Burkhard-Keller-Baum - Manual

Approximative String Suche mit unterschiedlichen String-Metriken, Version 1.0, 2019

Ninell Oldenburg. Matrikelnummer: 792821. Mail: oldenburg@uni-potsdam.de

Universität Potsdam, Seminar: PRO2-A, Sommersemester 2019

Das Programm dient zur Approximativen String Suche unter Bezugnahme von drei unterschiedlichen Metriken zur Berechnung des Editierabstands.
Synopsis: `./bktree FILENAME`

Beschreibung

Das Programm nimmt über die Befehlszeile den Dateinamen einer Wortliste und baut nach User_innenangabe über die zu verwendende String-Metrik den Burkhard-Keller-Baum auf. Dabei kann zwischen der Longest-Common-Subsequenz, der Damerau-Levenshtein-Distanz oder, im Default-Fall, der Levenshtein-Distanz gewählt werden. Im Anschluss wird eine `.dot`-Datei erzeugt. Diese visualisiert den Baum mit gerichteten Kanten die den jeweiligen Editierabstand zwsichen Knoten (also den Worten) angeben. Eine Statusausgabe informiert über die Anzahl der Worte (bzw. Knoten) und seine maximale Höhe.

Im anschließenden interaktiven Modus kann der Baum nach Strings in einem gegebenen Editierabstand gesucht werden. Alle passenden Strings werden unter Angabe der Suchdauer in der Konsole ausgegeben. Findet das Programm keine Strings im Baum, wird auch darüber informiert.

Das Programm wurde unter MacOS Mojave version 10.14.6 mit Compiler g++ 4.2.1 kompiliert und getestet.

Ordnerstruktur, `make` und Programmaufruf

bin - anfangs leer
data - `DeReWo.txt` (Wortliste mit 250.000 Worten, einige davon doppelt)
- `testwordlist.txt` (Wortliste mit 13 Worten, davon eines doppelt, Leerzeilen)

doc - `BKTree.doxygen` (Doxygen-Log-File)
- `manual.pdf` (Dieses PDF)

include - `BKTree.hpp` (Header mit BKTree-Klasse)
- `Ed_Dis.hpp` (Header mit Funktionsobjekten zur Berechnung des Editierabstands)

src - `Main.cpp` (Ausführbare Main-Klasse zur Verwendung des Programms)
- `Computation.cpp` (C-Hilfsdatei zur Verwendung von Main.cpp)

test - `Demo.cpp` (Demo-Datei für dieses Programm)

- `makefile` (Makefile-Datei zum Erstellen der Einzelnen ausführbaren Programme)

make

`make`	kompiliert `Main.cpp`
`make docs`	erzeugt die HTML-Datei aus dem Doxygen-Log
`make demo`	kompiliert `Demo.cpp`
`make clean`	löscht alle Dateien wieder

Ausführbare Datei

`./bktree FILENAME`
`FILENAME` ist eine `.txt`-Datei in Listenform

Referenzen

1. Burkhard, W. A., & Keller, R. M. (1973). Some approaches to best-match file searching. *Communications of the ACM, 16*(4), 230-236.

2. Johnson, N. (02. April 2007). *Damn Cool Algorithms, Part 1: BK-Trees*. Abgerufen von: `http://blog.notdot.net/2007/4/Damn-Cool-Algorithms-Part-1-BK-Trees`

3. Johnson, N. (28. Juli 2010). *Damn Cool Algorithms: Levenshtein Automata*. Abgerufen von: `http://blog.notdot.net/2010/07/Damn-Cool-Algorithms-Levenshtein-Automata`

4. Wikipedia. (2019). *String Metric*. Abgerufen von: `https://en.wikipedia.org/wiki/String_metric`

5. Wikipedia. (2019). *Edit Distance*. Abgerufen von: `https://en.wikipedia.org/wiki/Edit_distance`

6. Wikipedia. (2019). *Damerau-Levenshtein distance*. Abgerufen von: `https://en.wikipedia.org/wiki/Damerau-Levenshtein_distance`

7. Wikipedia. (2019). *Levenshtein distance*. Abgerufen von: `https://en.wikipedia.org/wiki/Levenshtein_distance`

8. Wikipedia. (2019). *Longest common subsequence problem*. Abgerufen von: `https://en.wikipedia.org/wiki/Longest_common_subsequence_problem`

9. Xenopax's Blog. (März 2013). *The BK-Tree ? A Data Structure for Spell Checking*. Abgerufen von: `https://nullwords.wordpress.com/2013/03/13/the-bk-tree-a-data-structure-for-spell-checking/`