Approximate string matching aka fuzzy string searching

Ladislav Šulák (laco.sulak@gmail.com)

Krisztian Benko (kristianbnk@gmail.com)

Brno University of Technology, Faculty of Information Technology

Fuzzy string matcher

- Matching string approximately
- Bitap algorithm – based on Levenstein distance, but it uses only substitution operation
- Number of mistakes (incorrect letters in a given pattern) can be set and it affects performance of algorithm

Naive string matcher

- Exact string matching
- It cycles an input text a letter by letter and it searches for the same beginning of a searched word
- There are faster variants regarding exact string matching algorithms like KMP (Knuth–Morris–Prat)

Achievements and outcomes

Both exact (naive) and approximate (fuzzy) matching:

- finding a position of a first occurence of a searched word in a given text
- finding all the positions of occurrences of a searched word in a given text
- marking all found words from a given text

Spell corrector (only fuzzy approach):

- it corrects an input word if there is a typo - substitution of a letter/s by another one/s
- correction is based on a given text (not external dict)

Performance & Benchmarking I

1. Benchmark tests were performed on 25,000 pairs of (word, text) where:
   - word is a searched word of size between 4 and 31 letters *(6,6 in average)*,
   - text contain at least 3 words and it is a searched area which always contains words in English written correctly *(121,1 letters in average)*

2. Each resulting runtime was calculated by performing 5 runs and calculating the average time

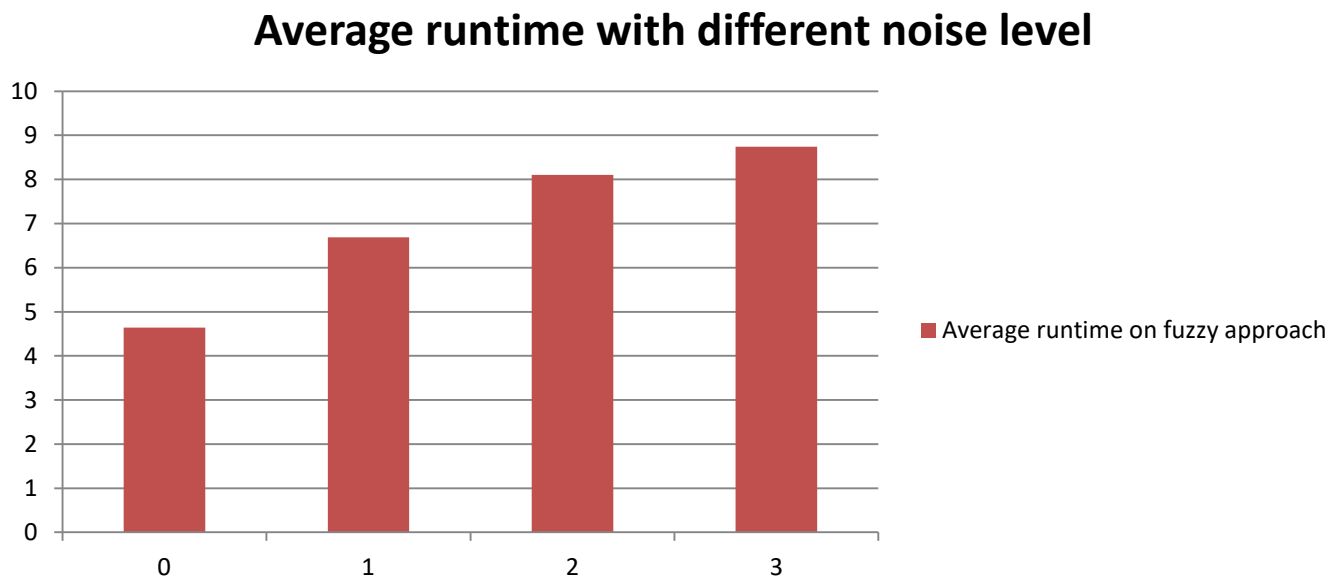
3. Dataset generated automatically with pseudo-random generator (normal distribution) and some pairs can be duplicated.

4. Data source: *The Project Gutenberg EBook of The Adventures of Sherlock Holmes by Sir Arthur Conan Doyle*

Performance & Benchmarking II

- *Noise level 0, searching for all occurences:*
  - *FuzzyStringMatcher*: 4,7409755229949952 seconds
  - *NaiveStringMatcher:* 1,73054976463317872 seconds

- Various noise level, fuzzy searching for all occurrences:



References

Fuzzy Bitap Algorithm:

- *https://www.programmingalgorithms.com/algorithm/fuzzy-bitap-algorithm*

Naive string searching:

- *https://www.geeksforgeeks.org/searching-for-patterns-set-1-naive-pattern-searching/*

Description and some basic info:

- *https://en.wikipedia.org/wiki/String_searching_algorithm*
- https://en.wikipedia.org/wiki/Approximate_string_matching
- https://en.wikipedia.org/wiki/Bitap_algorithm
- https://en.wikipedia.org/wiki/Levenshtein_distance