A Configurable-Hardware Document-Similarity Classifier to Detect Web Attacks

Craig Ulmer
cdulmer@sandia.gov

April 20, 2010

Craig Ulmer — Sandia National Laboratories, California
Maya Gokhale — Lawrence Livermore National Laboratory

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Overview

- Network security is challenging, especially at link speed
  - FPGAs offer convenient means of brute-force pattern matching
  - Attackers game network intrusion detection systems
- Network researchers: machine learning for better classification
  - Document Similarity via TFIDF and Cosine Similarity
  - Found >94% accuracy in HTTP attack classification
  - But, slow and utilized **46MB** of dictionary data
- Adapt document similarity to an embedded form
  - Simplifications, dictionary reductions, parallel Bloom filters
  - Tools to automatically generate FPGA hardware
  - Achieve 94% accuracy with 128KB dictionary at GigE rates

Outline

- Introduction
  - Discovery challenge and LLNL approach
- Adapting to embedded hardware
  - Algorithm modifications
  - Data modifications
- Implementation details
  - Core design
  - Tool flow
  - Performance and resource utilization
- Future work

ECML/PKDD 2007 Discovery Challenge

- HTTP Traffic Classification
  - Apply machine learning to identify malicious activity in HTTP
- Hand-labeled datasets of HTTP flows
  - Training: 50K inputs, 30% attacks
  - Testing: 70K inputs, 40% attacks
  - 7 Attack Types XSS, SQL/LDAP/XPATH injection, path traversal, command execution, and SSI

Flow Example

```
GET /eH/first_str/2hFnull6/oixsotcwrseamgit2/38PrR_Lkmmzo.htm
Host: www.a215Een.st:15
Connection: close
Accept: */*
Accept-Charset: *;q=0.4
Accept-Encoding: *
Accept-Language: boHEor-sen0, gte-htmse4 oS, 3TeoUsHn-asrao;q=0.2, paly-wreihi, 78iiqths-ar;q=0.3
Cache-Control: no-store
Client-ip: 200.91.18.159
Cookie: uciy2kleicl=%3C%21--+%23odbc+++++++++++++connect%3D%226at8h%2CHcteil%2CeHnNa%22+++++statement%3D%22drop+table+elkbO…
```

