

Studienarbeit 2

Bericht / Textanalyse - Plagiatfinder

Marco Costa

Mario Lanfranchi



2009

Bericht / Textanalyse – Plagiatfinder

Diese Studienarbeit wurde im Rahmen des Bachelor-Studienganges
Telekommunikation/Elektrotechnik an der Hochschule für Technik und Wirtschaft
HTW Chur erstellt.

Studierende

Marco Costa
Aino 440
7741 San Carlo

Mario Lanfranchi
L'Alt
7745 Li Curt

Dozent

Martin Studer, Dipl. Inf.-Ing. ETH
HTW Chur
Ringstrasse/Pulvermühlestrasse 57
7004 Chur

Inhaltsverzeichnis

1	Einleitung.....	3
2	Einführung zu Plagiate.....	3
3	Idee: Plagiatfinder.....	4
4	Projektziele	6
5	Ergebnisse.....	6
6	Algorithmus Plagiatfinder in R.....	8
7	Schlussfolgerung	17
8	Quellen	17

1 Einleitung

„Texte enthalten Buchstaben, Interpunktionszeichen, Wörter, Sätze, Abschnitte, usw. Die Möglichkeiten zur Kombination von Buchstaben, Zeichen, Wörtern, Sätzen sind unermesslich. Deshalb ist die Analyse von Texten ein spannendes Forschungsfeld. Beispielweise hat die Analyse von Texten gezeigt, mit welcher Wahrscheinlichkeit die einzelnen Buchstaben des Alphabets im Durchschnitt auftreten. Diese Erkenntnisse haben es früher erlaubt, verschlüsselte Texte ohne Kenntnis des Schlüssels zu dechiffrieren. Heute sind auf allen Mail-Servern Anti-Spam-Programme im Einsatz, welche die ausgetauschten Mitteilungstexte analysieren, um unerwünschte Nachrichten (Spam) von den korrekten Mails unterscheiden zu können. Ein anderer Bereich der Textanalyse befasst sich mit dem Erkennen von Plagiaten, d.h. mit dem Erkennen von Texten (beispielweise von Studierenden), die aus irgendwelchen Quellen (z.B. im Web) kopiert wurden.“¹

Das Gebiet der Textanalyse ist riesig. Jeder Teil dieses Gebiets ist interessant und bietet eine Vielzahl von Vertiefungsmöglichkeiten. Das Thema, welches uns am meisten interessiert hat, ist das Erkennen von Plagiaten. Wir finden diese sei ein sehr aktuelles und komplexes Thema.

2 Einführung zu Plagiate

„**Plagiat** (vom lat. Wort *plagium*, „Menschenraub“ abgeleitet) ist die Vorlage fremden geistigen Eigentums bzw. eines fremden Werkes als eigenes oder Teil eines eigenen Werkes. Dieses kann sowohl eine exakte Kopie, eine Bearbeitung (Umstellung von Wörtern oder Sätzen), eine Nacherzählung (Strukturübernahme) oder eine Übersetzung sein.“²

Obwohl diese kurze Einführung nicht unser geistiges Eigentum ist, handelt es sich nicht um ein Plagiat, da wir klar deklarierten, von wo diese Abklärung stammt.

Eigentlich heisst das lat. Wort *plagiatum* „Ausflucht“, aber im Römischen Recht bedeutete *Plagio* das Verkaufen eines freien Mannes als Sklave oder das Entführen eines Menschen.

