origins, is important because it was treated differently. It was used as a source of "chaff" (*Interference*), and it was not hand-vetted for incorrectly classified posts such as spam, meta-conversations, or wrongly directed messages. All of the other topics were human-vetted for correctness and uniqueness.

The Islam Questions & Answers site (www.islamqa.com) was the source for several documents in English, Arabic and Urdu. This site contains answers to questions on various aspects of Islamic law, organized by topic. Topics included cooking, marriage, politics and history.

The 20 Newsgroups corpus [19] is a well-known set of 20,000 postings to 20 newsgroups often used in text classification experiments.

All documents were cleaned of HTML, CSS, and JavaScript. Headers were removed, including names of newsgroups, authors, and dates; also removed were direct quotation of other documents. All of the messages were chosen to meet a minimum length constraint, either 40 or 100 words per document, depending on the source.

For experiments using parallel concepts in multiple languages, a mixture of sources (1), (2) and (3) were used. This was directly the result of a desire to test five languages: English, Spanish, Turkish, Arabic, and Urdu.

For all other experiments, either source (1) or source (4) was used.

7. DESCRIPTION OF EXPERIMENTS

In our experimental program we have considered five classes of factors. The factors are: (1) our testing corpus; (2) generation of the WMT; (3) hardware design factors; (4) classification algorithm; and (5) clustering algorithm. These factors are described below and enumerated in Tables 1 through 5.

The factors shown in Table 1 involve the testing corpus. Although the raw corpus consists of documents downloaded from Google groups, Wikipedia, and various other web sources, subsets of these documents are then combined to generate experimental training and testing conditions. The factors that we vary involve the number of languages, amount of chaff, whether or not chaff is present in the training set, and the size of the training corpus.

Factor	Levels	Notes
Language	1 to 125	How many languages are in the corpus?
Train Chaff	Yes/No	Is chaff present in training documents?
Test Chaff	0, 10, 90%	How much chaff is present in the testing documents?
Train Size	1, 33, 50%	Percentage of corpus used for training.

Table 1 Corpus Factors

The factors shown in Table 2 involve the word mapping table. The first variable that we vary is the algorithm used to create the word mapping table. HNC's algorithm uses co-clustering and maximizes mutual information between words and labels, whereas WU's algorithm uses an information retrieval-based method. Two other variables that we consider are whether or not multilingual training documents from parallel concepts should be merged across languages and whether or not to translate documents before generating the WMT and Score Table.

Factor	Levels	Notes
Algorithm	HNC or WU	HNC = mutual information & co-clustering. WU = information retrieval.
Merge Languages	Yes/No	Yes = merge training concepts across languages to create a single concept. No = separate training concepts by language.
Translate	Yes/No	Translate training documents using lexicons before WMT generation.

Table 2 WMT and Score Table Generation Factors

The factors shown in Table 3 involve the precision of the hardware. Hardware optimized circuits use much less memory than conventional processors. Tradeoffs are made between the precision and saturation points of internal counters and the maximum number of possible concepts.

Factor	Levels	Notes
Baseword Length	3, 4, 5 or 6	Vary minimum length of basewords for processing.
Concept Vector Precision	4, 8 bits	Vary precision (#bits) of concept vectors.
Document Counter Precision	1, 2, 3 or 4 bits	Vary precision (#bits) of document vectors.

Table 3 Hardware Design Factors

Table 4 shows the two scoring algorithms that we consider for classifying documents into concepts. The algorithms, described in Section 4, are the cosine angle between two vectors and the normalized order statistic.

Factor	Levels	Notes
Score Algorithm	Cosine or Order	Cosine = HNC's algorithm Order = WU's algorithm

Table 4 Classification Algorithm Factors

Finally, Table 5 shows the factors involved in a clustering run.

Factor	Levels	Notes
Hierarchical	Yes/No	Should hierarchical clusters be generated?
Clustering Algorithm	Ants, Co-clustering, K-Means, Nearest Neighbors	Which clustering algorithm should be used?
Clusters	30-200	Number of clusters to be discovered.

