Aufgabe 2: Twist

Team-ID: 00126

Team-Name: GSO

Bearbeiter dieser Aufgabe: Lennart Sandbothe

Programmiersprache: Java

Inhaltsverzeichnis:

Lösungsidee

Twisten

Zum Twisten eines Textes muss zuerst einmal jedes einzelne Wort gefunden und von Satzzeichen, Zahlen, etc. getrennt werden.
Ist dies geschehen, wird der erste und der letzt Buchstaben ignoriert und dalle mittleren Buchstaben zufällig vertauscht.

Enttwisten

Auch zum enttwisten eines Textes müssen die einzelnen Wörter zuerst einmal gefunden und isoliert werden.
Ist dies geschehen kann man bei jedem Twist-wort informationen über das Ursprungswort herausfinden.
Erkennbare Daten sind zum einen der erste und der letzte Buchstabe, die auf jeden fall an der richtigen Position sind, die länge des Wortes und auch die Anzahl der verschiedenen Buchstaben.
Twistet man beispielsweise ‘Informatik’ zu ‘Ifoinarmtk’’ so kann man erkennen,
dass das Ursprungswort mit ‘I’ anfängt, mit ‘k’ aufhört, 10 Buchstaben lang ist und zwei ‘I’s, ein ‘N’ ein ‘F’ usw. beinhaltet.
Nun benötigt man eine Wörterliste. Nach dem Einlesen, kann man diese Wörter nun auf die sicheren Kriterien überprüfen.
Sind alle Kriterien erfüllt hat man ein mögliches Ursprungswort gefunden.
Abweichungen entstehen nur noch wenn entweder, kein Wort gefunden wird, welches alle Kriterien erfüllt oder es mehrere Wörter gibt. In diesem Fall kann zwischen diesen nicht mehr weiter unterschieden werden.

Umsetzung

Wörter separieren

Wie in der Lösungsidee bereits beschrieben, müssen zuerst einmal die einzelnen Wörter gefunden werden. Hierzu wird bei jedem Zeichen des Textes überprüft ob es ein Buchstabe oder etwa eine Zahl oder Satz-/Sonderzeichen ist.
Jeder Bereich in dem nur Buchstaben vorhanden sind wird hier als Wort gewertet.

Twisten

Jedes individuelle Wort wird nun in einer ‘twist’-Methode verdreht.
Hierzu werden der erste und der Letzte Buchstabe zwischengespeichert und alle übrigen in eine Liste geschrieben.
Nun fügt man an den ersten Buchstaben immer ein zufälliges Element der Liste an und entfernt diesen aus der Liste. Schließlich wird der letzte Buchstabe angehängt und das Wort zurückgegeben.

Enttwisten

Auch beim enttwisten wird werden zuerst die Wörter aus dem Text separiert.
Bei jedem Wort werden nun die sicheren und auch bei getwisteten Wörtern leicht erkennbaren Eigenschaften festgehalten. Der erste sowie der letzte Buchstabe und auch die länge des Wortes werden zwischengespeichert.
Das vorher eingelesene Wörterverzeichnis wurde ebenfalls anhand dieser Eigenschaften in einer HashMap gespeichert.
Nun wird bei allen Wörtern, welche die gleichen Eigenschaften aufweisen, überprüft ob die Anzahl der Buchstaben mit dem zu enttwistendem Wort übereinstimmt.
Ist dies bei allen Buchstaben der Fall wird das enttwistete Wort zurückgegeben.

Bedienung

Die Parameter beim Starten des Programms geben die Pfade zu den Wörtersammlungen an, mehrere Angaben sind hierbei mit einem Leerzeichen zu trennen (absteigende Präferenz).
Werden keine Parameter angegeben werden die Standard-Pfade “./words.txt” (BwInf-Beispieldatei) und “./german.dic”
(Open-Source-Wortsammlung von Jan Schreiber) angewendet.

Nach dem Starten des Programms kann man folgende Befehle verwenden:

twist [text]	Twisted den angegebenen Text
twistFile [path]	Liest die angegebene Datei und twisted ihren Inhalt
untwist	Untwisted den zuletzt getwisteten Text
saveUntwisted [path]	Speichert den zuletzt enttwisteten Text in einer Textdatei. Wenn kein Pfad angegeben: “./untwisted.txt”
help	Listet alle Befehle auf
exit	Beendet das Programm

Beispiele

1. Wörter erkennen: “twist1.txt”

Bei dem ersten Beispiel lässt sich besonders die Worterkennung gut sehen.

