Konkordanzen aus dem Bonner Frühneuhochdeutschkorpus exportieren

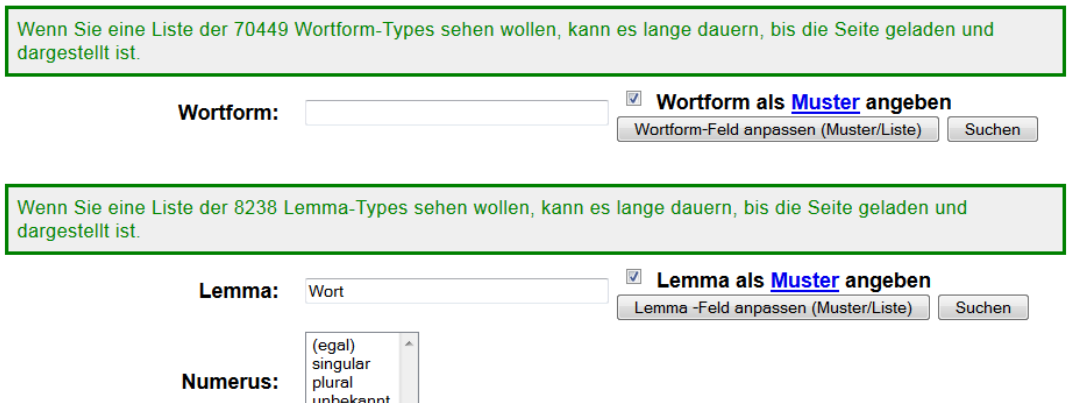
Das Bonner Frühneuhochdeutschkorpus (FnhdC) ist derzeit eine der besten Ressourcen, die wir zu dieser Periode der deutschen Sprachgeschichte haben. Einer der größten Vorteile besteht darin, dass es von Hand annotiert wurde und daher das Tagging nicht so fehlerbehaftet ist wie bei automatisch getaggten Korpora. Ein Nachteil besteht darin, dass es keine native Möglichkeit gibt, die Konkordanzen in Tabellenform zu exportieren. Daher empfiehlt es sich für viele Zwecke, die im xml-Format verfügbaren Korpusdateien herunterzuladen und damit zu arbeiten. Dies erfordert jedoch ein wenig Einarbeitung. Für alle, die dafür nicht die Zeit haben und denen eine relativ einfache Konkordanz mit den wichtigsten Daten genügt, habe ich ein Skript geschrieben, das die Daten aus der Suchschnittstelle des FnhdC direkt in Tabellenform überträgt.

1. Suche im FnhdC-Interface

Zunächst rufen wir unter http://www.korpora.org/fnhd/Suche das Such-Interface auf. Hier sind zunächst alle Texte des Korpus aufgelistet. Wir wählen „In allen Dokumenten suchen“.

