A Theoretic Framework for Evaluating Similarity Digesting Tools

LIWEI REN, PH.D, Trend Micro™

DFRWS EU 2015, Dublin, Ireland, March, 2015
Agenda

• Byte-wise Approximate Matching
• Similarity Digesting Tools
• Mathematical Models for Byte-wise Similarity
• Tool Evaluation with Theoretic Analysis
• Tool Evaluation with Data Experiment
• Further Research for Approximate Matching
Byte-wise Approximate Matching

- **Byte-wise similarity & approximate matching.**
  - What is byte-wise similarity?

- **4 Use Cases specified by NIST:**

  ![Image](image1)

  ![Image](image2)

  ![Image](image3)

  ![Image](image4)
Byte-wise Approximate Matching

(3) Containment

(4) Fragmentation
Similarity Digesting Tools

- **Similarity digesting**: A class of hash techniques or tools that preserve similarity.
- Typical steps for digest generation:

- Detecting similarity with similarity digesting:

- **Three similarity digesting algorithms and tools:**
  - ssdeep, sdhash & TLSH
Similarity Digesting Tools

- **ssdeep**
  - Two steps for digesting:

  1. **First step (STEP 1):**
     - 1st chunk
     - 2nd chunk
     - 3rd chunk

  2. **Second step (STEP 2):**
     - Concatenate

- **Edit Distance:** Levenshtein distance
Similarity Digesting Tools

- **sdhash**

  - Two steps for digesting:

    1. **STEP 1**
       - 1st 64-byte feature with high entropy
       - 2nd 64-byte feature with high entropy
      
      Bloom filtering

    2. **STEP 2**
       - Last 64-byte feature with high entropy
       - 256 byte bitmap or longer
    
    Bloom filtering

- **Edit Distance**: Hamming distance
Similarity Digesting Tools

- **TLSH**
  - Two steps for digesting:

  **STEP 1**
  
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>A</td>
<td>B</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>A</td>
<td>B</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>A</td>
<td>C</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>A</td>
<td>C</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>A</td>
<td>D</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td></td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>8.</td>
<td></td>
<td>B</td>
<td>C</td>
<td>E</td>
</tr>
<tr>
<td>9.</td>
<td></td>
<td>B</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>10.</td>
<td></td>
<td>C</td>
<td>D</td>
<td>E</td>
</tr>
</tbody>
</table>

**STEP 2**

- **Every 5-gram**

- **Selecting 6 triplets**

- **Edit Distance**: A diff based evaluation function
Mathematical Models for Byte-wise Similarity

• **Summary of Three Similarity Digesting Schemes:**
  
  − Using a **first model** to describe a binary string with *selected features*:
    - **ssdeep model**: a string is a *sequence* of *chunks* (split from the string).
    - **sdhash model**: a string is a *bag* of *64-byte blocks* (selected with entropy values).
    - **TLSH model**: a string is a *bag* of *triplets* (selected from all 5-grams).
  
  − Using a **second model** to map the selected features into a digest which is able to preserve similarity to certain degree:
    - **ssdeep model**: a sequence of chunks is mapped into a 80-byte digest.
    - **sdhash model**: a bag of blocks is mapped into one or multiple 256-byte bloom filter bitmaps.
    - **TLSH model**: a bag of triplets is mapped into a 32-byte container.
Mathematical Models for Byte-wise Similarity

- Three approaches for similarity evaluation:

  - Direct Comparison
  - Feature comparison
  - Digest Comparison

- 1\textsuperscript{st} model plays critical role for similarity comparison.
  - Let focus on discussing various 1\textsuperscript{st} models today.
  - Based on a unified format.

- 2\textsuperscript{nd} model saves space but further reduces accuracy.
Mathematical Models for Byte-wise Similarity

- **Unified format for 1st model:**
  - A string is described as a collection of tokens (aka, features) organized by a data structure:
    - ssdeep: a sequence of chunks.
    - sdhash: a bag of 64-byte blocks with high entropy values.
    - TLSH: a bag of selected triplets.
  - Two types of data structures: sequence, bag.
  - Three types of tokens: chunks, blocks, triplets.

- **Analogical comparison:**
  - Model to Language
  - Tokens to Words
  - Data Structure to Grammar
  - Describing similarity to Describing ideas
Mathematical Models for Byte-wise Similarity

- **Four general types of tokens from binary strings:**
  - *k*-grams where \( k \) is as small as 3, 4, ...
  - *k*-subsequences: any subsequence with length \( k \). The triplet in TLSH is an example.
  - **Chunks**: whole string is split into non-overlapping chunks.
  - **Blocks**: selected substrings of fixed length.

- Eight different models to describe a string for similarity.

- Analogical thinking:
  - we define different distances to describe a metric space.
Tool Evaluation with Theoretic Analysis

• **Data Structure:**
  - **Bag:** a bag ignores the order of tokens. It is *good at handling content swapping*.
  - **Sequence:** a sequence organizes tokens in an order. This is *weak for handling content swapping*.

