NTNU

Norwegian University of Science and Technology

# Approximate Search and Data Reduction Algorithms

## Research Questions

Kyle Porter
NTNU Gjøvik

# Outline of Presentation

- Introduction:
  - Problems
  - General Goals
- Research Questions
  - Brief theoretical/practical background
  - Methodological approach
- Conclusion

NTNU

# What's the Problem?

- There is too much data to process
  - Been known since 2004 that basic string processing algorithms are insufficient.
  - Backlogs of digital evidence awaiting analysis has real world consequences.
- It is difficult to defend against the variety of network attacks.
  - Current approximate matching techniques produce too many false positives.
  - Knowledgeable attackers can generally bypass IDS

# Goals

- Improve accuracy of approximate search techniques
  - Return more reliable approximate search results
- Build on and improve data reduction techniques.
  - Have a competent method of analyzing data without needing close examination.
  - Improvements in speed, memory consumption, accuracy are all welcome.
- Primary development for Big Data analysis and IDS.

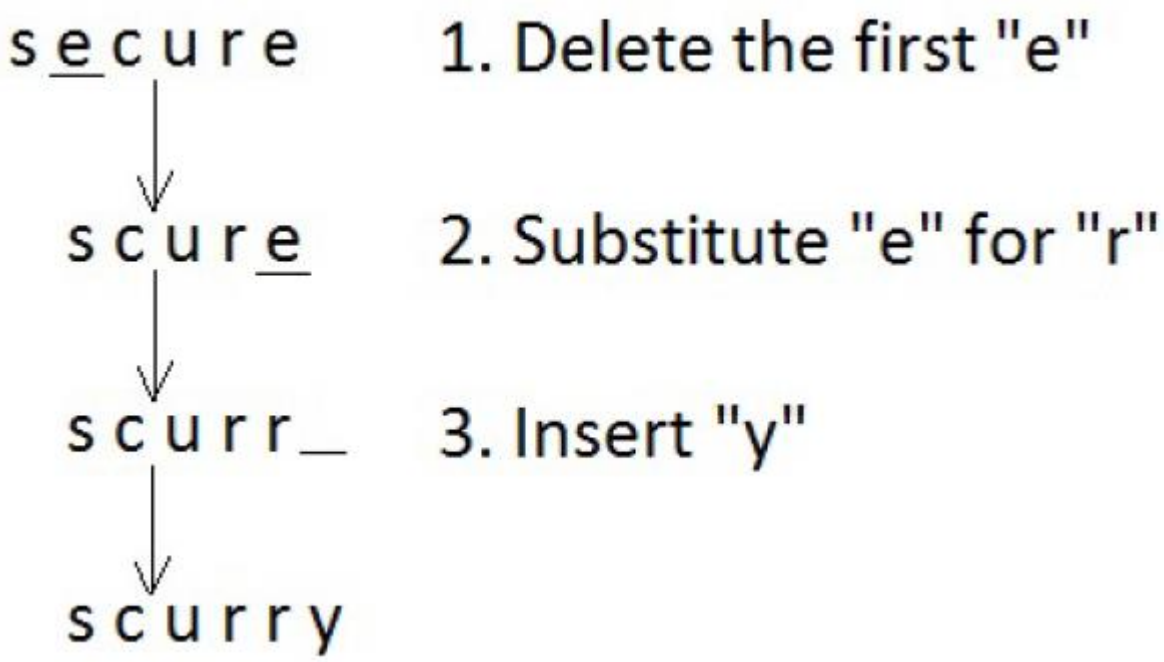
# Research Question 1

- How can we implement constrained edit operations into approximate string matching in an efficient way supported by theory, and how can we extend existing algorithms to support constrained edit distance?

# RQ1 Background

- Approximate string matching problem:
  - find pattern p in text T such that p and some substring x of T approximately resemble each other.
- Reason for large number of inaccuracies is due to the resemblance metric.
- Levenshtein (edit) distance: minimum number of insertions, deletions, substitutions necessary to transform one string into another.
- The neighborhood of possible matches can be large.
  - E.g. For allowed edit distance of 3, the word “secure” approximately matches “scurry”.