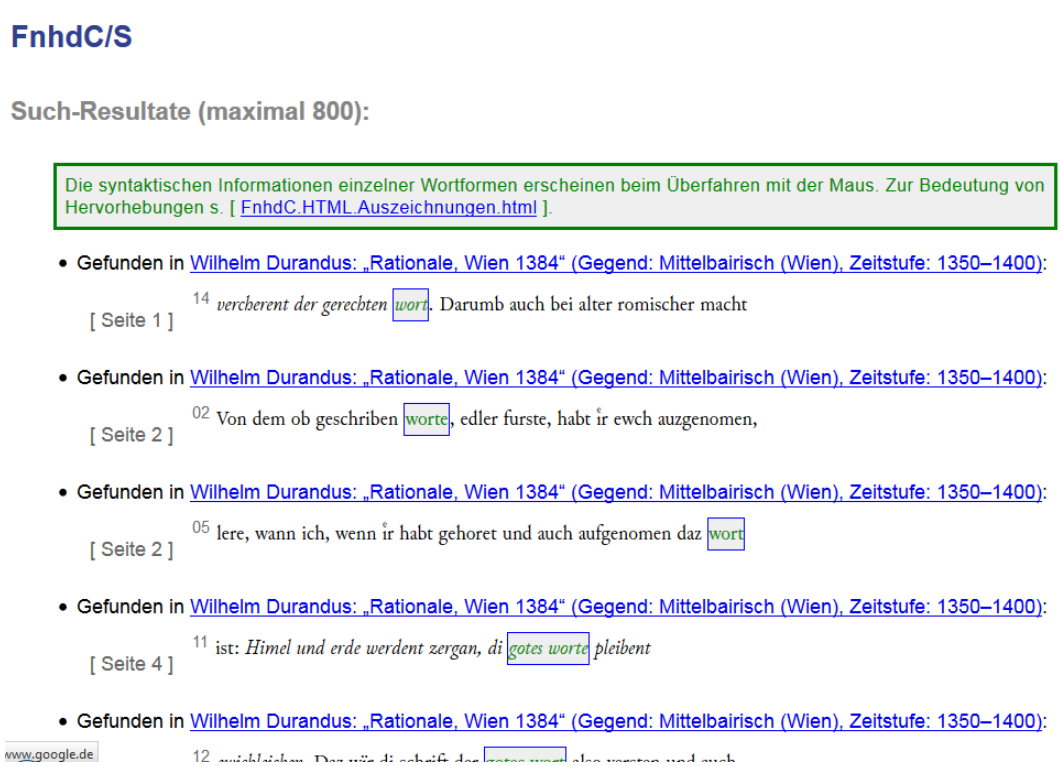
Tipp: In einigen Fällen kann es sinnvoll sein, nur in einem Teil der Dokumente zu suchen, gerade wenn man alle Belege für relativ frequente Tokens finden möchte. Das Such-Interface gibt nämlich nur max. 800 Belege aus.

Als Beispielanfrage suchen wir das Lemma „Wort“.



Wie Sie sehen, wenn Sie nach unten scrollen, sind durchaus auch komplexere Suchanfragen möglich, z.B. nach Numerus (bei Substantiven) oder Tempus und Modus (bei Verben). Für

unsere Beispielanfrage suchen wir aber einfach nach allen Belegen für das Lemma „Wort“ und klicken auf „Suchen“ rechts neben dem Lemma-Feld. Es erscheint die Ergebnisseite (eventuelle Warnmeldungen zu Javascript oder Cookies ganz oben können Sie ignorieren, solange Sie die Ergebnisse sehen):



2. Seiten-Quelltext anzeigen und exportieren

Wir müssen uns nun den Seitenquelltext anzeigen lassen. In den meisten Browsern (z.B. Firefox, Chrome) geht dies ganz einfach über Rechtsklick > Seitenquelltext anzeigen.

Hinweis: Wenn Sie über den Link auf http://www.korpora.org/fnhd auf die Seite gelangt sind und **nicht** den oben angegebenen direkten Pfad http://www.korpora.org/fnhd/Suche benutzt haben, müssen Sie sich statt des Seitenquelltexts den **Frame-Quelltext** anzeigen lassen (in Firefox unter: Rechtsklick > Aktueller Frame > Frame-Quelltext anzeigen).

Den kompletten Quelltext **kopieren** wir nun (Strg+A, Strg+C) und fügen ihn in ein neues Textdokument ein. Dafür empfehle ich den kostenlosen Editor Notepad++, der unter https://notepad-plus-plus.org/ kostenlos heruntergeladen werden kann. Für Mac ist z.B. Editra oder TextWrangler zu empfehlen, beide ebenfalls kostenlos.

Wichtig: Bitte benutzen Sie nicht Datei > Speichern, um den Seitenquelltext zu exportieren; dann funktioniert das Skript, das unten vorgestellt wird, nämlich nicht. Bitte copy&pasten Sie den Quelltext in das Textdokument.