odbc

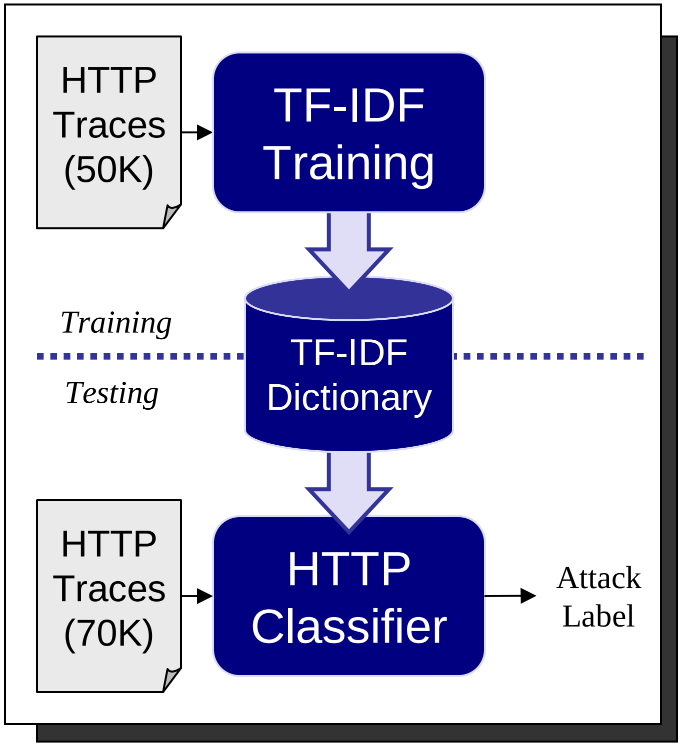
connect

statement

drop table

Prior LLNL Work

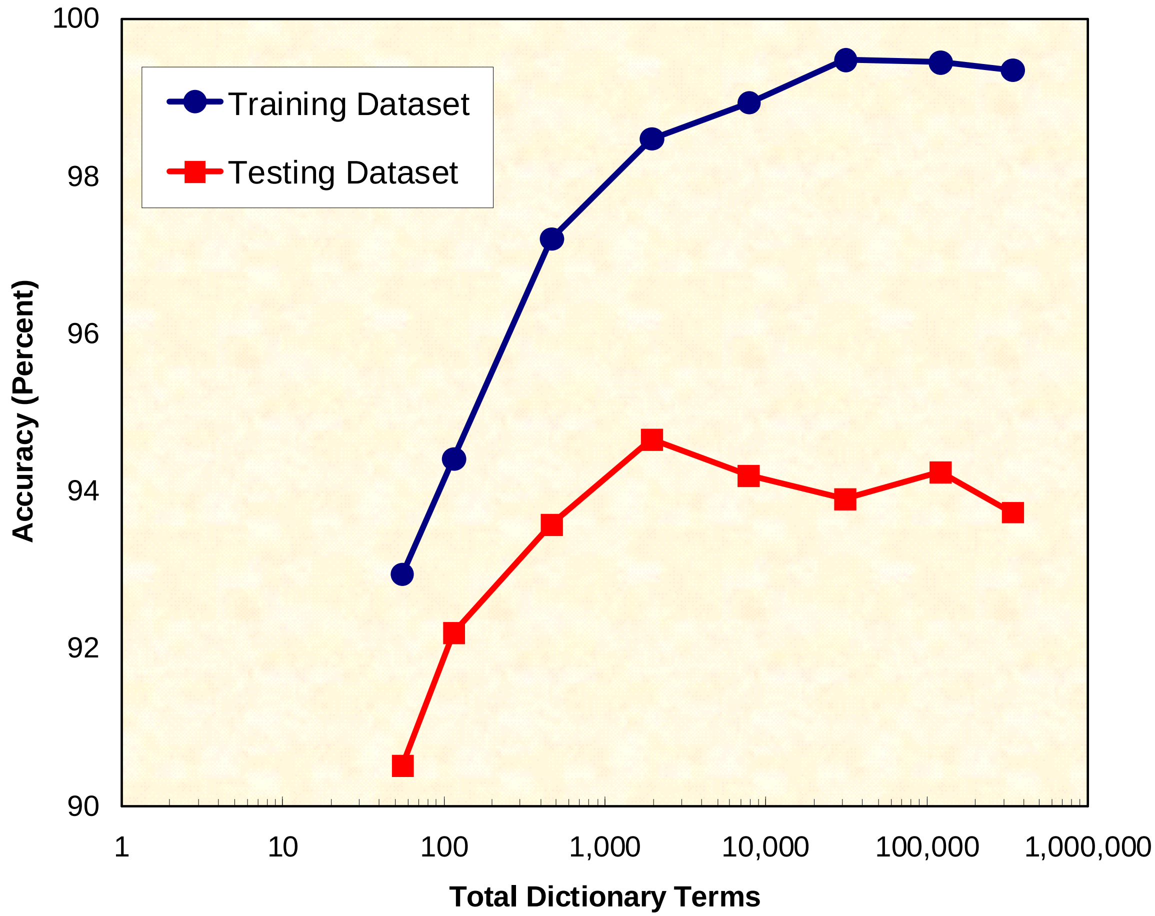
- Brian Gallagher and Tina Eliassi-Rad
- Document similarity: vector approach
  - Tokenize input
  - Assign weights to tokens via TFIDF
  - Cosine similarity for vector comparison
- Relies on a data dictionary
  - Generate term statistics during training
  - Reference statistics at runtime
  - Each term: IDF value and C weights



Term	IDF	CValid	CA1	CA2	CA3	CA4	CA5	CA6	CA7
odbc	2.079	0.0002	0	0	0.0134	0	0	0	0
statement	2.079	0.0013	0	0	0.0134	0	0	0	0
--	0.988	0	0	0.011	0.0126	0	0	0	0

Sandia National Laboratories

Original Accuracy



Outline

- Introduction
  - Discovery challenge and LLNL approach
- Adapting to embedded hardware
  - Algorithm modifications
  - Data modifications
- Implementation details
  - Core design
  - Tool flow
  - Performance and resource utilization
- Future work

Hardware Adaptation Challenges

- Computation
  - Blocking form
  - Floating point math
  - Divide and square root operations
- Dictionary: 46MB, 1.8M terms
  - Large storage
  - Lookup overhead
- Path for converting to hardware
  - Build the hardware design once
  - Automatically update with configuration data

Modification 1: Computation Adjustments

$$score[classifier] = \frac{\sum_{t \in a \cap R} \left(\frac{count[t]}{\#\, input\ terms} \cdot idf[t] \right) \cdot tfidf[classifier][t]}{\sqrt{\sum_{t \in a} \left(\frac{count[t]}{\#\, input\ terms} \cdot idf[t] \right)^2} \cdot tfidfMagnitude[classifier]}$$

Modification 1: Computation Adjustments

Count each term in Input

$$score[classifier] = \frac{\sum_{t \in a \cap R} \left(\boxed{\frac{count[t]}{\#\, input\, terms}} \cdot idf[t] \right) \cdot tfidf[classifier][t]}{\sqrt{\sum_{t \in a} \left(\frac{count[t]}{\#\, input\, terms} \cdot idf[t] \right)^2} \cdot tfidfMagnitude[classifier]}$$

Modification 1: Computation Adjustments

Lookup IDF for term
in dictionary

Count each term in Input

$$score[classifier] = \frac{\sum_{t\in a\cap R}\left(\boxed{\frac{count[t]}{\#\,input\,terms}}\cdot \boxed{idf[t]}\right)\cdot tfidf[classifier][t]}{\sqrt{\sum_{t\in a}\left(\frac{count[t]}{\#\,input\,terms}\cdot idf[t]\right)^2}\cdot tfidfMagnitude[classifier]}$$

Modification 1: Computation Adjustments

Lookup IDF for term
in dictionary

Lookup weight for term
in dictionary

Count each term in Input

$$score[classifier] = \frac{\displaystyle\sum_{t \in a \cap R} \left(\boxed{\frac{count[t]}{\#\,input\;terms}} \cdot \boxed{idf[t]} \right) \cdot \boxed{tfidf[classifier][t]}}{\sqrt{\displaystyle\sum_{t \in a} \left(\frac{count[t]}{\#\,input\;terms} \cdot idf[t] \right)^2} \cdot tfidfMagnitude[classifier]}$$

Modification 1: Computation Adjustments

Lookup IDF for term in dictionary

Lookup weight for term in dictionary

Count each term in Input

$$score[classifier] = \frac{\sum_{t \in a \cap R} \left(\boxed{\frac{count[t]}{\#\, input\ terms}} \cdot \boxed{idf[t]} \right) \cdot \boxed{tfidf[classifier][t]}}{\sqrt{\sum_{t \in a} \left(\frac{count[t]}{\#\, input\ terms} \cdot idf[t] \right)^2} \cdot \boxed{tfidfMagnitude[classifier]}}$$

