Text mining in SAS Programmen: Explorative Anwendung von Text mining Verfahren auf SAS Programme und Evaluation des Nutzens

Jörg Sahlmann
iOMEDICO AG
Ellen-Gottlieb-Straße 19
79106 Freiburg
joerg.sahlmann@iomedico.com

Zusammenfassung

Text mining - Verfahren werden heute in vielen Bereichen angewendet. Korpus, Index oder n-Gramme sind nur ein paar Stichworte, die in diesem Zusammenhang genannt werden können.
Was passiert, wenn man Text mining-Verfahren auf SAS-Programme anwendet? Welchen Nutzen kann man aus diesen Ergebnissen ziehen?
Diese Arbeit beschreibt, wie man die Verfahren strukturiert anwendet. Das Hauptaugenmerk ist dabei auf die sinngemäße Erstellung eines Korpus, eines Dictionary und einer n-Gramm-Liste für ein oder mehrere SAS Programme gerichtet.
Mögliche Anwendungsgebiete können die Fehlersuche in Programmen durch die Identifikation isoliert vorkommender Variablen (Schreibfehler im Namen) oder die Dokumentation von Programmen (Identifikation benutzter Prozeduren) sein.

Schlüsselwörter: Korpus, Text Mining, SAS Code, Dictionary

1 Definitionen

In dieser Arbeit werden die folgenden Begriffe mit Leben gefüllt und ihre Bedeutung auf den Programmcode übertragen.

- Korpus
- Index
- Konkordanz
- Dictionary/Wörterbuch
- n-Gramm

1.1 Korpus

Der Korpus ist die Sammlung der Rohtexte, die der Textanalyse zugrunde liegen. Diese können in unterschiedlichen Formaten (txt, html, pdf, xml) und in unterschiedlichen Kodierungen (ANSI, UTF-8) vorliegen.

1.2 Index

Der Index ist die Liste der Wörter, die in den Texten vorkommt. Er ist in der Regel sortiert und von Dubletten bereinigt. Er kann um die Häufigkeit des Vorkommens erweitert werden.

1.3 Konkordanz

Die Konkordanz ist ein Verzeichnis, das für jedes Wort aus dem Index aufzeigt, in welchem Text und in welcher Zeile es vorkommt.

1.4 Dictionary/Wörterbuch

Das Dictionary ist eine anwender-/anwendungsspezifische Liste von Wörtern, die mit den Texten verglichen werden kann, ob und wie oft Wörter daraus vorkommen.

1.5 n-Gramme

n-Gramme sind Sequenzen von n aufeinander folgenden Wörtern, die als Sequenz gelistet und gezählt werden.

2 Übertragung auf SAS Programmcode

2.1 Korpus

Übertragen auf SAS Programme ist der Korpus die Gesamtheit dieser Programme für ein Projekt. Sie liegen als ASCII-Dateien mit der Endung sas vor.

Da diese Endung keine übliche Endung für Textdateien ist, muss bei solchen Programmen wie AntConc, die Dateien zur Analyse einlesen, die Defaulteinstellung so geändert werden, dass auch sas-Dateien als relevant erkannt werden.

2.2 Index

Für die Liste der relevanten Wörter in einem SAS-Programm werden in einem Vorbereitungsschritt alle Stringkonstanten ausgelesen und durch fixe Strings ersetzt, da die Stringkonstanten sich nicht auf die funktionelle Struktur des Programmes auswirken.

Sie können in einer zweiten Liste aufgeführt werden, um ggf. für die Internationalisierung von Programmen benutzt zu werden.

Für Kommentare gilt dasselbe wie für Stringkonstanten. Sie werden ebenfalls durch fixe Strings ersetzt.

Im Gegensatz zu normalen Textanalysen, bei denen ein gültiges Wort nur die Zeichen A bis Z in Groß- und Kleinschreibung umfasst, gilt für die Wörter in SAS-Programmen ein erweiterter Zeichenvorrat. Unter der Annahme, dass die Option

VALIDVARNAME=V7 ist, beinhaltet der Zeichenvorrat Buchstaben, Zahlen und den Unterstrich (_). Da durch die LIBNAMES auch Punkte (.) als Wortverbinder vorkommen, gehören sie ebenso wie das Prozentzeichen (%) und das Ampersand (&) zu dem gültigen Zeichenvorrat.