Table 5 Clustering Algorithm Factors

To investigate these factors, we performed two sets of experiments named, respectively, the January and May experiments. Each experiment consisted of a sequence of runs. In the January experiment we considered:

- Language identification
- Spanish concept discovery
- Parallel concept discovery across languages

In the May experiment we considered:

- Concept discovery across languages (tests 1 to 4)
- Translating training documents into English for creating WMTs (tests 5 & 6)
- Parametric optimization of the FPGA hardware design factors (test 8)
- Concept discovery and clustering (test 10)

The experiments were run in software and duplicated by the hardware system.

8. EXPERIMENTAL RESULTS

8.1 January Experiments

In this section we describe our January experiments.

8.1.1 Multilanguage Identification

Our experimental corpus is made up of 800 documents in

12 languages: Danish, Dutch, English, French, German, Italian, Portuguese, Spanish, Swedish, Turkish, Uzbek and Arabic. Twenty-three thousand noise files in the form of Russian and Japanese PDF, GIF, JPEG and HTML files are added. Our system worked remarkably well. Our recall and precision are close to 100% at optimal thresholds. When the thresholds are relaxed, the noise files tend to get misclassified as English or German. HTML files contain a fair amount of English and thus would naturally be classified as English. Less obvious was the classification of PDF files as German. Examination of some of these files revealed that they all contained the ASCII strings "Acrobat" and "PDF". It turns out that exactly two of the training files contained references to Adobe's Acrobat Reader, and these two files happened to be German files.

The confusion matrix in Table 6 shows the misclassification rates. The true labels are shown on the left and the assigned concept labels are shown along the top. Precision, recall and F1 statistic are shown for each category. This confusion matrix shows that the system worked very well for language identification.

Assigned Concepts

True Concepts	danish	dutch	english	french	german	italian	portugues	spanish	swedish	turkish	uzbek	arabic	reject	RECALL	FMEASURE
danish	13													100%	100%
dutch		13												100%	100%
english			57										10	85%	88%
french				64									3	96%	98%
german					47								20	70%	76%
italian						66								100%	100%
portugues							60						7	90%	94%
spanish								66						100%	100%
swedish									14					100%	100%
turkish										13				100%	100%
uzbek											13			100%	100%
arabic												67		100%	100%
chaff			6		10								22984	100%	NaN
PRECISION	100%	100%	90%	100%	82%	100%	100%	100%	100%	100%	100%	100%	100%	NaN	NaN

Table 6 Confusion matrix for HNC algorithm, 33% training

To help us understand why certain documents get misclassified, we developed new visualization techniques to show words. Figure 3 shows such a misclassification for an experiment reported on in [1]. Here, an interference document was classified as *archaeology*. The actual label of this document is *art history*, which wasn't one of the topics in the training set. Words that contributed toward the archaeology score are in **blue**, while the weight of their contribution is denoted by the type size and superscript (1-255). The key words in this concept are 'archaeologist', 'clay', 'Neolithic', 'Mesolithic', and 'millennium'. Words not seen during training are in **green**. If a word in green has a non-zero superscript, it is the result of a hash collision with a relevant word for this concept. The most interesting of the green words are 'allure' and 'metallurgy'. Words in **black** are words that receive zero weight for this concept, either because they are short (fewer than 3 letters) or are not correlated with the concept.

Figure 3. Words contributing to document score in blue and contribution denoted by superscript

8.1.2 Spanish Concept Identification

We created a Spanish-only corpus of forum postings on astronomy, cell phones, cooking, dogs, elections, explosives, philosophy, videogames and the astrological signs Aries and Libra. No chaff was used. Using the WU WMT with 33% training, we achieved 89% precision and 92% recall (Table 7).