¹ Quelle: Studer M., Wenk B. Studienarbeit 2, Version 1, 15.01.2009

² Quelle: <http://de.wikipedia.org/wiki/Plagiarismus>, 14.01.2009

Im Laufe der Zeit hat sich die Bedeutung von Plagiat ein wenig verändert. Heutzutage kreiert Plagiate, wer Ideen oder geistiges Eigentum „raubt“. Plagiate kann man in der Literatur, in der Musik oder auch in der Wissenschaft finden. Das Problem ist, dass Plagiate eigentlich nicht strafbar sind. Wenn man sein geistiges Werk von Plagiaten schützen will, muss man es unter Urheberrecht setzen. Weiter kann man in einem Arbeits- oder Honorarvertrag festhalten, dass der Arbeiter kein Plagiat begehen darf. Bei Schulen oder Hochschulen sind Plagiate auch nicht akzeptiert. Je nach Reglement kann ein Plagiat zu einer schlechten Note oder sogar zum Ausschluss führen. Plagiate bei Semesterarbeiten oder Aufsätzen in der Schule haben uns besonders interessiert. Wir dachten, es sei eine echte Herausforderung ein kleines Programm zu schreiben, welches Plagiate finden und gleichzeitig auch noch die Ergebnisse graphisch darstellen könnte.

3 Idee: *Plagiatfinder*

In dieser Semesterarbeit mussten wir das Programm *R* benutzen. Dieses Programm eignet sich zum Analysieren graphischer Darstellung von Daten. Die Idee war, dass die Studierenden sich in den Stoff (Programm *R*) einarbeiteten. Sie sollten im Internet recherchieren und die Quellen mit den gefundenen Informationen sollten angegeben werden. Am Schluss konnte man Algorithmen im Internet suchen um die selber erfundene Daten analysieren.

Da wir schon Erfahrung mit Programmieren hatten, haben wir schnell festgestellt dass das Programm *R* ähnlich aufgebaut ist wie andere Programmiersprachen (z.B. *Java*, *c++*). Es gab die Möglichkeit, ein eigener Algorithmus zu kreieren. Wir haben das Recherchieren von Literatur über *R* ein bisschen vernachlässigt, um uns über das Programm (*Plagiatfinder* haben wir es benannt) zu konzentrieren. Wir haben uns oft getroffen, um zu diskutieren, was das Programm machen sollte, welchen Typ von Plagiat musste es erkennen, wie sollte das Programm die Plagiate finden und wie sollte es schlussendlich die Resultate graphisch darstellen.

Nach dieser Phase haben wir mit dem Programmieren begonnen. Wir programmierten *Plagiatfinder* sorgfältig. Auch wenn das Programm schon funktionierte, änderten wir manchmal noch Kleinigkeiten, um das Programm zu perfektionieren. Auch in die Ästhetik haben wir Zeit investiert. Die Variablen und Konstanten haben nach einem

ersten Entwurf nachvollziehbare Namen erhalten so, dass auch eine dritte Person das Programm schnell versteht und eventuell auch erweitern könnte.

Das Programm bietet eigentlich zwei Verfahren:

Verfahren 1: ganze Sätze

Dieser Verfahren eignet sich nur für Sätze, die eins zu eins vom Internet kopiert worden sind.

Verfahren 2: modifizierte Sätze

Dieser Verfahren ist langsamer, kann aber auch Sätze die leicht verändert worden sind, erkennen.

4 Projektziele

Plagiatfinder muss Sätze identifizieren, die eins zu eins vom Internet kopiert wurden sowie auch Sätze, in welchen einige Wörter ausgetauscht worden sind, Das Programm analysiert die Sätze von Wort zu Wort. So ist es schwer, Sätze zu identifizieren, die im Satzaufbau verändert wurden (z.B. Hauptsatz, Nebensatz -> Hauptsatz + eingeschobener Nebensatz). Wenn es zeitlich drin liegt, werden wir versuchen, das Programm so zu erweitern, dass auch diese Art von Sätzen erkannt wird.

