Aufgabe 1: Störung

Team-ID: 00772

Team: Aino Spring

Bearbeiter/-innen dieser Aufgabe:
Aino Spring

12. November 2022

Inhaltsverzeichnis

Lösungsidee

Es werden zuerst alle Wörter des Textes/Buches überprüft, ob sie dem ersten Wort des unvollständigen Satzes gleichen. Falls dies der Fall ist, werden die nächsten Wörter überprüft, ob sie mit dem unvollständigen Satz übereinstimmen. Ist ein „_“ im unvollständigen Satz, wird es als Wort erkannt, welches jedem anderen gleicht. Sonderzeichen im Text werden nicht beachtet.

Umsetzung

Findung der Satzmöglichkeiten

Die erste Funktion ist die „CheckWords(word string, wordInBook string)“ Funktion. Sie Überprüft, ob ein Wort einem anderen gleicht. Die Groß- und Kleinschreibung wird hierbei nicht beachtet. Falls der erste Parameter (das Wort aus dem unvollständigen Satz) ein „_“ ist, wird immer „true“ zurückgegeben.

Die Funktion „Search(sentence string, text *string, returnChannel chan string)” sucht alle Satzmöglichkeiten in einem Text. Der Text wird als Pointer übergeben, da er oft lang werden kann und es so effizienter ist. Die gefundenen Sätze werden über einen Kanal zurückgegeben, da es zu mehreren möglichen Sätzen kommen kann und so Multithreading ermöglicht werden kann. Die Sätze werden gefunden, indem über jedes Wort des Textes iteriert wird und mit der im obigen Punkt genannten „CheckWords“ Funktion überprüft wird, ob das erste Wort des unvollständigen Satzes und das aktuelle Wort sich gleichen. Falls dies der Fall ist, werden die nächsten Wörter im Text mit den nächsten Wörtern des unvollständigen Satzes überprüft. Wenn die „CheckWords“ Funktion bei jedem Wortpaar „true“ zurückgegeben hat, wird der vollständige Satz aus dem Text über den Kanal zurückgegeben.

Importierung des Buchtexts und der unvollständigen Wörter

Die „ImportBook()“ Funktion importiert den Text des Buchs aus der in der config-Datei angegebenen Datei. Dazu wird die erste Zeile der config-Datei ausgelesen und der Inhalt als Pfad für die Datei zum Buch verwendet. Der Inhalt dieser Datei wird ausgelesen und doppelte Leer-, Sonder- sowie CRLF-Zeichen werden entfernt. Die Rückgabe des Texts erfolgt als Pointer, wegen den im obigen Punkt zur „Search“ Funktion genannten Gründen.
Die „ImportFiles()“ Funktion importiert die in der config-Datei ab zweiter Zeile angegebenen Dateien mit den unvollständigen Sätzen. Dazu wird der ganze Inhalt ab der zweiten Zeile der config-Datei ausgelesen. Jede Zeile jener Datei repräsentiert einen Pfad zu einer Datei mit einem unvollständigen Satz. Diese Dateien werden ausgelesen und als string array (in diesem Fall ein string slice) zurückgegeben.

Alles zusammenfügen

In der main-Funktion wird alles zusammengefügt. Das Buch und die unvollständigen Sätze werden mithilfe der in den obigen Punkten zur Importierung des Buchtexts und der unvollständigen Sätze genannten Funktionen „ImportBook“ und „ImportFiles“ importiert und für jeden unvollständigen Satz werden alle Satzmöglichkeiten mit der „Search“ Funktion herausgefunden und diese werden mit dem Index des unvollständigen Satzes über die Standartausgabe (stdout) ausgegeben.