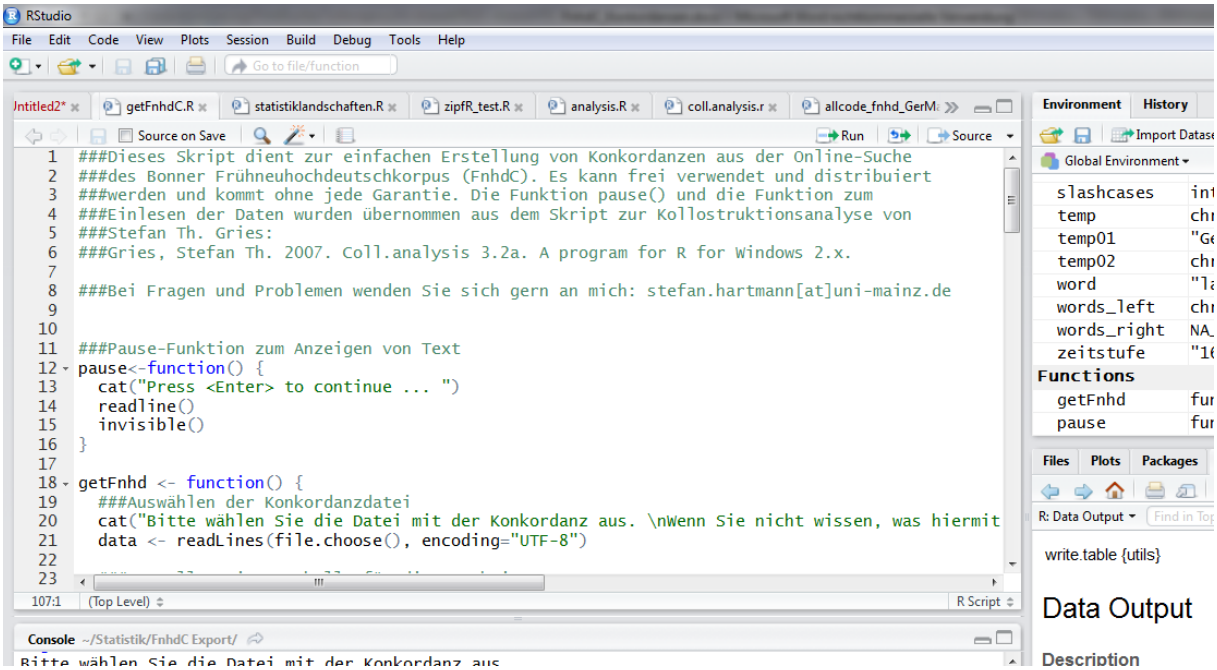
Wir speichern das Textdokument unter einem beliebigen Namen, z.B. wort.txt.

3. R herunterladen und installieren

Zum Ausführen des oben erwähnten und ganz unten hinterlegten Skripts benötigen Sie das Programm R, das nicht nur ein Statistikprogramm, sondern zugleich eine vollwertige Programmiersprache ist. Unter http://www.r-project.org/ können Sie es kostenlos herunterladen. Die Installation ist recht einfach und selbsterklärend.
Zusätzlich empfehle ich noch die Installation von R Studio, da Sie dort das Skript einfach öffnen und ausführen können. (Alternativ können Sie auch unter http://goo.gl/B3pbZM angegebenen Code einfach in R copy&pasten - bitte nicht den im Anhang stehenden Code verwenden, da dann die Zeilenumbrüche als Beginn eines neuen Codesegments interpretiert werden, was zu Fehlern führt). R Studio erhalten Sie kostenlos unter http://www.rstudio.com/.