5 Ergebnisse

Ergebnis beim analysieren eines Probetextes (Plagiat)

```
| #####  
#           Plagiatfinder v 1.2           #  
#####  
  
Bitte wählen Sie welches Verfahren anwenden wollen.  
  
1 -> Datei auswählen  
2 -> Text eingeben
```

Wir analysierten den folgenden Text:

„Die elektrische Ladung war seit Coulomb bekannt, die elektrische Spannung seit Volta und die Wirkung des elektrischen Stromes seit Ampere. Hier auf MSN Sport finden Sie Sportnachrichten aus aller Welt und allen Sportarten. In diese Semesterarbeit mussten wir das Programm *R* benutzen.“

Man wählt das gewünschte verfahren:

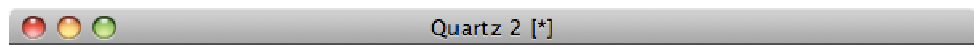
```
Bitte wählen Sie welches Verfahren anwenden wollen.  
  
1 -> Schnelles Verfahren (Besonders gut für Sätze, die eins zu eins kopiert wurden)  
2 -> Detailliertes Verfahren (Besonders für Sätze, die leicht geändert wurden)  
3 -> Beenden
```

Die kopierten Sätze werden mit den entsprechenden Links angegeben:

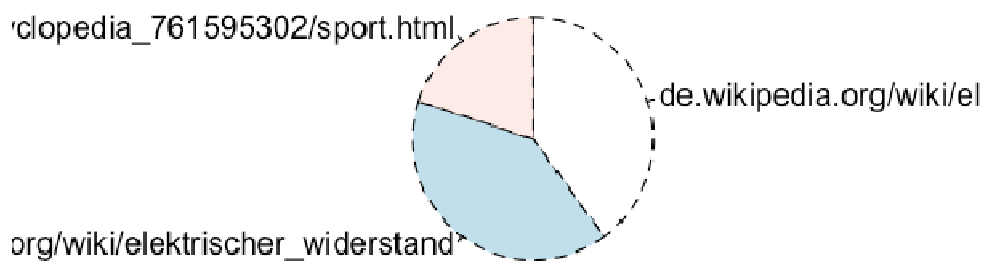
Die folgende Sätze wurden aus dem Internet kopiert:

1. Die elektrische Ladung war seit Coulomb bekannt -> de.wikipedia.org/wiki/elektrischer_widerstand
2. die elektrische Spannung seit Volta und die Wirkung des elektrischen Stromes seit Ampere -> de.wikipedia.org/wiki/elektrischer_widerstand
3. Hier auf MSN Sport finden Sie Sportnachrichten aus aller Welt und allen Sportarten. -> de.encarta.msn.com/encyclopedia_761595302/sport.html

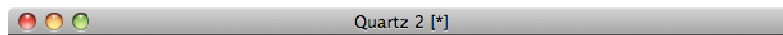
Graphische Darstellung des Ergebnisses:



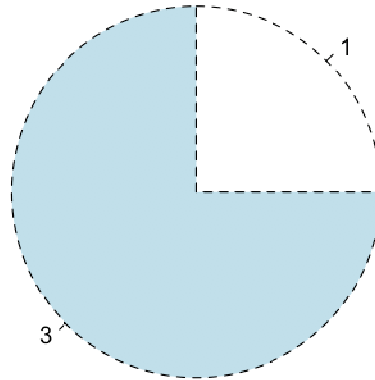
Ergebnisse



Die erste Grafik zeigt, von wo die Sätze kopiert worden sind und in welchem prozentualen Verhältnis.



Blau -> Anzahl kopierte Sätze



Die zweite Grafik zeigt, wie viele Plagiat-Sätze anteilmässig im Text vorkommen.

6 Algorithmus *Plagiatfinder* in R

In diesem Teil des Berichtes wird der *Plagiatfinder-Code* aufgezeigt und dessen Funktionen erklärt. Der *Plagiatfinder* beinhaltet sieben Funktionen:

1. *charClean()*
2. *splitter()*
3. *urlConnection()*
4. *analyze()*
5. *analyzePro()*
6. *printResult()*
7. *start()*

Auf den folgenden Seiten werden die einzelnen Funktionen unter die Lupe genommen:

1. *charClean()*

```
charClean <- function (text){  
  char <- matrix(0, 20,2)  
  char[1,1] <- "ä"  
  char[1,2] <- "ae"  
  char[2,1] <- "ö"  
  char[2,2] <- "oe"  
  char[3,1] <- "ü"  
  char[3,2] <- "ue"  
  ...  
  i<-1  
  
  while(i <= length(char) / 2){  
    text <- gsub(char[i,1], char[i,2], text)  
    i <- i+1  
  }  
  return (text)  
}
```

Das Programm *R* hat Schwierigkeiten beim Vergleichen von Sonderzeichen. So musste man eine Funktion programmieren, die die Sonderzeichen in einem Text in andere Zeichen umwandelte oder löschte. Am Ende wird der bereinigte Text zurückgegeben.