Adjust based on weight of classifier

Modification 1: Computation Adjustments

Lookup IDF for term
in dictionary

Lookup weight for term
in dictionary

Count each term in Input

$$score[classifier] = \frac{\sum_{t \in a \cap R} \left(\boxed{\frac{count[t]}{\# input\ terms}} \cdot \boxed{idf[t]} \right) \cdot \boxed{tfidf[classifier][t]}}{\boxed{\sqrt{\sum_{t \in a} \left(\frac{count[t]}{\# input\ terms} \cdot idf[t] \right)^2}} \cdot \boxed{tfidfMagnitude[classifier]}}$$

Scale based on TF-IDFs found
by ALL classifiers

Adjust based on weight
of classifier

Modification 1: Computation Adjustments

Lookup IDF for term
in dictionary

Lookup weight for term
in dictionary

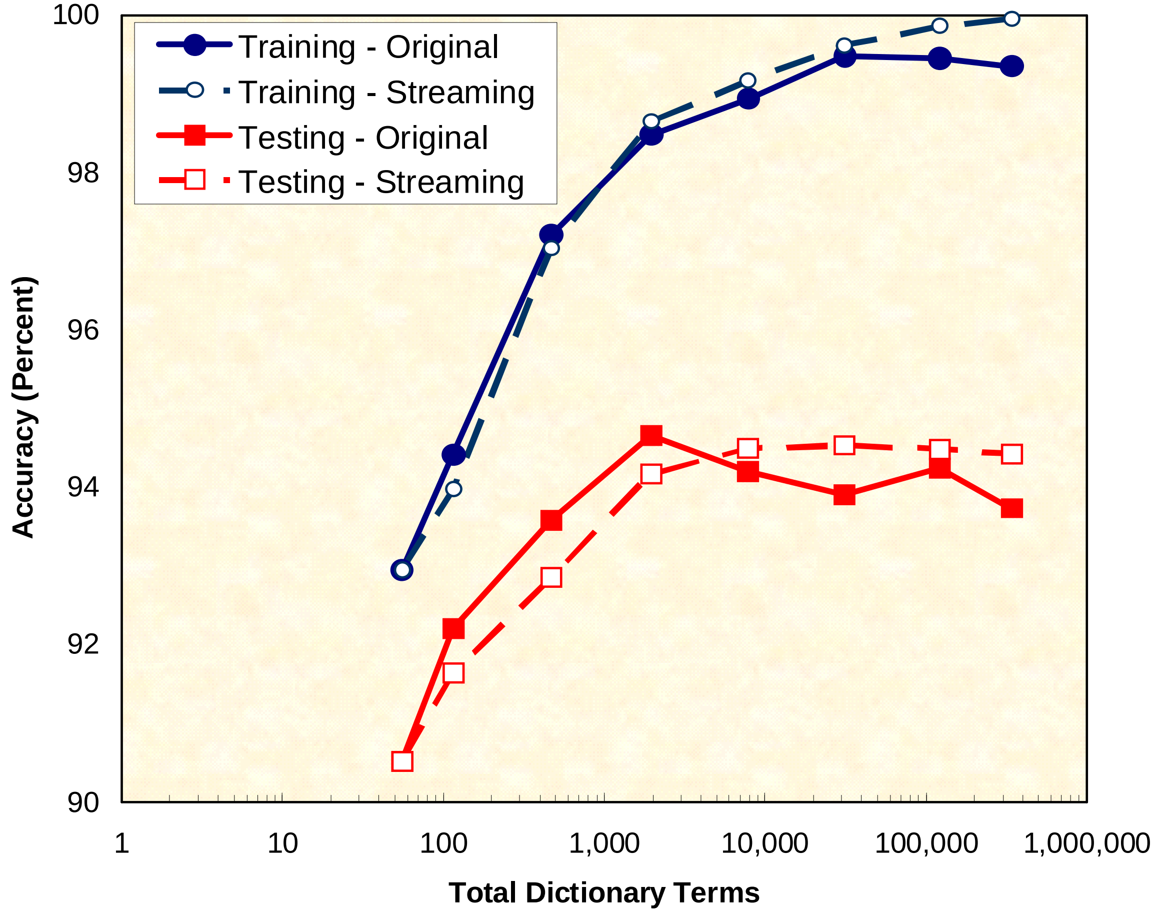
Count each term in Input

$$score[classifier] = \frac{\sum_{t \in a \cap R} \left(\cancel{\boxed{\frac{count[t]}{\#\,input\ terms}}} \cdot \boxed{idf[t]} \right) \cdot \boxed{tfidf[classifier][t]}}{\cancel{\boxed{\sqrt{\sum_{t \in a} \left(\frac{count[t]}{\#\,input\ terms} \cdot idf[t] \right)^2}}} \cdot \boxed{tfidfMagnitude[classifier]}}$$