Assigned Concepts

	philosophi	election	dogs	aries	astro	cooking	libra	explosives	cellular	videogames	reject	RECALL	FMEASURE
philosophi	478	5	10	9	6	0	0	4	0	0	44	86%	91%
election	0	455	2	4	1	0	0	0	0	0	24	94%	95%
dogs	1	0	378	3	0	1	0	3	0	1	20	93%	95%
aries	4	1	0	290	0	0	7	0	1	0	53	81%	86%
astro	7	5	0	2	289	1	0	0	0	0	27	87%	92%
cooking	0	0	0	0	0	66	0	0	0	0	7	90%	94%
libra	0	0	0	10	0	0	38	0	0	0	20	56%	67%
explosives	0	0	0	0	0	0	0	28	0	0	0	100%	86%
cellular	0	1	1	1	0	0	0	0	20	0	0	87%	89%
videogames	0	1	1	0	0	0	0	2	1	13	2	65%	76%
PRECISION	98%	97%	96%	91%	98%	97%	84%	76%	91%	93%	0%	NaN	NaN

Table 7 Confusion matrix for WU algorithm, 33% training

8.1.3 Arabic Concept Identification

We achieved similar performance from Arabic concept identification. The corpus was comprised of 9000 transliterated news articles from www.alwatan.com and categorized into topics of sports, economy, culture, state news, and religion. With only 1000 documents for training, the WU algorithm achieves 93% accuracy (with no abstentions).

8.1.4 Parallel Concept Identification

The hardware can only store fifteen 8-bit-wide concept vectors in the score table (or thirty 4-bit-wide vectors). This constraint poses a problem for multilingual multiple concept identification. If we wanted to train the system to recognize five concepts in three languages, we could create 15 concept vectors, one for each language-concept pair. To accommodate a larger number of concepts, one possibility is to merge parallel documents across languages into one super-concept. This would give us the ability to recognize up to 15 concepts in any number of languages.

In this experiment, we created a corpus from newsgroup articles in English, Spanish and Turkish. We trained on 33% of the documents from the following topics: astrology, beauty, cooking, marriage, music, poetry and travel. Interference documents from other topics were also included during testing, but there was no Noise. The best results achieved were 56% precision and 78% recall with the WU algorithm. Parallel concept identification remains an area of research for the group.

8.2 May Experiments

In this section we describe our May experiments.

8.2.1 Tests 1 & 2: Marriage vs. Other

In Test 1, we assessed the ability of the classifier to differentiate between one *Signal* category and several *Interference* categories. We created a multilingual "marriage" category composed of marriage documents in English, Spanish, Turkish, Arabic and Urdu. The "other" category comprised of multilingual documents on Islam (fasting, politics, history and proselytizing) and newsgroup articles (cooking, astrology and travel). The classifier was trained on the two categories with 33% of the corpus. During testing, we only scored against the "marriage" concept: the only choices for the classifier were "marriage" or "reject". The number of documents assigned to each of the labeled categories is shown in Table 8. At the threshold that gives 80% precision, the system's recall was 47% and overall accuracy was 69%.

Label/assigned	Marriage	Reject
Marriage	345	391
Other	86	708

Table 8 Confusion matrix for HNC

In order to quantify the loss of performance caused by merging categories across languages, we split the marriage category into 5 sub-categories according to language as shown in Table 9 (Test 2). We kept the "other" category multilingual.