2. *splitter()*

```
splitter <- function (text){  
  splitter <- c("\\. ", "\\, ")  
  i <- 1  
  while(i<=length(splitter)){  
    text <- unlist(strsplit(text, splitter[i]))  
    i <- i+1  
  }  
  return (text)  
}
```

Hier werden die einzelne Sätze in einem Array gespeichert und zurückgegeben.

3. *urlConnection()*

```
urlConnection <- function (satz, verfahren, seite){
  splittext <- splitter(originaltext)
  splittexturl <- gsub(' ', '%20', splittext[satz])
  if(verfahren == 0){
    urlcon = paste('http://cosweb.dyndns.org/search.php?s=%22',
  splittexturl, '%22', sep = "")
  }
  else{
    urlcon = paste('http://cosweb.dyndns.org/search.php?res=1&s=',
  splittexturl, '&start=', seite, sep = "")
  }
  con <- url(urlcon , "r")
  quelltext <- readLines(con, warn = FALSE)

  Encoding(quelltext)
  Encoding(quelltext) <- "latin1"

  close(con)
  return (quelltext)
}
```

Diese Funktion erstellt eine Verbindung zwischen dem Webserver und dem Programm R. Grundsätzlich sucht der Webserver auf der Google-Suchmaschine nach dem weitergegebenen Satz mit Hilfe des Parameters und der gesuchte Quelltext wird von der Suchmaschine zurückgegeben. Anhand der Adresse sieht man, dass drei Parameter weitergegeben werden:

1. *res(restriction)*: Wenn *res=1* wird automatisch den letzten Teil des Codes abgeschnitten, sodass Fehler in der Funktionen *analyzePro* vermieden werden.
2. *s(Satz)*: Hier wird den Satz weitergegeben. „%22“ bedeutet, dass die Suchmaschine nach dem genauen Satz (genaue Reihenfolge der Worte) im Internet suchen musste.
3. *start*: Entspricht der Seitennummer der Google-Suchmaschine. Zum Beispiel Quelltext der zweiten Seite. Dieser Parameter wird nur für das Verfahren *analyzePro* verwendet, um auch auf der Google-Seiten zu suchen.

4. analyze()

```
analyze <- function () {
  u<-1
  a<-1
  splittextOrg <- splitter(originaltext)
  splittext <- tolower(splitter(originaltext))
  cop <- matrix(0, length(splittext),2)
  while(u <= length(splittext)){
    quelltext <- urlConnection(u,0)

    start <- gregexpr("<div class=\"s\"", quelltext[4])
    stop <- gregexpr("aehnliche seiten", quelltext[4])

    article <- array()
    urlarticle <- array()

    article[1] <- substr(quelltext[4], start[[1]][[1]], stop[[1]][[1]])

    start <- gregexpr("<cite>", article[1])
    stop <- gregexpr("</cite>", article[1])
    urlarticle[1] <- substr(article[1], start[[1]][[1]]+6, stop[[1]][[1]]-
4)

    if(nchar(splittext[u]) > 130){
      splittext[u] <- substr(splittext[u], 0, 130)
      lastspace <- gregexpr(" ", splittext[u])
      splittext[u] <- substr(splittext[u], 0,
lastspace[[1]][[length(lastspace[[1]])]]-1)
    }

    article[1] <- gsub("'", "", article[1])
    splittext[u] <- gsub("'", "", splittext[u])
    splittext[u] <- gsub("\\.", "", splittext[u])

    if(regexpr(splittext[u], article[1], ignore.case = TRUE, extended =
TRUE) != -1){
      cop[a,1] <- splittextOrg[u]
      urlarticle[1] <- gsub("<b>", "", urlarticle[1])
      urlarticle[1] <- gsub("</b>", "", urlarticle[1])
      cop[a,2] <- urlarticle[1]
      a <- a+1
    }
    u <- u+1
  }
  return (cop)
}
```