Hier wird beispielsweise die öffnende Klammer vor ‘Englisch twist’ erkannt und beim twisten und enttwisten ignoriert.

Auch längere Sequenzen ohne Buchstaben wie zum Beispiel ‘4/4-[Takt]’ werden problemfrei erkannt und dementsprechend behandelt.

```
twistFile ./resources/twist/twist1.txt
Der Tsiwt
(Eslgcinh tsiwt = Dehnrug, Vuerehnrdg)
war ein Metdnaoz im 4/4-Tkat,
der in den feeurhn 1960er Jharen poupelar
wurde und zu
Rcok'n'Roll, Rythhm and Buels oder slezipleer
Tswit-Muisk gnztaet wrid.

untwist
Der Twist
(Englisch Twist = Drehung, Verdrehung)
war ein Modetanz im 4/4-Takt,
der in den fruehen 1960er Jahren populaer
wurde und zu
Rock'n'Roll, Rythhm and Blues oder spezieller
Twist-Musik getanzt wird.
```

2. Die Grenzen: ‘twist3.txt’

Bei diesem Beispieldurchlauf mit der ‘twist3.txt’ kann man gut die Grenzen des enttwisten erkennen.
Diese sind stark von den Wörterlisten abhängig.
Das Wort ‘carte’ zum Beispiel ist in keiner der verwendeten Wortsammlungen und kann demnach auch nicht enttwisted werden.
In diesem Fall wird die ursprüngliche getwistete Version übernommen.

```
twistFile ./resources/twist/twist3.txt
Ein Rreuntasat, wclehes a la crate abtireet,

untwist
Ein Restaurant, welches a la crate arbeitet,
```

Quelltext

Twister.java: twistWord

```
private String twistWord(String s) {
   s = this.normalize(s);
   if(s.length() <= 3) {
       return s;
   }

   char first = s.charAt(0);
   char last = s.charAt(s.length()-1);

   s = s.substring(1, s.length()-1);
   List<Character> chars = new ArrayList<>();
   for(char c : s.toCharArray()) {
       chars.add(c);
   }
   Random r = new Random();
   String twisted = ""+first;
   for(int i = chars.size(); i > 0; i = chars.size()) {
       int charIndex = r.nextInt(i);
       twisted += chars.get(charIndex);
       chars.remove(charIndex);
   }
   twisted += last;

   return twisted;
}
```

Twister.java: untwistSentence

```
public String untwistSentence(String sentence) {
   String untiwsted = "";

   int start = -1;
   for(int i = 0; i < sentence.length(); i++) {
       char c = sentence.charAt(i);
       if(Character.isLetter(c) && start == -1) {
           start = i;
       } else if((!Character.isLetter(c) || i ==
           sentence.length()-1) && start != -1) {
           String twistedWord =
                this.untwistWord(sentence.substring(start, i));
           untiwsted += twistedWord;
           start = -1;
           untiwsted += c;
       } else if(!Character.isLetter(c)) {
           untiwsted += c;
       }
   }

   return untiwsted;
}
```

Twister.java: untwistWord

```
public String untwistWord(String word) {
   char first = Character.toUpperCase(word.charAt(0));
   char last = Character.toUpperCase(word.charAt(word.length()-1));
   int length = word.length();
   String key = ""+first+last+length;

   for(Map<String, List<String>> wc : this.collections) {
       if(wc.containsKey(key)) {
           eqKeyCheck:
           for (String eqKey : wc.get(key)) {
               for (char c : word.toCharArray()) {
                   if (this.numCharInString(word, c) !=
                     this.numCharInString(eqKey, c)) {
                       continue eqKeyCheck;
                   }
               }
               return (Character.isUpperCase(word.charAt(0))) ?
                firstUpperCase(eqKey): eqKey;
           }
       }
   }

   return word;
}
```

Twister.java: Twister

```
...
Map<String, List<String>> words = new HashMap<>();
BufferedReader br = new BufferedReader(new
FileReader(wordFilePath));
for (String line = br.readLine(); line != null; line =
     br.readLine()) {
   String normalized = this.normalize(line);

   char first = Character.toUpperCase(normalized.charAt(0));
   char last =
Character.toUpperCase(normalized.charAt(normalized.length()-1));
   String key = ""+first+last+normalized.length();

   words.putIfAbsent(key, new ArrayList<>());
   words.get(key).add(normalized);
}
...
```