Information Retrieval
WS 2016 / 2017

Lecture 5, Tuesday November 22\textsuperscript{nd}, 2016
(Fuzzy Search, Edit Distance, q-Gram Index)

Prof. Dr. Hannah Bast
Chair of Algorithms and Data Structures
Department of Computer Science
University of Freiburg
Overview of this lecture

- Organizational
  - Experiences with ES4 Compression, Codes, Entropy

- Contents
  - Fuzzy search type breifurg, find freiburg
  - Edit Distance a standard similarity measure
  - Q-gram Index index for efficient fuzzy search

Exercise Sheet 5: implement error-tolerant prefix search using a q-gram index and prefix edit distance
Experiences with ES4  1/3

- Summary / excerpts
  - Some liked it, for some it was OK, some didn't like it
    "Very elegant explanations ... no problems with exercises"
    "Some natural frustration ... but an enjoyable challenge"
    "Did not enjoy ... don't like mathematical proofs a lot"
  - Very helpful to understand the concepts from the lecture
  - Help in the forum was much appreciated
  - Looking forward to the master solution (it's there!)
  - Looking forward to coding exercises again
  - Entropy of human DNA is 7.13 on average according to
    https://www.hindawi.com/journals/mpe/2012/132625/tab1
Proof sketch for Exercise 4.2

- Show that Gollum is optimal for $p_x = (1 - p)^x - 1 \cdot p$

1. $p_x = (1 - p)^x - 1 \cdot p = (1 - p)^x \cdot \frac{p}{1 - p} \leq e^{-px} \cdot \frac{p}{1 - p}$

   $\log_2 \frac{1}{px} \geq \frac{\log_2 e^{+px}}{px/\ln 2} + \log_2 \frac{1}{p} + \log_2 (1 - p) \geq -1$

   $M = \left\lceil \frac{\ln 2}{p} \right\rceil \geq \frac{\ln 2}{p}$

2. $L_x = \lfloor x/M \rfloor + \lceil \log_2 M \rceil + 1 \leq x/M + \log_2 M + 2$

   $\leq \frac{px}{\ln 2} + \log_2 \frac{1}{p} + \frac{\log_2 \ln 2 + 3}{2} \leq 0$

3. $L_x \leq \log_2 \frac{1}{px} + 4$

4. $E(L_x) = \sum x \cdot p_x \cdot L_x \leq \sum x \cdot p_x \cdot \frac{\log_2 1}{px} + \sum x \cdot p_x \cdot 4$
Your DNA

- The nucleotides of your DNA are asymmetric, with a phosphate group attached to the 5' side of the ring.
- Synthesizing only works in the 5'-to-3' direction, because making bonds in that direction is more energy efficient.
- However, if one strand of DNA goes in the 5'-to-3' direction, the other must go in the 3'-to-5' direction.
- So how does the cell manage to copy both strands?
  The answer is quite amazing.
- You are quite a machine ... on the biomolecular level.
- More about that on future sheets.
Problem setting

- Given a "dictionary" = a list of "names" of any kind

  For ES5, a list of 181,296 cities in Western Europe

- For a given query, find matching names from that dict.

  Query: frei  Match: freiburg  \textbf{prefix} search
  Query: fr*rg  Match: freiburg  \textbf{wildcard} search
  Query: breifurg  Match: freiburg  \textbf{fuzzy} search

- Similar challenges as for our search so far:

  Challenge 1: good model of what \textbf{matches}

  Challenge 2: preprocess the input (= build a suitable index), so that we find the matching names \textbf{fast}
Possible origins for the dictionary

- Popular queries extracted from a query log
  
  Basis for Google's query-suggestion feature

- Words + common phrases from a text collection

  Extracting common phrases from a given text collection is an interesting problem by itself, however, not one we will deal with in this course

- A list of names of entities

  For example: person names, movie titles, places, street addresses, ...
Combining matching and search

- One could simply search for the top match, for example:
  
  Type: freib Search: freiburg

- Or one could search for several matches

  Type: freib Search: freiburg OR freibach OR ... OR ...