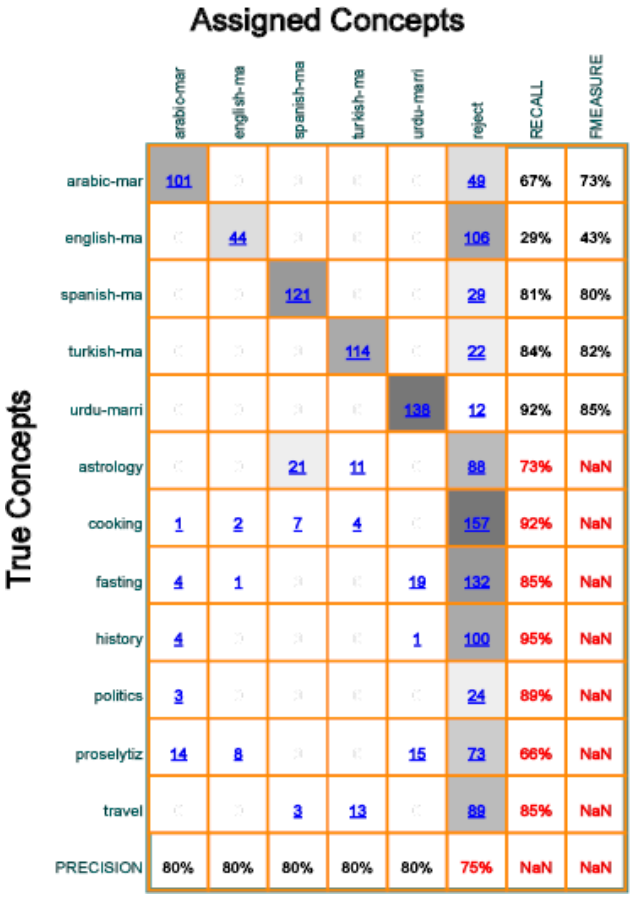

Table 9. Confusion matrix for HNC

Splitting up marriage by language increased recall to 70% and accuracy to 77% (still at 80% precision). A downside is that documents from the "other" category were classified according to their language. For example, all Arabic cooking documents were classified as Arabic-marriage.

8.2.2 Tests 3 & 4: Parallel Concepts

The next pair of tests follows the same format as the first two. In Test 3, instead of a single Signal category, we had 5 categories: astrology, cooking, marriage, poetry and travel. The "other" category was made up of documents on beauty, dogs, gaming, nature and philosophy. Each category was "integrated" from documents in English, Spanish and Turkish. Performance was 69% recall and 69% accuracy at 80% precision. These results are shown in Table 10.

For Test 4, we segregated each of the five Signal categories by language to get fifteen Signal categories. In this test, performance did not significantly improve over the integrated version.

True Concepts \ Assigned Concepts	astrology	cooking	marriage	poetry	travel	reject	RECALL	FMEASURE
astrology	124	8	9	7	6	47	62%	70%
cooking	5	135	3	3	5	10	84%	82%
marriage	4	4	122	8	3	60	61%	69%
poetry	9	5	8	123	11	45	61%	69%
travel	1	3	3	2	149	29	80%	80%
beauty	5	5	1	2	2	52	78%	NaN
dogs	0	4	5	1	3	55	81%	NaN
gaming	2	2	1	2	1	24	75%	NaN
nature	3	0	0	2	5	5	33%	NaN
philosophy	2	2	0	3	1	7	47%	NaN
PRECISION	80%	80%	80%	80%	80%	43%	NaN	NaN

Table 10 Confusion matrix for HNC

8.2.3 Tests 5 & 6: Translation

The following two tests involved machine translation of Spanish and Turkish documents into English using lexicons. In Test 5, we built the word mapping table out of translated documents from the astrology, cooking, marriage, music, poetry, travel and beauty categories. We then tested native English documents from those categories and additional interference categories (art history, cats, dating, dogs, gaming and gardening). At 80% precision, recall was only 25%, and accuracy was 38%.

Test 6 added 1/3 of the native English files into the training set. Recall increased to 80% and accuracy to 74%, but this gain came solely from the addition of the English training files. There is still much for us to research if we hope to build reliable WMTs using translation.

8.2.4 Test 8: Parameter Variation

We used a cleaned-up version of the newsgroup corpus employed in [1]. This corpus contains 735 Signal documents from 11 newsgroups, along with 10,768 Interference documents from the talk.origins newsgroup.