4. Skript ausführen

Wenn Sie R Studio heruntergeladen haben, können Sie darin ganz einfach die Datei „getFnhdC.R“ öffnen. Sie wird Ihnen nun im Fenster links oben angezeigt:



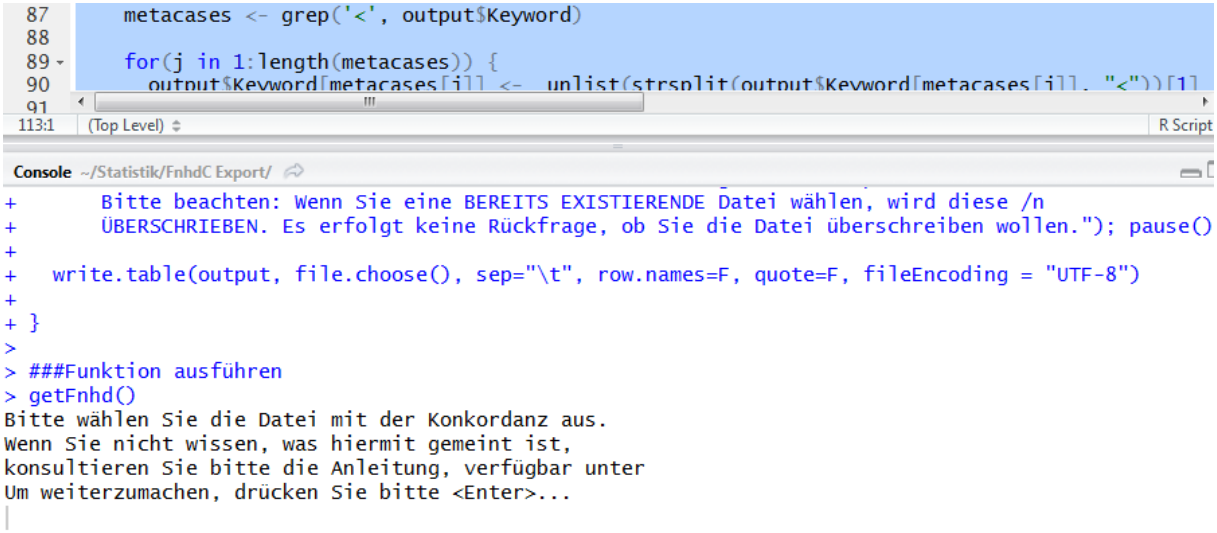
Das Fenster links oben zeigt **Skripts** an. Befehle, die in einem Skript gespeichert sind, kann man sich wie Kugeln in einer Kanone vorstellen: Sie tun nichts, solange sie nicht abgefeuert werden. Um die Befehle „abzufeuern“, müssen sie in die **Konsole** übertragen werden, die unten links zu sehen ist. Da das Skript recht lang ist, wäre es natürlich sehr mühsam, jede einzelne Zeile auszuführen. Zum Glück können wir einfach mit Strg+A den gesamten Text markieren und mit Klick auf „Run“ über dem Skriptfenster oder mit Strg+Enter komplett ausführen.

Hinter den Kulissen (Diese Passage können Sie überspringen, wenn Sie das Skript einfach nur benutzen wollen, ohne genau zu wissen, wie es funktioniert)

Das Skript besteht aus drei Teilen:
Teil 1 definiert eine Funktion *pause()*, die innerhalb der nächsten Funktion verwendet wird. Die Funktion ist aus Gries (2007) übernommen.
Teil 2 definiert die für uns relevante Funktion *getFnhdC()*.
Teil 3 besteht nur aus dem Befehl *getFnhdC()* und sorgt dafür, dass die Funktion ausgeführt wird.