Um die Suche nach Plagiaten zu verbessern, wurde die Funktion *tolower()* verwendet (alle Buchstaben werden klein geschrieben). Das *cop*-Array ist zweidimensional und enthält die gefundenen und die kopierten Sätze sowie, wo die Internetadresse, gefunden wurde. In der Schleife wird der Quelltext analysiert und wird gesucht, ob der Satz auch in ihm enthalten ist. Wenn das der Fall ist, bedeutet es, dass ein Plagiat entdeckt wurde.

Mit den Variablen *start* und *stop* wird die Position gespeichert, wo der Artikel anfängt und endet. So werden die einzelnen Artikel (auf der Google-Seite) getrennt und in der

Variable *article* gespeichert. Bei diesem Verfahren werden nur die Sätze mit den Sätzen, die im ersten Artikel enthalten sind, gesucht. Es hat keinen Sinn, auch in den anderen Artikeln zu suchen, weil bei diesem Verfahren nur die Sätze, die eins zu eins kopiert wurden, entdeckt werden. Dank der Google-Suchmaschine ist sicher, dass im ersten Artikel der gesuchte Satz enthalten ist. Wenn das nicht der Fall ist, bedeutet dies, dass es kein Plagiat ist.

Die Variable *a* zählt, wie viele Sätze *Plagiatfinder* bis jetzt entdeckt hat.

Wenn ein Satz zu lang ist, wird die Google-Suchmaschine beim Erstellen der Ergebnisse den Satz automatisch verkürzen. Wenn man die Sätze in der Originallänge beibehalten würde, könnte sie das Programm nicht als Plagiat erkennen. Um dieses Problem zu beheben, wird die Länge des Satzes vor dem Vergleichen überprüft. Wenn der Satz mehr als 130 Zeichen hat, wird er gekürzt.

Im letzten Schritt der Funktion wird die Variable *cop* zurückgegeben.

5. *analyzePro()*

```
analyzePro <- function (){
  seite <- 0
  splittextOrg <- splitter(originaltext)
  splittext <- tolower(splitter(originaltext))
  cop <- matrix(0, length(splittext)*3, 2)
  a <- 1
  while(seite <= 2){
    u<-1
    while(u <= length(splittext)){
      quelltext <- urlConnection(u,1,seite)

      start <- gregexpr("<div class=\"s\"", quelltext[4])
      stop <- gregexpr("aehnliche seiten", quelltext[4])

      article <- array()
      urlarticle <- array()

      i <- 1
      while(i <= length(start[[1]])){
        article[i] <- substr(quelltext[4], start[[1]][[i]],
stop[[1]][[i]])
        i <- i+1
      }

      i <- 1
      #estrae il link
      while(i <= length(article)){
        start <- gregexpr("<cite>", article[i])
        stop <- gregexpr("</cite>", article[i])
        urlarticle[i] <- substr(article[i], start[[1]][[1]]+6,
```

```
stop[[1]][[1]]-4)
    i <- i+1
  }

  if(nchar(splittext[u]) > 130){
    splittext[u] <- substr(splittext[u], 0, 130)
    lastspace <- gregexpr(" ", splittext[u])
    splittext[u] <- substr(splittext[u], 0,
lastspace[[1]][[length(lastspace[[1]])]]-1)
  }

  splittext[u] <- gsub("'", "", splittext[u])
  splittext[u] <- gsub("\\.", "", splittext[u])

  i <- 1
  words <- unlist(strsplit(splittext[u], " "))

  while(i <= length(article)){
    article[i] <- gsub("'", "", article[i])
    h <- 1
    find <- 0
    count <- 0
    ok <- 0
    while(h <= length(words)){
      if(regexpr(words[h], article[i]) != -1){
        count <- count+1
      }
      h <- h+1
    }

    find <- gregexpr("<em>", article[i])
    find <- length(find[[1]])

    if(find == 1 || find == 2){
      ok <- 1
    }

    if(count >= length(words)-1 & ok == 1){
      cop[a,1] <- splittextOrg[u]
      urlarticle[i] <- gsub("<b>", "", urlarticle[i])
      urlarticle[i] <- gsub("</b>", "", urlarticle[i])
      cop[a,2] <- urlarticle[i]
      a <- a+1
      break
    }

    i <- i+1
  }
  u <- u+1
}
seite <- seite+1
}

#Doppelte erfasste Einträge löschen
i <- length(cop[])/2
while(i >= 1){
  u <- 1
  while(u <= length(cop[])/2){
    if(i != u) {
      if(cop[i,1] == cop[u,1]) {
        cop[i,1] <- "0"
        cop[i,2] <- "0"
      }
    }
    u <- u+1
  }
  i <- i-1
}
```

```
        i <- i-1
    }
    return (cop)
}
```