We compared the classification performance of the system while varying two independent parameters: minimum baseword length (variable, between 2 and 12) and concept vector precision (4 or 8 bits). Rather than display a large table of results, we shall summarize our observations.

Somewhat surprisingly, the system's performance degraded only a little when using just 4 bits of precision in the score vectors. The topics which suffered from the lowered precision tended to be those for which classification accuracy was low to begin with.

We found that average recall--at 80% precision--was maximized when the minimum acceptable baseword length was around 7 or 8 characters, as displayed in Table 11a. While performing the same type of test on a different corpus, we found that recall was maximized when the minimum word length was three, the current system default (Table 11b). The major difference between the two corpora is the prevalence of interference. The test set in the first corpus is 94% interference, while the second is only 64% interference. We hypothesize that increasing the minimum word length is helpful only in an extremely low signal to interference environment.

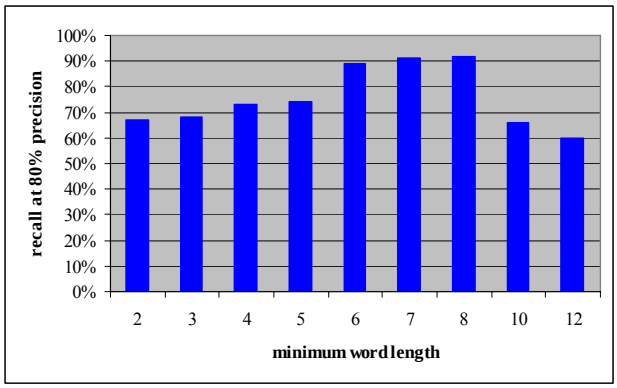

Table 11a. Recall versus minimum word length for 94% chaff corpus.

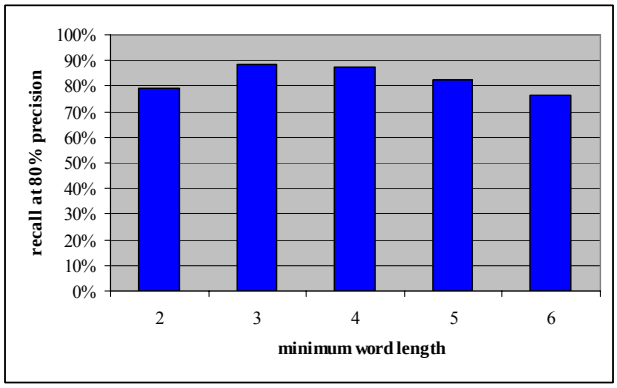

Table 11b. Recall versus minimum word length for 64% chaff corpus.

8.2.5 Test 10: Clustering.

The goal of the clustering experiment was to determine our ability to discover concepts that were not made available during any supervised part of training; i.e., during creation of the WMT and the ST. We used the CMU 20 Newsgroups corpus [19] where only 2007

documents taken from 13 of the 20 newsgroups were available during creation of the WMT. We then ran clustering algorithms on the full corpus, augmented with 10,768 chaff documents from the talk.origins newsgroup (not one of the 20 Newsgroups). The target number of clusters was 60.

One natural way to view the results of the clustering with respect to ground truth is to assign all documents in each cluster to the concept to which the plurality of the documents belong, then look at the resulting confusion matrix. The confusion matrix shown in Table 12 is for the information theoretic technique described in Section 3, in which document clusters are chosen to maximize mutual information with the WMT features. Cluster labels are given on the left , while true labels appear on the top.