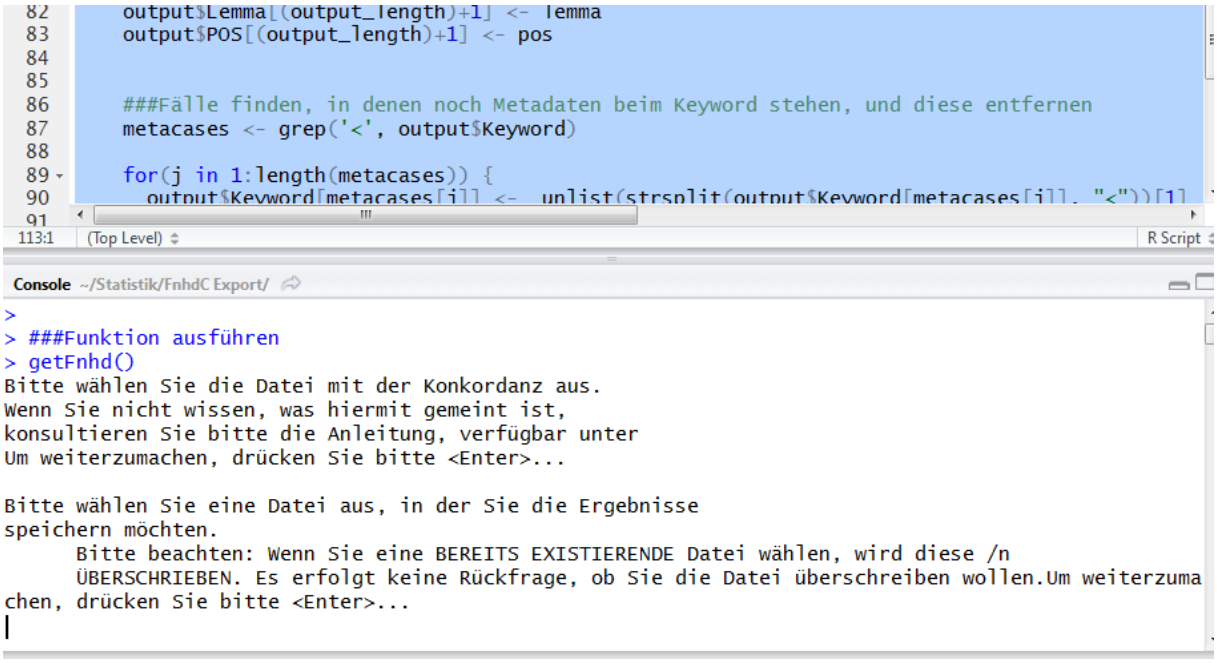
5. Daten bearbeiten

Am Ende wird die im Skript definierte Funktion automatisch ausgeführt. Nun spielt sich alles in der **Konsole** (unten links) ab. Dort werden Sie aufgefordert, eine Datei auszuwählen.



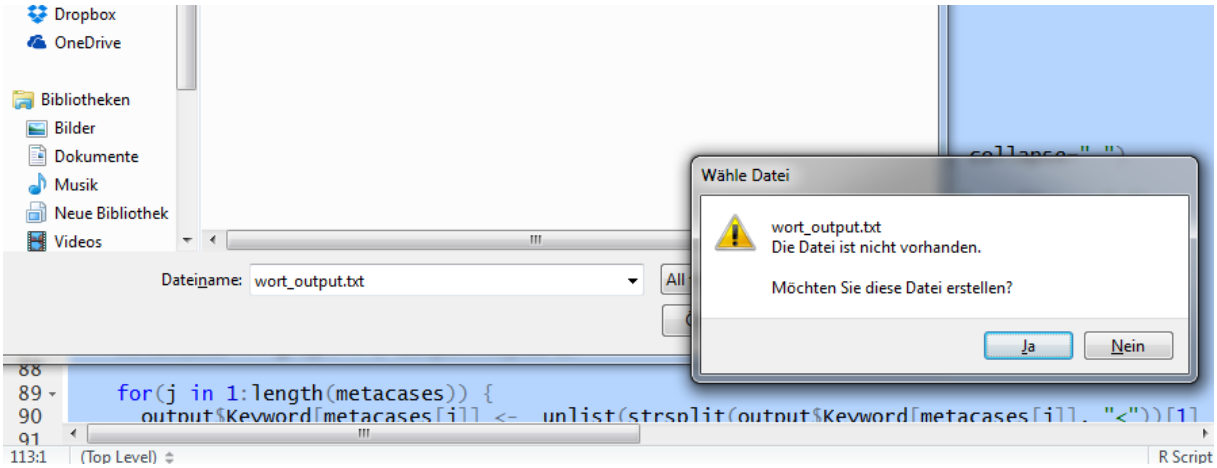
Wichtig: Bevor Sie die Eingabetaste drücken, wechseln Sie zunächst mit einem Mausklick in das Konsolenfenster, denn sonst wird der gesamte markierte Text im Skriptfenster gelöscht (wenn das passiert ist, einfach mit Strg+Z wieder rückgängig machen). Wenn Sie im Konsolenfenster die Eingabetaste gedrückt haben, öffnet sich ein Fenster, in dem Sie eine Datei aussuchen können. Navigieren Sie zu dem Ort, wo Sie wort.txt gespeichert haben, und wählen Sie es aus.