2.3 Konkordanz

Die Konkordanz ist ein Verzeichnis, das für jedes Wort aus dem Index aufzeigt, in welchem SAS-Programm und in welcher Programmzeile es vorkommt.

2.4 Dictionary/Wörterbuch

Mögliche Wörterbücher sind Listen der reservierten SAS-Wörter oder Listen vorgegebener Variablen, die bestimmten Standards genügen (z.B. CDISC SDTM Variablen).

2.5 n-Gramme

Typische 2-Gramme sind die Prozeduraufrufe mit PROC+Schlüsselwort oder DATA+Datensatz.

Unter 4-Grammen kann man die Prozeduraufrufe mit PROC+Schlüsselwort und dem zugehörigen Datensatz betrachten.

Obwohl ein Semikolon das logische Ende einer zusammengehörigen Folge von Wörtern bildet, kann es sinnvoll sein, n-Gramme auch über die Semikolongrenze hinweg zu betrachten. Ein PROC oder DATA an zweiter Position eines 2-Gramms ohne ein vorheriges RUN oder QUIT an der ersten Position kann einen Hinweis auf einen fehlenden Befehl darstellen.

3 Benutzte Werkzeuge

Der SAS-Editor bietet wenig Unterstützung für die Erstellung der oben aufgeführten Konstrukte und Datenstrukturen. Hierfür sind entsprechende Anwendungsprogramme oder selbst erstellte Programme in unterschiedlichen Programmiersprachen (SAS, Python) notwendig.

3.1 Notepad++

Notepad++ (NPP) zusammen mit dem Plugin TextFX eignet sich sehr gut, um schnell eine Liste aller vorkommenden Wörter in einem Skript zu erstellen. Dazu werden alle Leerzeichen durch Zeilenumbrüche ersetzt und dann wird der ganze Text markiert und mit dem unique sort Befehl von TextFX alphabetisch sortiert. Fertig ist die Wortliste. Sie enthält natürlich noch alle Sonderzeichen und ggf. Reste der Silbentrennung, aber für einen ersten Überblick ist sie gut geeignet.
Leider wird TextFX seit 2009 nicht mehr gepflegt und die Zukunft dieses an Funktionen mächtigen Plugins ist offen.

3.2 Python

Mit Python lassen sich viele der sonst händisch durchzuführenden Schritte automatisch per Skript durchführen. Hierauf wird im Anwendungsbeispiel genauer eingegangen.

3.3 AntConc

AntConc ist ein kompaktes Programm zur Analyse eines Textkorpus. Es erstellt sehr einfach Wortlisten, Konkordanzen und n-Gramme. Durch die Konfiguration des Zeichenvorrates ist es auch für die Analyse von SAS-Code geeignet.

4 Anwendungsbeispiel

Grundlage des Beispiels ist ein Ordner, in dem sich SAS-Programme zur Erstellung von SDTM-Datensätzen für eine Studienauswertung befinden. Die Abfolge einer möglichen Textanalyse wird schrittweise beschrieben. Pythonskripte werden in Auszügen präsentiert und die Bearbeitung mit AntConc wird skizziert.

4.1 Entfernung von Stringkonstanten und Kommentaren

Kommentare und Stringkonstanten haben keinen Einfluss auf die Struktur der Programme und werden entfernt. Eine mögliche Lösung, die im vorliegenden Python-Skript-Auszug beschrieben wird, ist die Suche nach den folgenden Strukturen mit Hilfe von regulären Ausdrücken:

- /** … */
- /* … */
- " … "
- %* …;
- * …;

```
options1 = re.IGNORECASE | re.VERBOSE | re.DOTALL
options2 = re.IGNORECASE | re.VERBOSE
# /** ... */
data = re.sub(r"/\*\*.*?\*/", "#", data, 0, options1)
# /* ... */
data = re.sub(r"/\*[^\*].*?\*/", "#", data, 0, options1)
# " ... "
data = re.sub(r"'.*?'", '"#"', data, 0, options1)
# ' ... '
data = re.sub(r'".*?"', "'#'", data, 0, options1)
# %* ... ;
data = re.sub(r"%{1,1}\*.*?;", "#", data, 0, options1)
# * ... ;
data = re.sub(r"\*.*?;", "#", data, 0, options2)
```

Einzelne Zeichen wie Klammern, Kommata und Semikola werden ebenfalls angepasst, um eine Aufteilung des Programmcodes in Einzelwörter zu vereinfachen. Das Leerzeichen kann als Trenner benutzt werden.

```
data = re.sub(r"\(", " (  ", data, 0, options1)
data = re.sub(r"\)", " )  ", data, 0, options1)
data = re.sub(r"\[", " [  ", data, 0, options1)
data = re.sub(r"\[", " ] ",  data, 0, options1)
data = re.sub(r"\{", " { ", data, 0, options1)
data = re.sub(r"\}", " } ", data, 0, options1)
data = re.sub(r"=", " = ", data, 0, options1)
data = re.sub(r",", " , ", data, 0, options1)
data = re.sub(r";", " , ", data, 0, options1)
```

Die modifizierten Programmzeilen werden in neue Dateien mit der Endung txt geschrieben. Dieser Arbeitsschritt kann mit AntConc nicht durchgeführt werden. AntConc greift deswegen auf die modifizierten Dateien zu. Abbildung 1 zeigt den Hauptbildschirm. Die zwei Dateien, die sich aktuell im Korpus befinden, werden in der linken Spalte dargestellt.