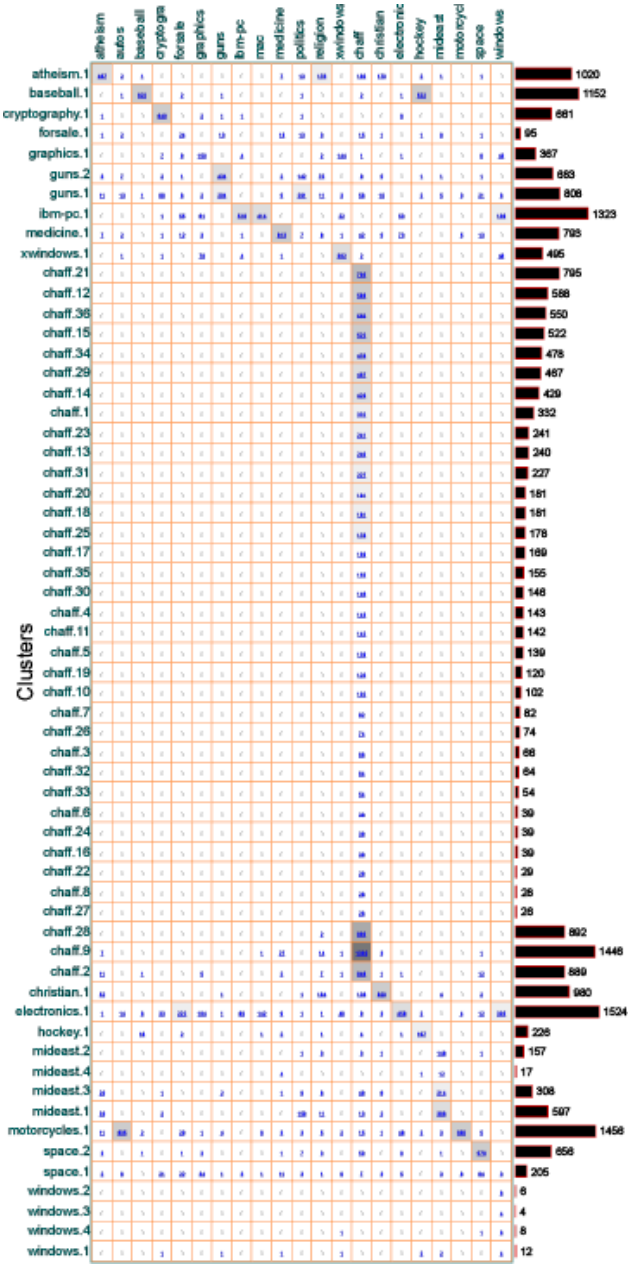

Table 12 HNC Confusion matrix

In the second row, we see that there is confusion between the *baseball* category and the *hockey* category. The seventh row from the bottom shows that there also was confusion between *motorcycles* and *autos*. Finally, we observe that the system clustered most of the chaff documents into 36 pure chaff clusters, starting from the eleventh row.

Clearly, the best possible clustering would have a single non-zero value in each row. An appropriate ordering of the rows would give the table somewhat of a diagonal appearance (although some columns would necessarily appear in multiple rows, as there are 21 columns and 60 rows). The underlying diagonal structure of the matrix in Table 12 can be seen from the shaded blocks. In the worst possible clustering, each row would have the same distribution across columns as the full population. A common way to formalize these intuitions and place a numeric score on those partitions which are neither worst nor best possible is to compute the mutual information between the rows and the columns of the table (i.e., between the cluster labels and the ground truth labels). The mutual information $I(X;Y)$ between two random variables X and Y is given by

$$I(X;Y) = H(X) + H(Y) - H(X,Y)$$

where $H(X)$ is the entropy of X:

$$H(X) = -\sum_{i=0}^{m-1} P(X = x_i) \log P(X = x_i)$$

and $H(X,Y)$ is the entropy of the joint distribution over X and Y:

$$H(X,Y) = -\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} P(X = x_i, Y = y_j) \log P(X = x_i, Y = y_j)$$

where m and n are the cardinalities of the X and Y grounding spaces, respectively. Logarithms are taken base two.