Nun rechnet R kürzere oder längere Zeit, dann erscheint wieder ein Text in der Konsole, der Sie auffordert, eine Datei auszuwählen, in der Sie den Output speichern möchten.



Nachdem Sie erneut die Eingabetaste gedrückt haben, erscheint ein Fenster, in dem Sie eine Datei auswählen können, in der die erstellte Tabelle gespeichert wird. Dazu zwei wichtige Hinweise:

1. Sie müssen die **Dateiendung** mit angeben, z.B. „wort_output**.txt**“ oder „wort_output**.csv**“.
2. Wenn Sie eine **bereits bestehende** Datei verwenden, wird diese **überschrieben!** R fragt nicht vorher nach. Hingegen fragt es nach, wenn Sie eine neue Datei erstellen:



Für unser Beispiel erstellen wir nun also die Datei wort_output.txt, die wir wiederum in Notepad++ öffnen.
(Es erscheint möglicherweise noch eine Warnung, die wir jedoch ignorieren können.)

6. Letzte Schritte

Nun sollten Sie in wort_output.txt die Belege für „Wort“ in einer tab-separierten Tabelle sehen, die Sie einfach in ein Tabellenkalkulationsprogramm wie Excel oder Calc copy&pasten können.
Hinweis: Die in FnhdC relativ häufigen Sonderzeichen gehen beim Export leider verloren, deshalb werden Sie relativ häufig Unicode-Nummern im Format [U+Nummer] in der Konkordanz finden.