Das zweite Verfahren funktioniert wie das erste. Der Variablenname und die Funktion ist immer die gleiche. Anhand des Codes merkt man, dass mehrere Schleifen enthalten sind:

Hier wird zusätzlich noch auf der zweiten und dritten Google-Seiten gesucht und die Sätze werden in allen Artikeln nachgeschaut und nicht nur im ersten. Es kann beim Suchen und Vergleichen von Sätzen vorkommen, dass das Skript in dem Variable *cop* die gefundenen Plagiate doppelt speichert. So wurde am Ende dieser Funktion noch eine Schleife programmiert, die die doppelt erfassten Einträge löscht.

Beim Suchen werden die einzelnen Worte eines Satzes im Artikel gesucht. Wenn alle Worte im Artikel enthalten sind und wenn nur ein oder zwei Sätze im Artikel fett sind (Variabel *find*, Suche nach ein oder zwei **), dann wurde ein Plagiat erkannt.

6. *printResult()*

```
printResult <- function (x){
  if(x == 0){results <- analyze()}
  else{results = analyzePro()}
  resultsm <- results
  cat("Die folgende Sätze wurden aus dem Internet kopiert:", "\n\n")
  i <- 1
  anzahlSaetze <- 0

  while(i <= length(results)/2){
    if(results[i,1] != "0"){anzahlSaetze <- anzahlSaetze+1}
    i <- i+1
  }

  data <- array()
  data1 <- c(AnzahlSaetzeStart-anzahlSaetze, anzahlSaetze)
  labels <- array()
  labels1 <- c(AnzahlSaetzeStart-anzahlSaetze, anzahlSaetze)
  text<-array()
  i <- 1
  a <- 1

  while(i <= length(results)/2) {
    u <- 1
    count <- 0
    if(results[i,1] != 0) {
      while(u <= length(results)/2) {
        resultsm[i,2] <- substr(results[i,2], 0, 15)
        if(regexpr(resultsm[i,2], resultsm[u,2]) != -1 &
resultsm[i,2] != 0){
          count<-count+1
        }
        u <- u+1
      }
    }
  }
}
```

```
    }
    cat(i, ". ", results[i,1], " -> ", results[i,2], "\n\n", sep = "")
    text[a] <- paste(i, ". ", results[i,1], " -> ", results[i,2], "\n\n",
sep = "")
    labels[a] <- results[i,2]
    data[a] <- count
    a <- a+1
  }
  i <- i+1
}

cat("Wollen Sie die Ergebnisse als Diagramme sehen? (j/n)", "\n", sep="")
id <- readline()
if(id == "j") {
  pie(data, labels = labels, main=text, clockwise=TRUE, lty = 2,
radius=0.5)
}
cat("Wollen Sie einen Übersicht als Diagramm von wie vielen Sätzen kopiert
wurden? (j/n)", "\n", sep="")
id <- readline()
if(id == "j") {
  pie(data1, labels = labels1, main="Blau -> Anzahl kopierte Sätze",
clockwise=TRUE, lty = 2, radius=0.5)
}
cat("Wollen Sie einen anderen Text analysieren? (j/n)", "\n", sep="")
id <- readline()
if(id == "j") {start()}
else{q(save = "no")}
}
```