Mutual information does not provide a meaningful comparison between partitions having different a number of classes. A metric that is similar to mutual information but *is* intended to provide meaningful comparisons between partitions of differing granularity has recently been introduced by Meila [10] under the name *variance of information* (VI):

$$VI(X;Y) = 2H(X,Y) - H(X) - H(Y) = H(X,Y) - I(X;Y)$$

VI acts like a distance metric: lower values indicate greater similarity. Since mutual information puts the highest values on the most similar pairs, it is natural that it be negated in the VI formula. The only real difference between the two is that VI penalizes those distributions which have higher joint entropy. If one of the partitions

being considering is fixed (such as a ground truth partition), then this is a penalty on the entropy of the learned partition. This matches the intuition that we would like to penalize those clusterings with a larger number of clusters. In fact, VI does have the desirable property that consecutive refinements of a clustering continually move further away.

We consider information-theoretic clustering as described above on a few different data sets and in a few different feature spaces. The first WMT was derived from the 2007 training documents: this is called the "labeled" training set. The second WMT was derived using only 705 of these labeled documents, but was also given the other 1302 documents without their labels. This set was further augmented with 1632 documents from the remaining 7 newsgroups, again without labels, and with 7118 unlabeled documents from *talk.origins*. The idea is that the large amount of unlabeled data should help to make up for the missing labels. This is referred to as the "mixed" training set in the table below. We applied the clustering algorithms to: (a) just the CMU 20 newsgroups, and (b) the 21 groups including *talk.origins*. For each, we considered clustering: (a) all documents, and (b) only those documents not included in the large mixed training set (we call this the "test" set of documents since it is the complement of the training set). We report the ground truth (category) entropy for each data set and the cluster entropy, mutual information, and variance of information for each resulting partition. The "% max" column of Table 12 indicates the percentage of category entropy recovered in the mutual information.

Training	Data	Category	Cluster	MI	% max	VI
Labeled	21: all	3.352	5.450	2.608	77.8%	3.59
Mixed	21: all	3.352	5.513	2.452	73.1%	3.96
Labeled	21: test	3.957	5.525	3.042	76.9%	3.40
Mixed	21: test	3.957	5.604	3.028	76.5%	3.51
Labeled	20: all	4.301	5.568	3.404	79.1%	3.06
Mixed	20: all	4.301	5.579	3.257	75.7%	3.37
Labeled	20: test	4.304	5.609	3.267	75.9%	3.38
Mixed	20: test	4.304	5.588	3.200	74.4%	3.49

Table 13 Clustering metrics

We conclude that training on the mixed data does bring us close to the performance of training on the labeled data set. This is important in environments where large amounts of data are available at training time, but very little of it has ground truth labeling. Continuing research in this direction will explore using even smaller sets of labeled documents.

9. DISCUSSION

Our system is designed to classify and cluster documents. The classification task is to answer the question: Is a given document on one (or more) of the Concepts Of Interest? The clustering task is to answer the question: Is a given document (which appears not to be on a specified Concept Of Interest) similar to any other documents?

Concepts transcend language. If a concept of interest is "Attack at dawn" then it hardly matters whether the document is written in English, Esperanto, or Klingon. The concept of the content is the essence of the information contained within the document and the specific language is merely an additional layer of coding used to transmit the document from one human to another. One of the goals for this system is the ability to train the system on concepts in one language and have it identify documents on those concepts in any language. Preliminary experience with a Wikipedia-based corpus suggests much higher precision and recall numbers than achieved in this paper when the concepts are truly parallel across languages.

On any high-speed link we can expect a vast amount of Interference and Noise. It is not difficult to distinguish Noise, as defined above, from Signal and Interference. The difficulty lies in effectively distinguishing the Signal from the Interference where the Signal to Interference ratio is extremely small. Hence high precision is much more important than high recall.

Human communications are full of ambiguity and assumptions of common knowledge. Much of our testing has been performed using English language newsgroup documents from which we removed the name of the newsgroup, the author and the subject line. These deletions were done to eliminate obvious clues as to the concept under discussion. However, this also hid much of the assumed shared knowledge between the author and the other newsgroup members. Imagine that the Concepts Of Interest include "Cat Behavior" and "Dog Behavior". Once the context of the document is lost, as often happens in newsgroups like *rec.pets.cats.anecdotes*, it is a difficult task to properly classify a story about "Freddy chasing a ball of yarn".