- In today's lecture, we will only look at the problem of finding matching names in a list of names

  The search part is also interesting when the number of matching strings is very large; then a simple OR of a lot of strings will be too slow and we need better solutions
Simple solution

- Iterate over all strings in the dictionary, and for each check whether it matches

- This is what the Linux commands `grep` and `agrep` do

  ```
  grep -x uni.* <file>
  grep -x un.*ity <file>
  agrep -x -2 univerty <file>
  ```

  All matching lines in `<file>` will be output

  The option `-x` means match whole line (not just a part)

  The option `-2` means allow up to two "errors" ... next slide
Simple solution, check match of single string

- Given a query $q$ and a string $s$

- **Prefix search:** easy-peasy

  Just compare $q$ and the first $|q|$ characters of $s$ ... can be accelerated by finding the first match with a binary search

- **Wildcard search:** also easy if only one $\ast$

  If $q = q_1 \ast q_2$, check that $|s| > |q_1| + |q_2|$ and then compare the first $|q_1|$ characters of $s$ with $q_1$ and the last $|q_2|$ characters of $s$ with $q_2$

- **Fuzzy search:** more complicated

  Compute edit distance between $q$ and $s$ ... slides 11 – 16
Simple solution, time complexity

- The time complexity is obviously $n \cdot T$, where
  
  $n = \#\text{records}$, $T = \text{time for checking a single string}$

- For fuzzy search, $T \approx 1\mu\text{s} \ldots \text{find out yourself in ES5}$

- In search, we always want interactive query times

  Respond times feel interactive until about $100\text{ms}$

- So the simple solution is fine for up to $\approx 100K$ records

- For larger input sets, we need to pre-compute something

  We will build a q-gram index \ldots\ slides 20 – 26
Definition ... aka Levenshtein distance, from 1965

- Definition: for two strings $x$ and $y$

  $\text{ED}(x, y) := \text{minimal number of tra'fo's to get from } x \text{ to } y$

- Transformations allowed are:

  insert($i$, $c$) : insert character $c$ at position $i$

  delete($i$) : delete character at position $i$

  replace($i$, $c$) : replace character at position $i$ by $c$

$x = \text{DOOF} \rightarrow \text{REPLACE(1, B)}$

$\text{BOOF} \rightarrow \text{REPLACE(2, L)}$

$\text{BLOOF} \rightarrow \text{INSERT(4, E)}$

$y = \text{BLOED} \rightarrow \text{REPLACE(5, D)}$

$\text{This just proves } \text{that } \text{ED}(x, y) \leq y$. 
Some simple notation

- The empty word is denoted by $\varepsilon$
- The length (number of characters) of $x$ is denoted by $|x|$
- Substrings of $x$ are denoted by $x[i..j]$, where $1 \leq i \leq j \leq |x|$

Some simple properties

- $ED(x, y) = ED(y, x)$
- $ED(x, \varepsilon) = |x|$
- $ED(x, y) \geq \text{abs}(|x| - |y|)$
- $ED(x, y) \leq ED(x[1..n-1], y[1..m-1]) + 1$  
  \[ n = |x|, \ m = |y| \]
Recursive formula

- For $|x| > 0$ and $|y| > 0$, $ED(x, y)$ is the minimum of
  
  (1a) $ED(x[1..n], y[1..m-1]) + 1$
  
  (1b) $ED(x[1..n-1], y[1..m]) + 1$
  
  (1c) $ED(x[1..n-1], y[1..m-1]) + 1$ if $x[n] \neq y[m]$
  
  (2) $ED(x[1..n-1], y[1..m-1])$ if $x[n] = y[m]$

- For $|x| = 0$ we have $ED(x, y) = |y|$

- For $|y| = 0$ we have $ED(x, y) = |x|$

For a proof of that formula, see e.g. Algorithmen und Datenstrukturen SS 2015, Lecture 11a, slides 18 – 23
Algorithm for computing $\text{ED}(x, y)$

- The recursive formula from the previous slide naturally leads to the following dynamic programming algorithm

- Takes time and space $\Theta(|x| \cdot |y|)$
Prefix edit distance

- The prefix edit distance between $x$ and $y$ is defined as
  \[ \text{PED}(x, y) = \min_{y'} \text{ED}(x, y') \] where $y'$ is a prefix of $y$

- For example
  \[
  \begin{align*}
  \text{PED}(\underline{uni}, \underline{university}) &= 0 \quad \text{... but ED = 7} \\
  \text{PED}(\underline{uniwer}, \underline{university}) &= 1 \quad \text{... but ED = 5}
  \end{align*}
  \]

- Important for fuzzy search-as-you type suggestions

  By now, all the large web search engines have this feature, because it is so convenient for usability
Computation of the PED

- Compute the entries of the $|x| \cdot |y|$ table, just as for ED
- The PED is just the minimum of the entries in the last row
- Important optimization: when $|x| << |y|$ and you only want to know if $\text{PED}(x, y) \leq \delta$ for some given $\delta$:

Enough to compute the first $|x| + \delta + 1$ columns ... verify!

Let's say $\delta = 2$

Then you don't have to look beyond this point.