Literatur

Gries, Stefan Th. (2007): Coll.Analysis 3.2a. A program for R for Windows 2.x

```
###Dieses Skript dient zur einfachen Erstellung von Konkordanzen aus der Online-Suche
###des Bonner Frühneuhochdeutschkorpus (FnhdC). Es kann frei verwendet und distribuiert
###werden und kommt ohne jede Garantie. Die Funktion pause() und die Funktion zum
###Einlesen der Daten wurden übernommen aus dem Skript zur Kollostruktionsanalyse von
###Stefan Th. Gries:
###Gries, Stefan Th. 2007. Coll.analysis 3.2a. A program for R for Windows 2.x.

###Bei Fragen und Problemen wenden Sie sich gern an mich: stefan.hartmann[at]uni-mainz.de


###Pause-Funktion zum Anzeigen von Text
pause<-function() {
  cat("Um weiterzumachen, drücken Sie bitte <Enter>... ")
  readline()
  invisible()
}

getFnhd <- function() {
  ###Auswählen der Konkordanzdatei
  cat("Bitte wählen Sie die Datei mit der Konkordanz aus.\nWenn Sie nicht wissen, was hiermit gemeint ist, \nkonsultieren Sie
bitte die Anleitung, verfügbar unter\nhttp://www.germanistik.uni-mainz.de/abteilungen/historische-sprachwissenschaft-des-
deutschen/stefan-hartmann/korpuslinguistik/\n"); pause()
  data <- readLines(file.choose(), encoding="UTF-8")

  ###Erstellen einer Tabelle für die Ergebnisse
  output <- as.data.frame(matrix(nrow=0, ncol=8))
  colnames(output) <- c("Quelle", "Gegend", "Zeitstufe", "Kontext_links", "Keyword", "Kontext_rechts", "Lemma", "POS")

  ###Auffinden der Treffer
  gefunden <- grep("<li>Gefunden in", as.character(data))
  satz <- gefunden+4

  for(k in 1:length(gefunden)) {
    ###Keyword des Treffers finden
    find_keyword <-unlist(strsplit(data[gefunden[k]], "wf=|#"))[2]

    ##Zeitstufe und Quelle des Treffers finden
    find_keyword <-unlist(strsplit(data[gefunden[k]], "wf=|#"))[2]
    temp01 <- unlist(strsplit(data[gefunden[k]], "<a href.*'>"))[2]
    quelle <- unlist(strsplit(temp01, "\\(Gegend\\:"))[1]
    temp02 <- unlist(strsplit(temp01, "\\(Gegend\\:"))
    zeitstufe <- gsub(" ", "", gsub(")", "", unlist(strsplit(temp02, "Zeitstufe\\:|<\\/a"))[3]))
    gegend <- gsub(",", "", unlist(strsplit(temp02, "Zeitstufe\\:|<\\/a"))[2])

    ###jeder einzelne Beleg als temporäres Element
    temp <- unlist(strsplit(data[satz[k]], "<span id="))
    temp <- temp[2:length(temp)]

    ##Position des Keywords im Satz
    position_keyword <- grep(find_keyword, temp)[1]

    ###finde linken Kontext
    words_left <- c()

    for(i in 1:(position_keyword-1)) {
      words_left[i] <- gsub(" ", "", gsub("<\\/span>", "", unlist(strsplit(temp[i], "\\\">"))[2]))
    }

    ###finde Keyword, Lemma und POS
    keyword <- gsub(" ", "", gsub("<\\/span>", "", unlist(strsplit(temp[position_keyword], "\\\">"))[2]))
    metadaten <- unlist(strsplit(temp[position_keyword], "\\\">"))[1]
    lemma <- gsub(" ", "", unlist(strsplit(unlist(strsplit(temp[position_keyword], "Lemma:  "))[2], "&"))[1])
    pos <- gsub(" ", "", unlist(strsplit(unlist(strsplit(temp[position_keyword], "Typ:  "))[2], "&"))[1])


    ###finde rechten Kontext
    words_right <- c()

    for(i in 1:(length(temp)-position_keyword)) {
      words_right[i] <- gsub(" ", "", gsub("<\\/span>", "", unlist(strsplit(temp[i+position_keyword], "\\\">"))[2]))
    }


    ###in Output-Tabelle eintragen
    output_length <- length(output$Quelle)
    output[(output_length+1),] <- NA

    output$Quelle[(output_length+1)] <- quelle
    output$Gegend[(output_length)+1] <- gegend
    output$Zeitstufe[(output_length)+1] <- zeitstufe
    output$Kontext_links[(output_length)+1] <- paste(words_left, sep=" ", collapse=" ")
    output$Keyword[(output_length)+1] <- keyword
    output$Kontext_rechts[(output_length)+1] <- paste(words_right, sep=" ", collapse=" ")
    output$Lemma[(output_length)+1] <- lemma
    output$POS[(output_length)+1] <- pos


    ###Fälle finden, in denen noch Metadaten beim Keyword stehen, und diese entfernen
    metacases <- grep('<', output$Keyword)

    for(j in 1:length(metacases)) {
      output$Keyword[metacases[j]] <-  unlist(strsplit(output$Keyword[metacases[j]], "<"))[1]
    }

    ###Interpunktion aus Keywords entfernen
    output$Keyword <- gsub("[[:punct:]]", "", output$Keyword)



  }

  cat("Bitte wählen Sie eine Datei aus, in der Sie die Ergebnisse \nspeichern möchten. Bitte beachten: Wenn Sie eine BEREITS
EXISTIERENDE Datei wählen, wird diese \nÜBERSCHRIEBEN. Es erfolgt keine Rückfrage, ob Sie die Datei überschreiben wollen.");
pause()
```