The performance of our system works surprisingly well given that it uses a "bag of words" approach. The result of classification (or clustering) performed by our system is independent of word order within the documents. One may wonder how well humans could classify documents if they were only given a list of words that appear in the document. Machines perform this task surprisingly well.

10. FUTURE WORK

Going forward, we are developing additional hardware-accelerated modules that enhance the accuracy of content classification and clustering by using more information about the communication. An Application-Level Processing module extracts text and metadata from HTML, XML, email, and other well-defined formats. A lexicon processing module enables processing of multi-word phrases. A clustering module groups together content that is related but previously unclassified. Additional modules can be integrated into the system to

perform signal processing and image processing functions for the additional data in media-rich documents.

11. REFERENCES

[1] Lockwood, J., Eick, S., Weishar, D., Loui, R., Moscola, J., Kastner, C., Levine, A., Attig, M., "Transformation Algorithms for Data Streams", IEEE Aerospace Conference, March, 2005.

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

[3] http://www.globalvelocity.com/, Oct. 2005

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

[5] Kastner, C., . Covington, G, Levine, A., Lockwood, J., "HAIL: A Hardware-Accelerated Algorithm for Language Identification", 15th Annual Conference on Field Programmable Logic and Applications (FPL); Tampere, Finland; August 24-26, 2005.

[6] Byrnes, J. and Rohwer, R., "Text Modeling for Real-Time Document Categorization", IEEE Aerospace Conference, March, 2005.

[7] R. Rohwer, D. Freitag. "Towards Full Automation of Lexicon Construction", *Human Language Technology/North American chapter of the Association for Computational Linguistics*, Boston, 2004.

[8] Rohwer, R., "The Co-clustering Algorithm," Internal Memo, HNC Software, February 8, 2002.

[9] Rocchio, J., "Relevance Feedback in Information Retrieval", in The SMART Retrieval System: Experiments in Automatic Document Processing, Chapter 14, pp. 313—323, Prentice-Hall Inc., 1971.

[10] Meila, M., "Comparing Clusterings", *The 22nd International Conference on Machine Learning*. Bonn, Germany, 7-11 August, 2005

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] http://people.csail.mit.edu/jrennie/20Newsgroups

12. BIOGRAPHIES

John Byrnes *is a software engineer at HNC/Fair Isaac. He currently works to develop and apply information-theoretic techniques to the problem of machine-understanding of unstructured data. He previously served as a research scientist at Kromos Technology, where he helped to develop novel signal processing techniques. As research associate at Carnegie Mellon University he investigated proof-theoretic properties for the efficient implementation of natural deduction proof search. He holds a PhD in Pure and Applied Logic and a BS in Mathematics and Logic and Computation from Carnegie Mellon University.*

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). Dr. 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 60 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 principal 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 and 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.*

Andrew Levine *is a graduate student working in the FPX group of the Applied Research Laboratory in the Computer Science and Engineering Department of Washington University in St. Louis. He has a B.S. in Physics from Washington University in St. Louis.*

Justin Mauger *is a Senior Technical Engineer at SAIC Advanced Systems & Concepts. He holds a B.S. in Mathematics from the University of Washington (1995), and a Ph.D. in Mathematics from Northwestern University (2001). Prior to joining SAIC, Dr. Mauger taught Mathematics at California State University, Channel Islands.*

Alan Ratner *is a Senior Expert at the National Security Agency. He has over 30 years of experience in many aspects of electronic and cognitive communications. Mr. Ratner has a B.S. in Physics and M.S. in Nuclear Engineering from MIT, an M.Phil. in Engineering and Applied Science from Yale, and an MBA from University of Bradford in England.*

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.*