• **Tokens:**
  - **k-grams:** Due to the small k (3, 4, 5, ...), this fine granularity is *good at handling fragmentation*.
  - **k-sequences:** Due to the small k (3, 4, 5, ...), this fine granularity is *good at handling fragmentation*.
  - **Chunks:** This approach takes account of every byte in raw granularity. It should be *OK at handling containment and cross sharing*.
  - **Blocks:** Depending on different selection functions, even though it does not take account of every byte, but it may present a string more efficiently and that is good for generating similarity digests. Due to the nature of fixed length blocks, it is *good at handling containment and cross sharing*. 
## Tool Evaluation with Theoretic Analysis

<table>
<thead>
<tr>
<th>Tool</th>
<th>Model</th>
<th>Minor Changes</th>
<th>Containment</th>
<th>Cross sharing</th>
<th>Swap</th>
<th>Fragmentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssdeep</td>
<td>M1.3</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>sdhash</td>
<td>M2.4</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>TLSH</td>
<td>M2.2</td>
<td>High</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Sdhash + TLSH</td>
<td>Hybrid</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
</tbody>
</table>
# Tool Evaluation with Data Experiment

<table>
<thead>
<tr>
<th>Purpose of Tests</th>
<th>Edit Operations</th>
<th>ssdeep</th>
<th>sdhash</th>
<th>TLSH</th>
<th>ssdeep</th>
<th>sdhash</th>
<th>TLSH</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Containment</strong></td>
<td>Cut 30% at the beginning</td>
<td>82</td>
<td>60</td>
<td>31</td>
<td>79</td>
<td>89</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>Cut 60% at the end</td>
<td>54</td>
<td>100</td>
<td>x</td>
<td>58</td>
<td>99</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Cut 90% at the beginning</td>
<td>x</td>
<td>77</td>
<td>x</td>
<td>x</td>
<td>100</td>
<td>x</td>
</tr>
<tr>
<td><strong>Cross sharing</strong></td>
<td>Substitute 30% at the end</td>
<td>72</td>
<td>70</td>
<td>69</td>
<td>75</td>
<td>59</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>Substitute 60% in the end</td>
<td>47</td>
<td>40</td>
<td>54</td>
<td>47</td>
<td>37</td>
<td>62</td>
</tr>
<tr>
<td></td>
<td>Substitute 90% at the end</td>
<td>29</td>
<td>10</td>
<td>47</td>
<td>x</td>
<td>6</td>
<td>42</td>
</tr>
<tr>
<td><strong>Swap</strong></td>
<td>Swap with 2-1</td>
<td>52</td>
<td>71</td>
<td>99</td>
<td>54</td>
<td>68</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>Swap with 4-3-2-1</td>
<td>36</td>
<td>59</td>
<td>98</td>
<td>33</td>
<td>54</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>Swap with 8-7-6-5-4-3-2-1</td>
<td>32</td>
<td>62</td>
<td>99</td>
<td>x</td>
<td>48</td>
<td>96</td>
</tr>
<tr>
<td><strong>Fragmentation</strong></td>
<td>Modify the bytes at 64*j</td>
<td>x</td>
<td>x</td>
<td>58</td>
<td>x</td>
<td>x</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>Modify the bytes at 128*j</td>
<td>x</td>
<td>x</td>
<td>78</td>
<td>x</td>
<td>x</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>Modify the bytes at 256*j</td>
<td>x</td>
<td>15</td>
<td>86</td>
<td>x</td>
<td>33</td>
<td>82</td>
</tr>
<tr>
<td><strong>Minor changes</strong></td>
<td>Swap with 1-2-3-4-5-7-6-8-9-10-11-12-13-14-15-16. Subst 1% at the end. Cut 1% at the beginning.</td>
<td>90</td>
<td>88</td>
<td>93</td>
<td>83</td>
<td>93</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>Swap with 1-2-3-5-4-6-7-8-9-10-11-12-13-14-15-16. Cut 2% in the beginning. Subst 1% at the end.</td>
<td>91</td>
<td>85</td>
<td>92</td>
<td>82</td>
<td>82</td>
<td>86</td>
</tr>
</tbody>
</table>
Further Research for Approximate Matching

- A Roadmap for Further Research:

  - NIST use cases
  - Relationship: byte-wise relevance
  - 6 use cases
  - Extended to Define

  - Matching
    - Identicalness
    - Containment
    - Cross-sharing
    - Similarity
    - Approx. Containment
    - Approx Cross-sharing
  - RK Algorithm
  - Searching
    - Proposal of various searching problems
  - Clustering
    - To be explored
  - Fingerprinting Algorithms
  - Solutions for searching problems
  - Math models for describing byte-wise similarity
  - Evaluate
    - TLSH
    - sdhash
    - ssdeep
  - To be solved

Today’s topic
Q&A

• Thank you for your interest.

• Any questions?

• My Contact Information:
  – Email: liwei_ren@trendmicro.com
  – Linkedin: https://www.linkedin.com/in/drliweiren
  – Academic Page: https://pitt.academia.edu/LiweiRen