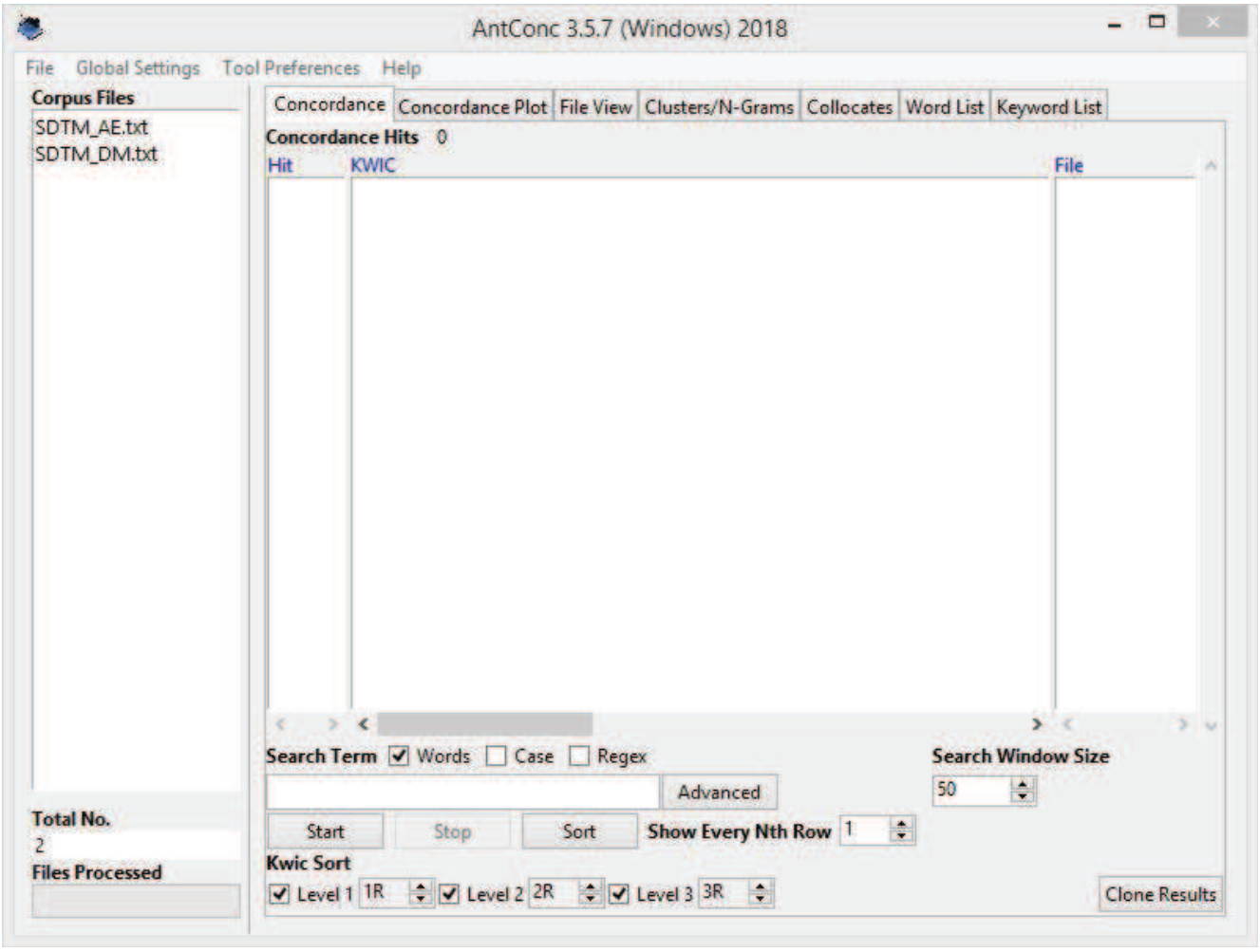

Abbildung 1: AntConc: Hauptfenster links mit den zwei Dateien im Korpus

4.2 Erzeugung des Index

Die im vorigen Schritt erzeugten modifizierten Programmzeilen werden anhand der Leerzeichen aufgetrennt. Daraus ergibt sich eine Liste L1 mit der Sequenz der vorkom-

menden Wörter. Diese Liste L1 wird für die Konstruktion der n-Gramme benötigt. Für die Erzeugung der Wortliste wird L1 der Wörter wird dann alphabetisch sortiert und von Dubletten bereinigt. Daraus ergibt sich die Liste L2.

```
L1 = re.split(r"\s", data, 0, options2)
L2 = sorted(set(L1))
```

Bereinigt werden beide Listen um numerische Konstanten, um einzelne Sonderzeichen und um sogenannte Stop-Wörter, also die Wörter, die nicht im Index und in den n-Grammen berücksichtigt werden sollen.

In AntConc findet sich der Index auf dem Reiter Word List. In Abb. 2 werden die Wörter in der Reihenfolge ihrer Häufigkeit angezeigt. Wörter, die nur einmal vorkommen, kann man bei alphabetischer Sortierung der Wörter mit den dann benachbarten Wörtern vergleichen und auf ggf. fehlerhafte Schreibung prüfen.