Minimum of those $= \text{PED}(\text{FIBU}, \text{FREIBURG}) = 2$
Definition of a q-gram

- The q-grams of a string are simply all substrings of length q
  freiburg: fre, rei, eib, ibu, bur, urg

The number of q-grams of a string x is exactly |x| - q + 1

- For fuzzy search, we will **pad** the string with q – 1 special symbols (we use $) in the beginning and in the end
  freiburg → $$freiburg$$

3-grams: $$f, fr, fre, rei, eib, ibu, bur, urg, rg$$, g$$

The number is then |x| + q – 1, where x is the original string

We will see in a minute, why that padding is useful
Definition of a q-gram index

- For each q-gram store an inverted list of the strings (from the input set) containing it, sorted lexicographically

\[ \text{fr} : \text{fraberg, frallach, freiburg, freiberg, frouville, ...} \]
\[ \text{ibu} : \text{biburg, freiburg, garcibuey, seibuttendorf, ...} \]

As usual, store ids of the strings, not the strings themselves

Note: very similar to an inverted index, just with q-grams instead of words

Let's adapt our code from Lecture 1 to q-grams
Space consumption

- Each record $x$ contributes $|x| + O(1)$ ids to the inverted lists.
- The total number of ids in the lists is hence about the number of **characters** (not words) in the dictionary.
- If we use 4 bytes per id, the index would hence be at least four times bigger than the original dictionary.
- This can be reduced significantly using **compression**

For ES5, it is fine to store the lists uncompressed.
Fuzzy search with a q-gram index, using ED

- Consider $x$ and $y$ with $ED(x, y) \leq \delta$
- Intuitively: if $x$ and $y$ are not too short, and $\delta$ is not too large, they will have one or more q-grams in common
- Example: $x = \text{HILLARY}$, $y = \text{HILARI}$

\[
\begin{align*}
\text{HILLARY} & \rightarrow \underline{H}, \underline{HI}, \underline{HIL}, \underline{ILL}, \underline{LLA}, \underline{LAR}, \underline{ARY}, \underline{RY}, Y \\
\text{HILARI} & \rightarrow \underline{H}, \underline{HI}, \underline{HIL}, \underline{ILA}, \underline{LAR}, \underline{ARI}, \underline{RI}, I
\end{align*}
\]

number of q-grams in common = 4

Note: the padding in the beginning gives us two additional 3-grams in common (because no mistake in first letter)
Fuzzy search with a q-gram index, using ED

- Formally: let $x'$ and $y'$ be the padded versions of $x$ and $y$
  Then: $\text{comm}(x', y') \geq \max(|x|, |y|) - 1 - (\delta - 1) \cdot q$

Example from slide before: $|x| = 7$, $|y| = 6$, $\delta = 2$, $q = 3$
Hence $\text{comm}(x', y') \geq 3$ ... and in the example $\text{comm} = 4$
Verify: in the worst case, $\text{comm}(x', y') = 3$ can happen

- Proof: consider the longer string, which has $\max(|x|, |y|) + q - 1$ q-grams ... because of the left and right $\$ padding
Then one tra'fo (insert / delete / replace) changes at most $q$ q-grams, and hence $\delta$ tra'fos affect at most $\delta \cdot q$ q-grams
Query algorithm, using ED (for PED: analogous)

- Given a query $x$ and a q-gram index for the input strings
- Compute q-grams of $x'$ and fetch their inverted lists
  
  For example: $x = HILARI$, $x' = $$$HILARI$$$

  Fetch lists for: $$H, $HI, HIL, ILA, LAR, ARI, RI$$, I$$$

- Merge these lists and keep track of which record contains how many q-grams ... see TIP file on the Wiki

- For each record $y$ in the merge results, check whether the count is $\geq \max(|x|, |y|) - 1 - (\delta - 1) \cdot q$
  
  If no: discard this $y$, we know that $ED(x, y) > \delta$
  
  If yes: compute $ED(x, y)$ and check if $ED(x, y) \leq \delta$
Fuzzy prefix search

- Use the same algorithm, but with a different bound
- Assume that $\text{PED}(x, y) \leq \delta$
- Let $x'$ and $y'$ be $x$ and $y$ with $q - 1$ times $\$ to the left only.

  Padding on the right makes no sense for prefix search
- Then we have: $\text{comm}(x', y') \geq |x| - q \cdot \delta$

  Note that for $\delta = 1$, this is $\geq 1$ only for $|x| > q$
- **Proof:** Consider $x$, which has exactly $|x|$ $q$-grams.

  Then one tra'fo (insert / delete / replace) changes at most $q$ $q$-grams, and hence $\delta$ tra'fos change at most $\delta \cdot q$ $q$-grams.
References

- **Textbook**
  
  Section 3: Tolerant Retrieval, in particular:
  
  Section 3.2: Wildcard queries
  
  Section 3.3: Spelling correction

- **Wikipedia**

  
  