# String Transformation Example



# RQ1: Background

- We propose use of *constrained edit distance*.
  - Each edit operation is constrained.
  - The distance between strings is measured by the minimum number of allowed edit operations given the constraints.
- E.g. If no insertions allowed, one deletion, and two substitutions are allowed, then “secure” does not approximately match “scurry” under the constraints.
- The matching neighborhood has been reduced to an area defined by the constraints.
- Motivation: if you have a priori knowledge of expected errors/obfuscation, then you can obtain more accurate results.

# RQ1: Methodology

- Develop Hypotheses
- State-of-the-art approximate matching algorithms primarily use two theoretical :
  - Dynamic Programming Matrices
    - Flexibility with metrics
  - Deterministic and Nondeterministic finite automata
    - DFA's faster, run in linear time, but have exponential memory consumption.
    - NFA's are often easier to design, far fewer necessary states, slower since they must be simulated.

# Research Question 1.a

- How can we increase the efficiency of any approximate string matching algorithms we create by utilizing existing techniques?

# RQ 1.a Methodology

- Bit-parallelism
  - Simulate nondeterministic finite automata
  - Test all possible edit operations of each pattern character in parallel.
- Filtering
  - Skip text
- Dynamic Programming speedups.

# Research Question 2

- How could constrained approximate search be effectively realized in various kinds of hardware?

# RQ2: Methodology

- Multi-pattern search algorithms have been implemented into specialized hardware (ASIC, FPGA, GPU) with very good results.
- Actual implementation into hardware will likely a require a partner.
- Item of interest is bit-splitting implementation.
  - Far more scalable methodology (w.r.t memory)
  - Can be applied to general state machines

# Testing Algorithms

- For any algorithm we create:
  - Perform an average and worse case time and memory complexity analysis.
  - Perform tests with different character sets, edit constraints, pattern lengths, and text corpora.
  - Compare results with state-of-the-art.
- Important data:
  - Accuracy
  - Time consumption
  - Memory Consumption

# Research Question 3

- How can we reduce the size of data processed by these research algorithms and preserve the similarity between the data objects at the same time?

# RQ3: Background

- Similarity-preserving hash functions, or fuzzy hashes.
- Similar in use to cryptographic hashes, but no avalanche effect.
  - For similar inputs m and n into the fuzzy hash function, the output x and y will also be very similar.
- Goals:
  - Identify that two digital artifacts resemble each other
  - Embedded object detection
  - Detect traces of known artifact
  - Detect if two artifacts share a common object.

# RQ3 Background

- Output of a fuzzy hash is called a *sketch.*
  - This is a feature vector.
- Comparisons of sketches typically compare each feature, and return a binary yes/no match result.
- Hamming distance or Levenshtein distance often used for determining similarity.
- Levels of abstraction:
  - Byte-wise
  - Syntactic
  - Semantic

# RQ3 Methodology

- Study the existing methodology and look for potential areas of improvement:
  - Context triggering piecewise hashing and rolling hashes.
  - Use of Shannon Entropy
- Look for practical non-cryptographic hash functions, as well as other potential methodologies.
- Use existing framework to test quality of any produced fuzzy hash algorithms
  - Tests processing time, comparison time, resistance to noise, calculate DET curves, false positive rates, false negative rates, etc.

# Research Question 4

- How does digital forensics (Big Data analysis and intrusion detection) benefit from utilizing constrained edit distance approximate search and similarity-preserving hash functions?

# RQ4 Methodology

- Results from first three RQs will partially answer this.
- Interview digital forensic analysts.
- Test algorithms using the Hansken Digital Forensics as a Service system once available for testing.

# Conclusion

- Improved accuracy of approximate string matching algorithms for Big Data analysis and Intrusion Detection.
- Improved overall quality of fuzzy hashing (data reduction) algorithms for Big Data analysis.
- Current Projects:
  - Develop paper for new CED algorithm
  - Interview digital forensic analysts
  - Work with Fuzzy Hash Algorithms