Beispiele

stoerung0.txt

"Das kommt mir gar nicht richtig vor"

stoerung1.txt

"Ich muß in Clara verwandelt"
"Ich muß doch Clara sein"

stoerung2.txt

"Fressen Katzen gern Spatzen"
"Fressen Katzen gern Spatzen"
"Fressen Spatzen gern Katzen"

stoerung3.txt

"das Spiel fing an"

stoerung4.txt

"ein sehr schöner Tag"

stoerung5.txt

"Wollen Sie so gut sein"

Eigener unvollständiger Satz

"Setzt euch ihr Alle und hört mir zu ich will euch bald genug trocken machen"

Ich wählte diesen Satz, da der originale Satz („Setzt euch, ihr Alle, und hört mir zu! ich will euch bald genug trocken machen!“ [Z. 594 – Z. 595]) mehrere Sonderzeichen an den Enden und in der Mitte des Satzes enthielt und über zwei Zeilen ging.

Quellcode

Findung der Satzmöglichkeiten

Die Funktion, um zu überprüfen, ob sich zwei Wörter gleichen:

```
func CheckWords(word string, wordInBook string) bool {
   word = strings.ToLower(word)
   wordInBook = strings.ToLower(wordInBook)
   if "_" == word {
      return true
   }
   return word == wordInBook
}
```

Die Funktion, um alle Satzmöglichkeiten zu finden:

```
func Search(sentence string, text *string, returnChannel chan string) {
   defer close(returnChannel)
   var sentenceSlice = strings.Split(sentence, " ")
   var textSlice = strings.Split(*text, " ")
   for idx, word := range textSlice {
      if CheckWords(sentenceSlice[0], word) {
         var isSame = true
```

```
            for i := 1; i < len(sentenceSlice); i++ {
                if idx+i >= (len(textSlice) - 1) {
                    isSame = false
                } else {
                    if !CheckWords(sentenceSlice[i], textSlice[idx+i]) {
                        isSame = false
                        break
                    }
                }
            }
            if isSame {
                returnChannel <- strings.Join(textSlice[idx:idx+len(sen-
tenceSlice)], " ")
            }
        }
    }
}
```

Importierung des Buchtexts und der unvollständigen Wörter

Der Pfad zur config-Datei:

```
const (
    CONFIG = "config"
)
```

Die ignorierten Zeichen:

```
var (
    IGNORED_CHARACTERS = []string{"»", "«", ",", ".", "?", "!", "_", "\""}
)
```

Die Funktion, um das Buch zu importieren:

```
func ImportBook() *string {
    data, err := os.ReadFile(CONFIG)
    if err != nil {
        log.Panic(err)
    }
    bookData, bookErr := os.ReadFile(strings.TrimSuf-
fix(strings.Split(string(data), "\n")[0], "\r"))
    if bookErr != nil {
        log.Panic(bookErr)
    }
    var bookString = string(bookData)
    bookString = strings.Replace(bookString, "\n", " ", -1)
    bookString = strings.Replace(bookString, "\r", "", -1)
    bookString = strings.Replace(bookString, "  ", " ", -1)
    for _, ignoredCharacter := range IGNORED_CHARACTERS {
        bookString = strings.Replace(bookString, ignoredCharacter, "", -1)
    }
    return &bookString
}
```

Die Funktion, welche die Dateien mit den unvollständigen Sätzen importiert:

```
func ImportFiles() []string {
   var sentences = make([]string, 0)
   data, err := os.ReadFile(CONFIG)
   if err != nil {
      log.Panic(err)
   }
   for _, line := range strings.Split(string(data), "\n")[1:] {
      line = strings.TrimSuffix(line, "\r")
      sentenceData, sentenceErr := os.ReadFile(line)
      if sentenceErr != nil {
         log.Panicf("%#v: %v", line, sentenceErr)
      }
      sentences = append(sentences, strings.Replace(strings.Re-
place(string(sentenceData), "\r", "", -1), "\n", "", -1))
   }
   return sentences
}
```

Alles zusammenfügen

Die main-Funktion, welche alle Funktion zusammenfügt:

```
func main() {
   var uncompletedSentences = ImportFiles()
   var book = ImportBook()
   for idx, uncompletedSentence := range uncompletedSentences {
      var sentenceChannel = make(chan string)
      fmt.Printf("Sentence possibilities for uncompleted sentence %v: \n",
idx)
      go Search(uncompletedSentence, book, sentenceChannel)
      for sentence := range sentenceChannel {
         fmt.Printf("%#v\n", sentence)
      }
      fmt.Print("\n")
   }
}
```