Scale based on TF-IDFs found
by ALL classifiers

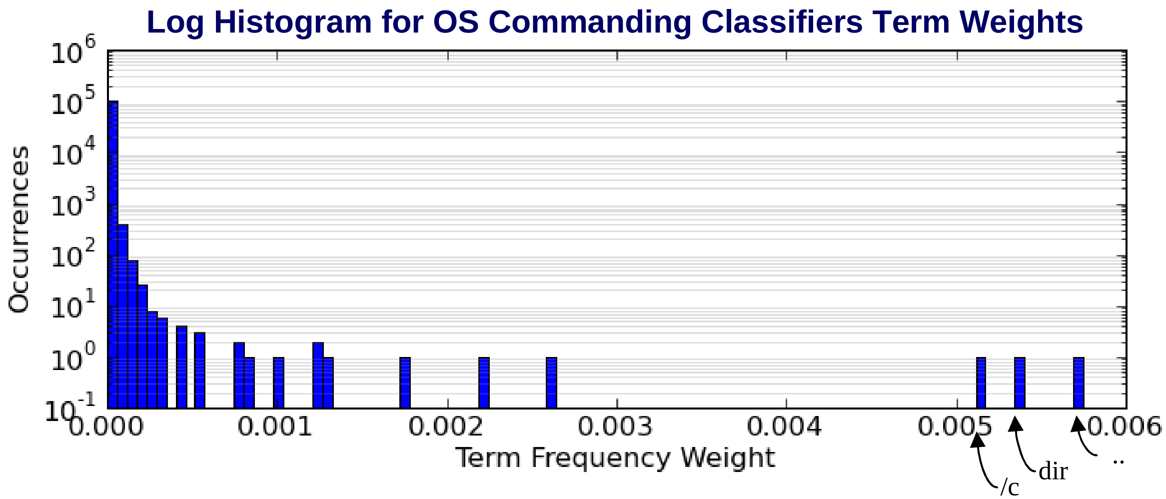
Adjust based on weight
of classifier

Modification 1: Impact on Accuracy

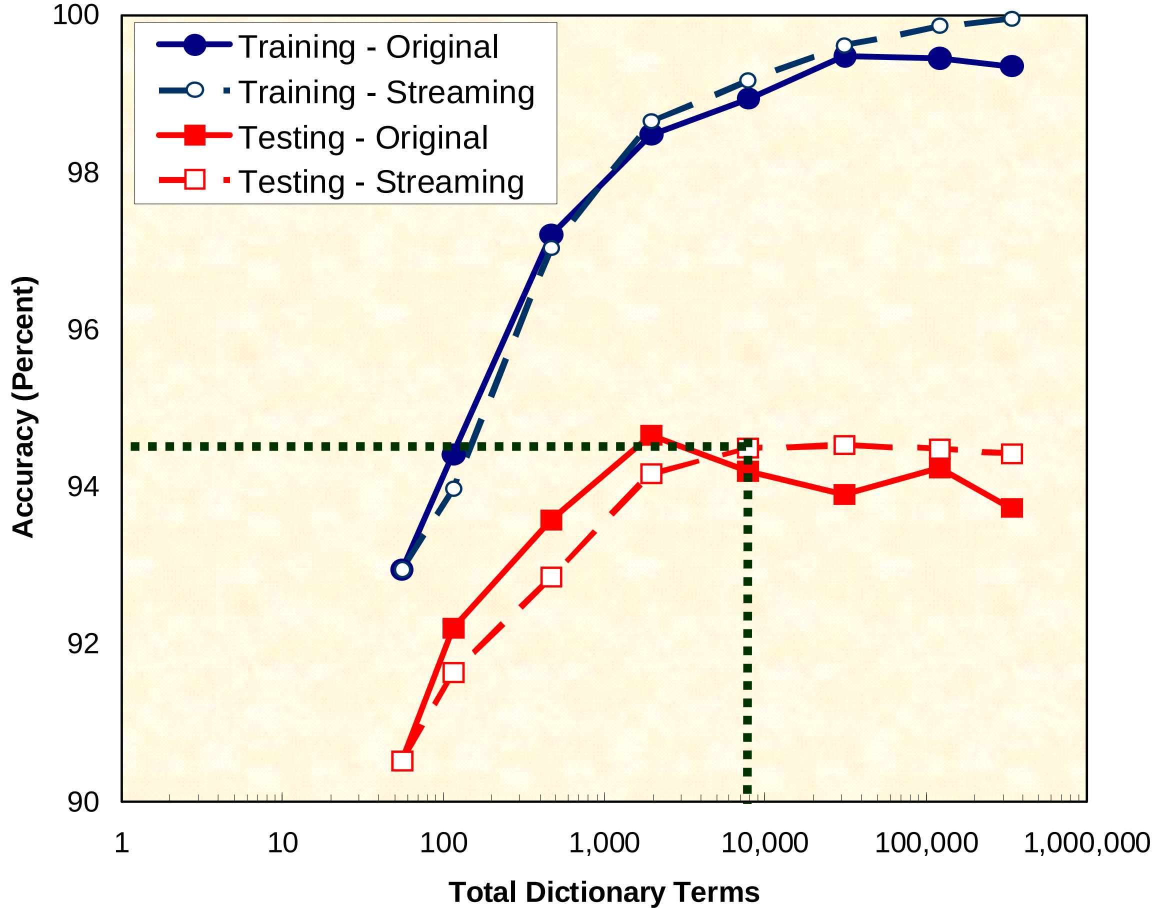


Dictionary Observations

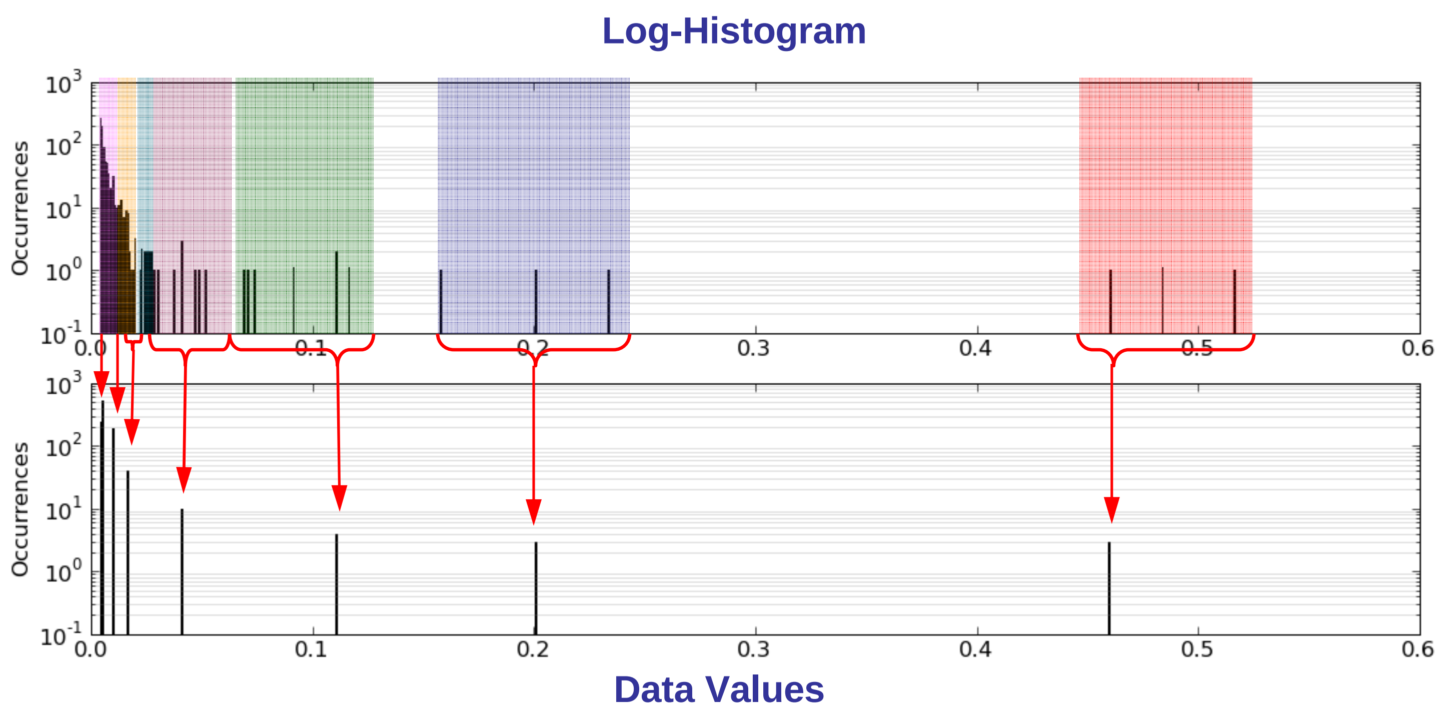
- Many terms in the dictionary
  - 1.8M terms (46MB text, 128MB data)
  - Many terms are junk (“rv:0.7.8”), but they also get very low weight
- Data values are not very diverse
  - Total unique values is < 2% of population
  - Eg: OSC Classifier has 102K terms, but only *415* unique weights