Mit Hilfe dieser Funktion wird die Variable *cop* analysiert und die Ergebnisse als Diagramme dargestellt.

7. start()

```
start <- function(){
  cat("Bitte wählen Sie welches Verfahren anwenden wollen.", "\n\n", "1 -> Datei
auswählen", "\n", "2 -> Text eingeben", "\n", sep = "")
  id<-readline()
  if(id == 1){
    textfile <- file(file.choose(), "r")
    originaltext <- readLines(textfile, warn = FALSE)

  }
  if(id == 2){
    originaltext <- readline()

  }
  if(id != 1 & id != 2){
    cat("Bitte geben Sie eine Nummer zwischen 1 und 2 an!", "\n\n")
    start()
  }

  originaltext <<- charClean(originaltext)
  splittext <- splitter(originaltext)
  anzahlSaetzeStart <<- length(splittext)

  cat("Bitte wählen Sie welches Verfahren anwenden wollen.", "\n\n", "1 ->
Schnelles Verfahren (Besonders gut für Sätze, die eins zu eins kopiert
wurden)", "\n", "2 -> Detailliertes Verfahren (Besonders für Sätze, die leicht
gändert wurden)", "\n", "3 -> Beenden", "\n", sep = "")
  id <- readline()
  if(id == 1){printResult(0)}
  if(id == 2){printResult(1)}
  if(id == 3){q(save = "no")}
  if(id != 1 & id != 2){
    cat("Bitte geben Sie eine Nummer zwischen 1 und 3 an!", "\n\n")
    start()
  }
}
```

Um das Programm *Plagiatfinder* zu starten, muss man nur den Befehl `start()` ausführen. Die Variablen `originaltext` und `anzahlSaetzeStart` werden als globalen Variablen definiert, sodass die anderen genannten Funktionen auf diese Variablen zugreifen können.

7 Schlussfolgerung

In der Studienarbeit 2 haben wir das Programm *R* kennengelernt. Wir konnten uns in das neue Gebiet einarbeiten und dank unseren allgemeinen Kenntnissen im Programmieren, konnten wir ein gutes Programm erstellen, um Plagiate zu suchen. Wir sind mit dem Resultat unserer Studienarbeit zufrieden. Das Programm *Plagiatfinder* funktioniert so wie wir uns das am Anfang der Studienarbeit vorgestellt haben. Die zwei Verfahren funktionieren beide gut und die Ergebnisse können graphisch dargestellt werden. Am Anfang haben wir auch noch über die Möglichkeiten diskutiert, ein drittes Verfahren zu programmieren. Dieses sollte auch kompliziertere Sätze erkennen, z.B. wenn das Original aus einem Satz und einem Nebensatz bestand und im Plagiat dem Hauptsatz, ein Nebensatz eingeschoben wurde. Das Programmieren dieses Verfahrens wäre jedoch zu kompliziert und zu zeitaufwendig gewesen. So haben wir uns entschieden, auf die zwei anderen Verfahren zu konzentrieren und dafür diese genauer zu programmieren.

Dank der Studienarbeit 2 hatten wir auch die Gelegenheit, etwas über die Plagiate sowie deren Geschichte zu lernen.

8 Quellen

- <http://de.wikipedia.org/wiki/Plagiat>, 5.03.2009
- <http://wiki.r-project.org/rwiki/doku.php>, 24.03.2009
- Studer M., Wenk B. Studienarbeit 2, Version 1, 6.03.2009