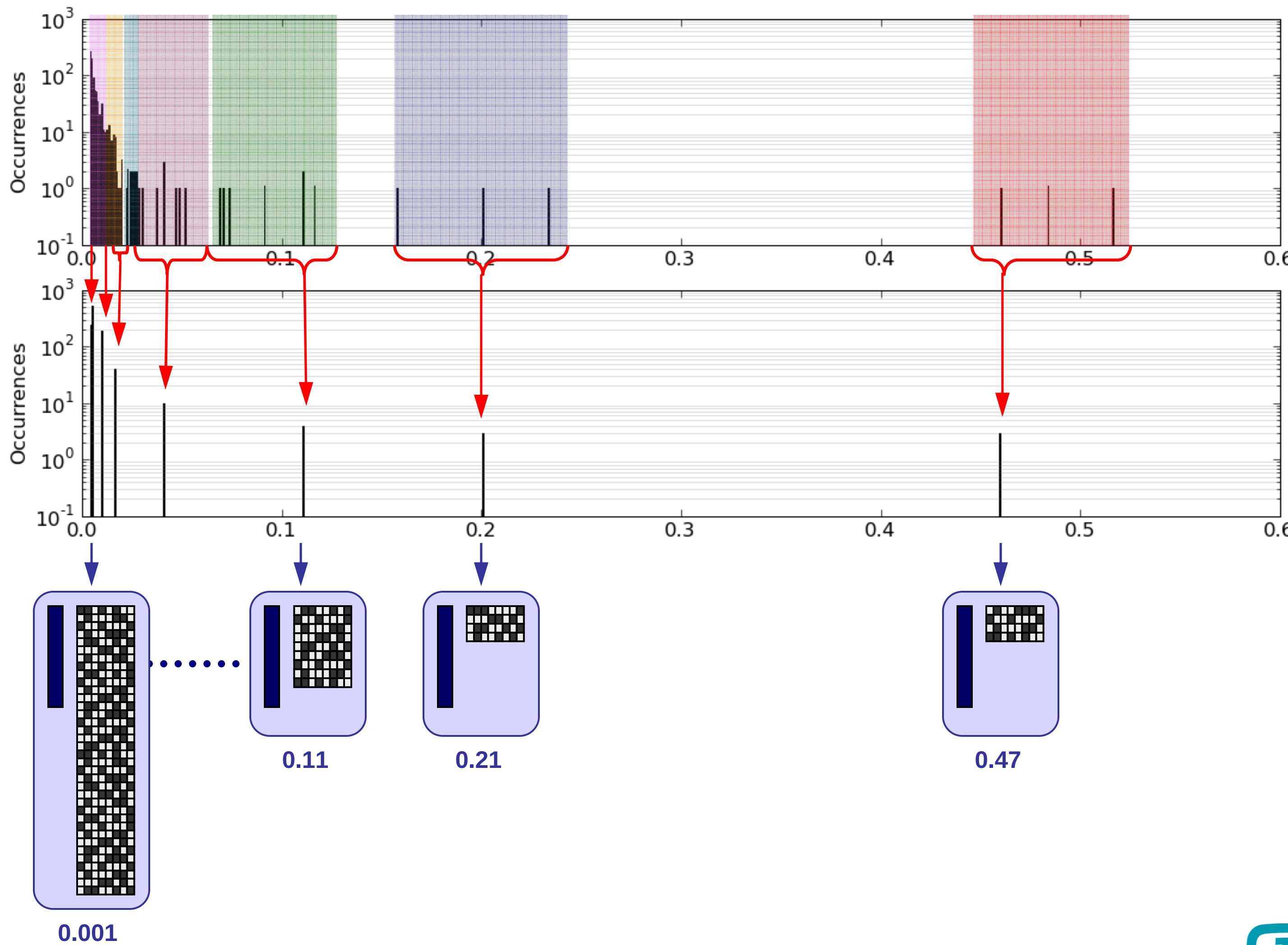
Modification 2: Truncate Dictionary



Modification 3: Re-Quantize Data Values

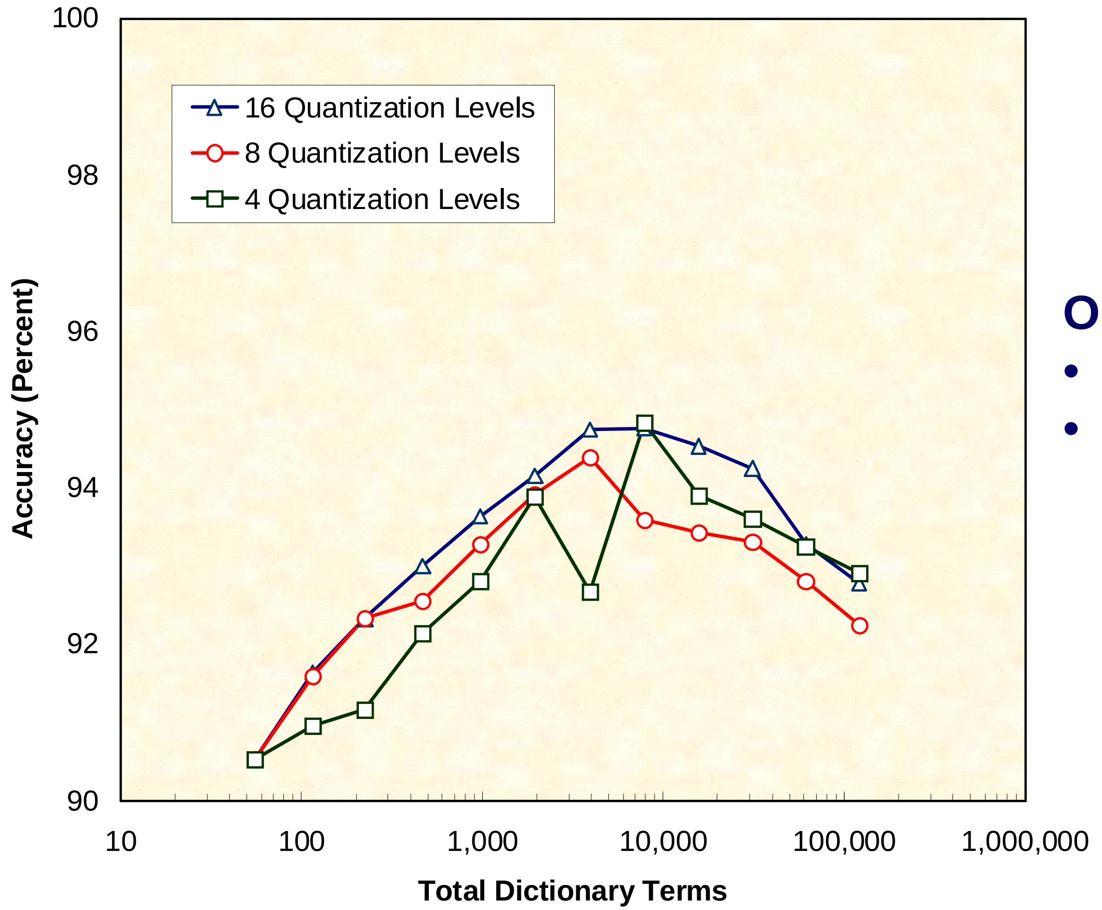


Modification 4: Map to Q Bloom Filters



Bloom Filters

End Impact on Accuracy



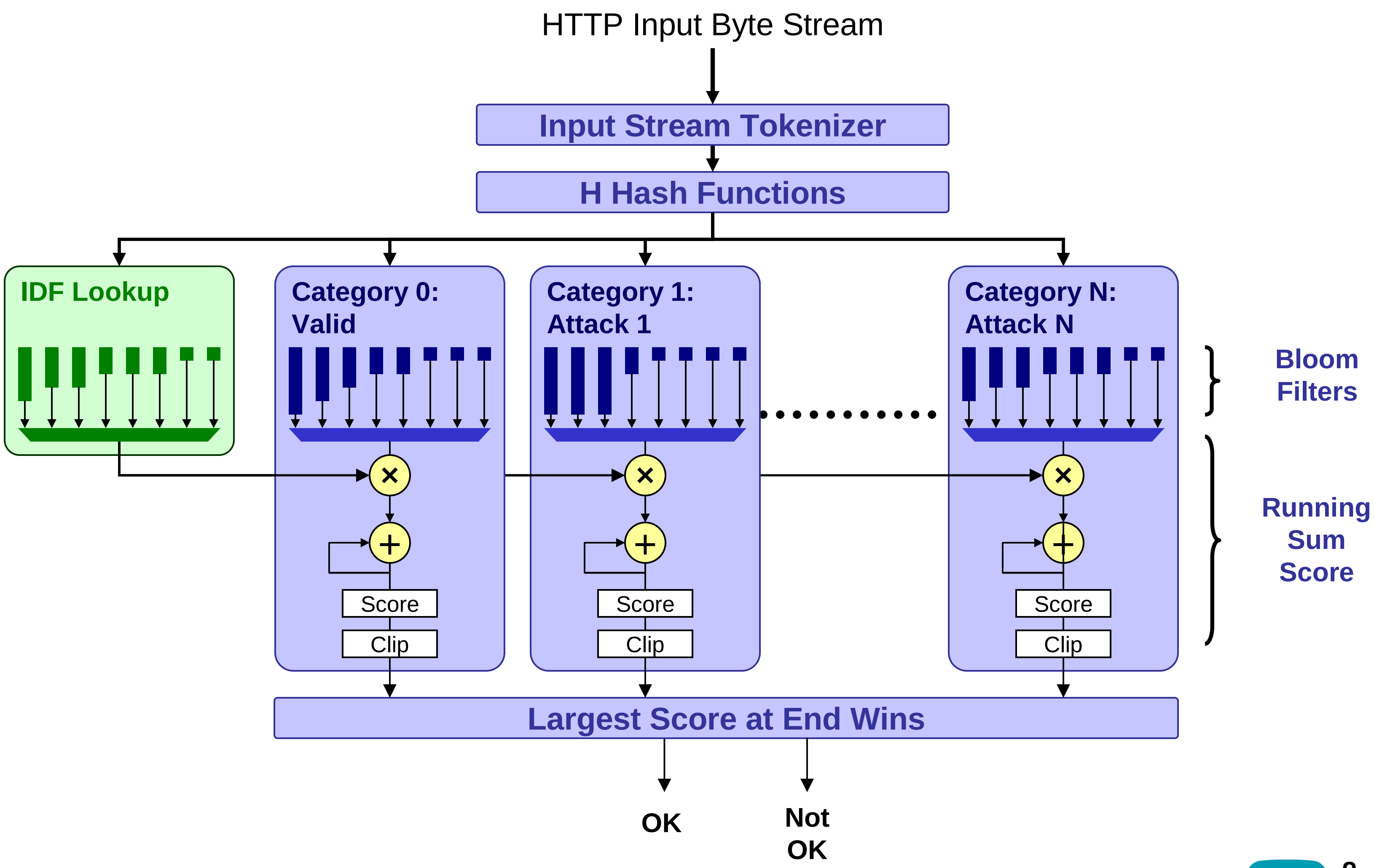
Our Choice:

- 8 quantization levels/classifier
- 4K total terms

Outline

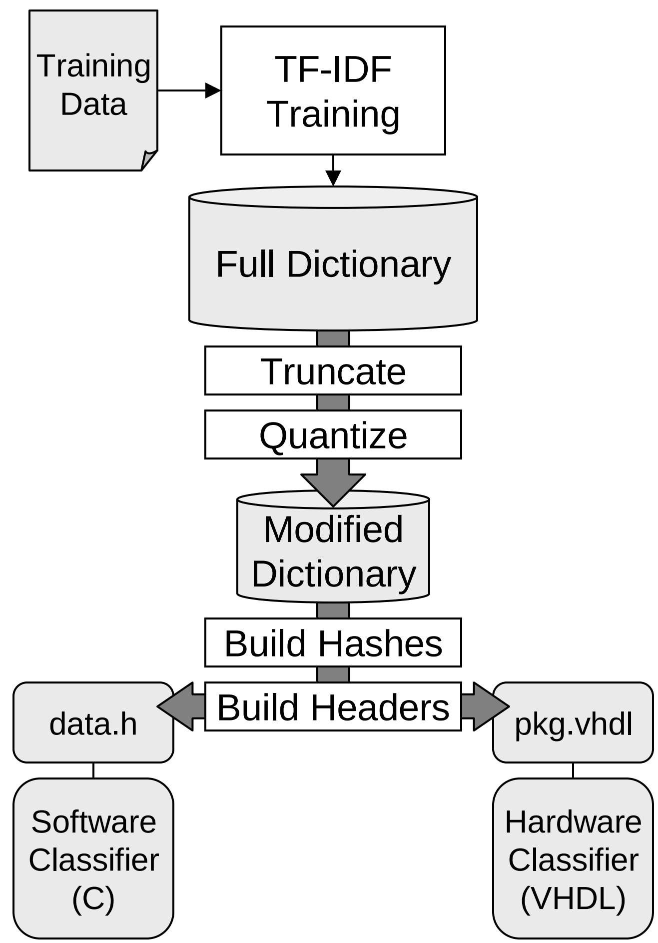
- Introduction
  - Discovery challenge and LLNL approach
- Adapting to embedded hardware
  - Algorithm modifications
  - Data modifications
- Implementation details
  - Core design
  - Tool flow
  - Performance and resource utilization
- Future work

Core Architecture



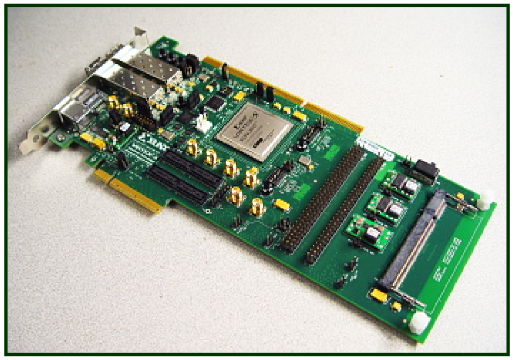
Build Flow

- Desire tools for automatic generation
  - Infrequently rebuild and deploy
  - Utilize in other applications
- User provides
  - Labeled training data
  - Number of dictionary terms
  - Number of hash functions
  - Quantization levels
  - Bloom filter error rate
- Tool chain generates header files
  - C header or VHDL package
  - Static classifier software/hardware
  - Requires a rebuild of design



Sandia National Laboratories

Performance Measurements

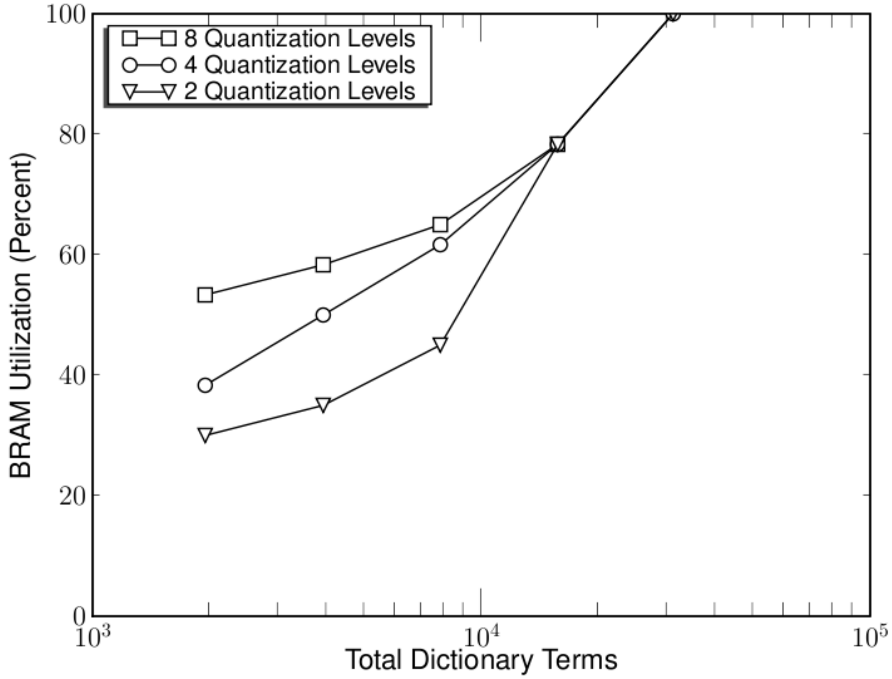


- Built and tested on Xilinx ML555 board
  - Xilinx Virtex5 LX50T -1 FPGA
  - Target GigE speeds (125MB/s)
  - Maximum clock rate: 196MHz
- Bottleneck: Input tokenizing/hashing
  - Byte stream interface
  - Append each token with 2-byte length during hashing
  - Results in extra stall cycle between tokens

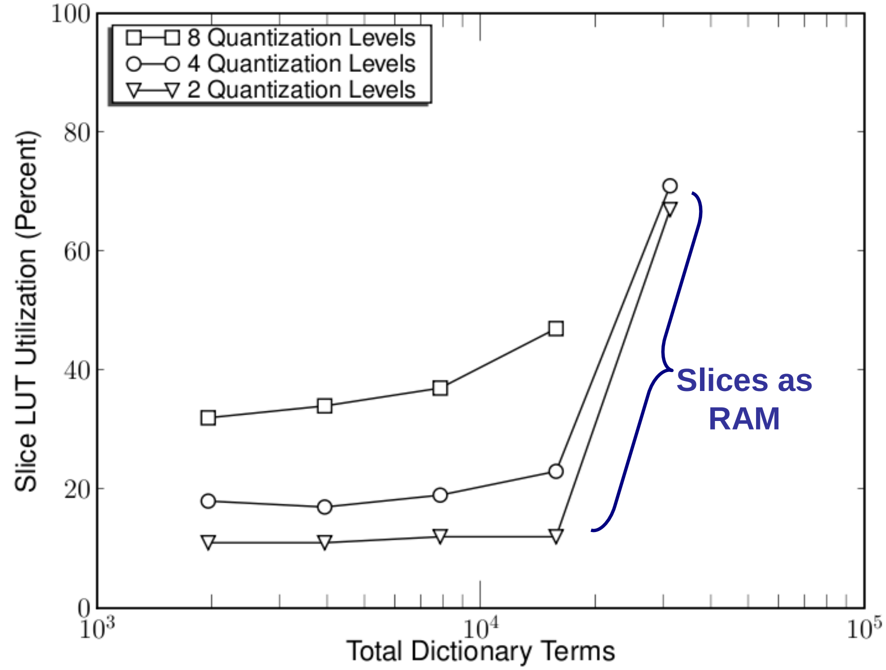
Situation	Efficiency	Rate @ 196MHz
Worst case	0.50	98 MB/s
Best case	0.99	194 MB/s
Average for actual data	0.85	166 MB/s

Resource Utilization

Block RAM Utilization



Slice Utilization



Future Directions

- Architecture improvements
  - Hybrid hashing: employ efficient hash structure for housing one-offs
  - Transition from compile-time data to run-time data
- Additional platforms
  - Tilera: Assign Bloom filters to different processor cores
  - GPU: Possible, but less attractive due to lack of network options
- Application
  - Apply to other data classification applications
  - Continued work in applying data classification techniques

Examining the Algorithm

- Term-Frequency, Inverse Document Frequency
  - TF: How often does each term appear in an attack?
  - IDF: How specific is the term to an attack?

$$tfidf(t,d) = \underbrace{\frac{count(t,d)}{\sum_{v \in d} count(v,d)}}_{Term\ Frequency} \cdot \underbrace{\log \frac{|D|}{|\{d_j : t \in d_j\}|}}_{Inverse\ Document\ Frequency}$$

- Cosine Similarity
  - Vector dot product: estimate angle between input and each attack category

$$\mathrm{sim}_{\cos}(a,R) = \frac{\vec{a} \cdot \vec{R}}{\|\vec{a}\| \cdot \|\vec{R}\|} = \frac{\sum_{t \in a \cap R} tfidf(t,a) \cdot tfidf(t,R)}{\sqrt{\sum_{t \in a} tfidf(t,a)^2} \cdot \sqrt{\sum_{t \in R} tfidf(t,R)^2}}$$