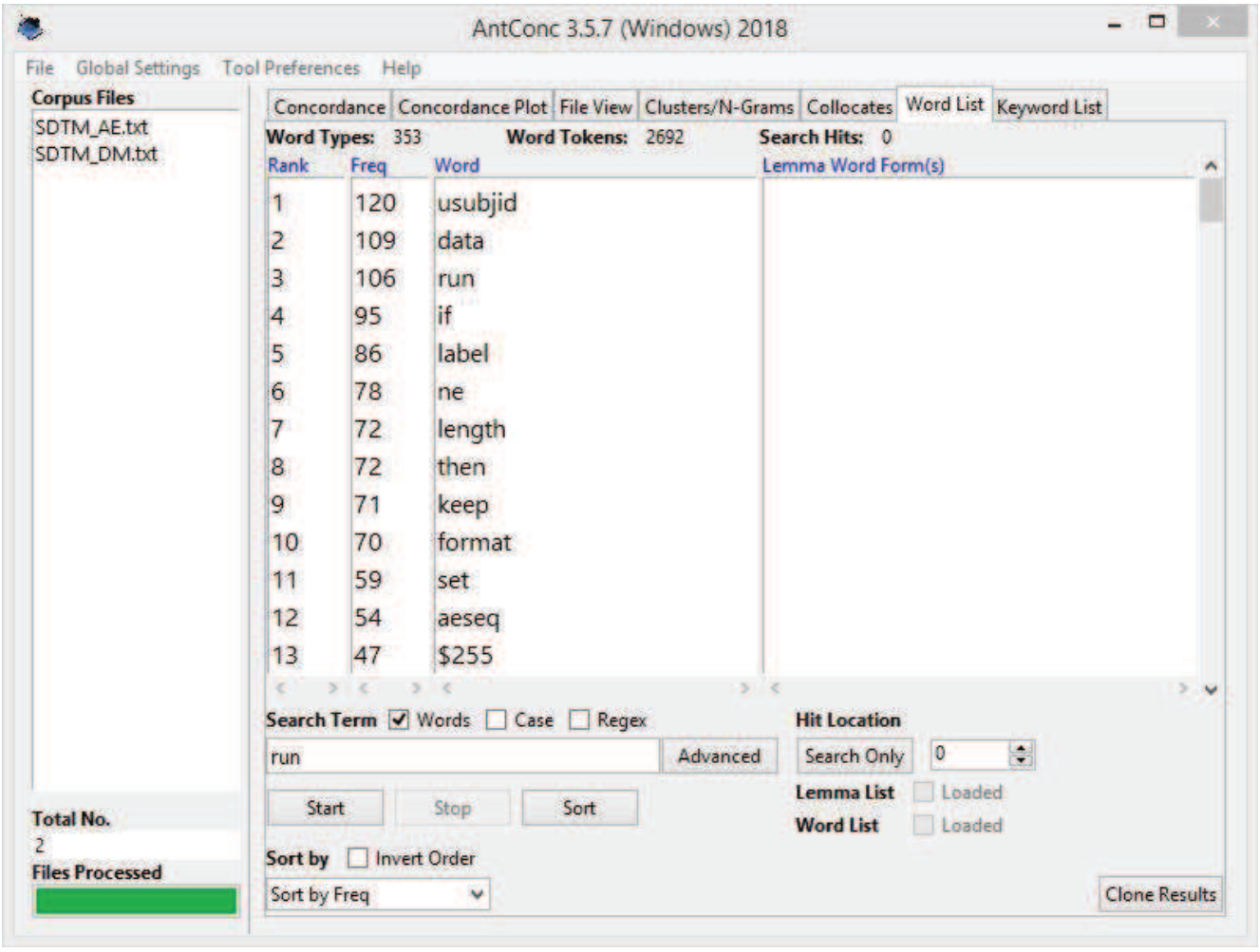

Abbildung 2: AntConc: Hauptfenster mit der Wortliste von beiden Dateien

Im Menü Global Settings wurde der Zeichenvorrat, aus dem die Wörter gebildet werden dürfen, um den Unterstrich, den Punkt, das Dollarzeichen und das Prozentzeichen erweitert.

4.3 Erzeugung der Konkordanz

Die Konkordanz wird aufgebaut, indem alle Dateien des Korpus nach dem Vorkommen eines jeden Wortes der Liste L2 durchgesucht und die Fundorte aufgelistet werden. Beispielsweise kann die folgende Funktion über alle Dateien und alle Schlüsselworte iteriert werden.

```
def grep_file(grep_file, grep_word):
    with open(grep_file, 'r') as mysasfile:
        lineno = 0
        for line in mysasfile:
            lineno = lineno + 1
            if re.search(grep_word, line, re.IGNORECASE):
                print(grep_file, "-", lineno, ":", line, end = '')
```

In AntConc wird die Konkordanz auf dem Reiter Concordance dargestellt. In Abb. 3 finden sich alle Vorkommen des Wortes PROC mit den darauf folgenden Wörtern, wobei das zweite, dritte und viere Wort farblich unterschiedlich gekennzeichnet ist.

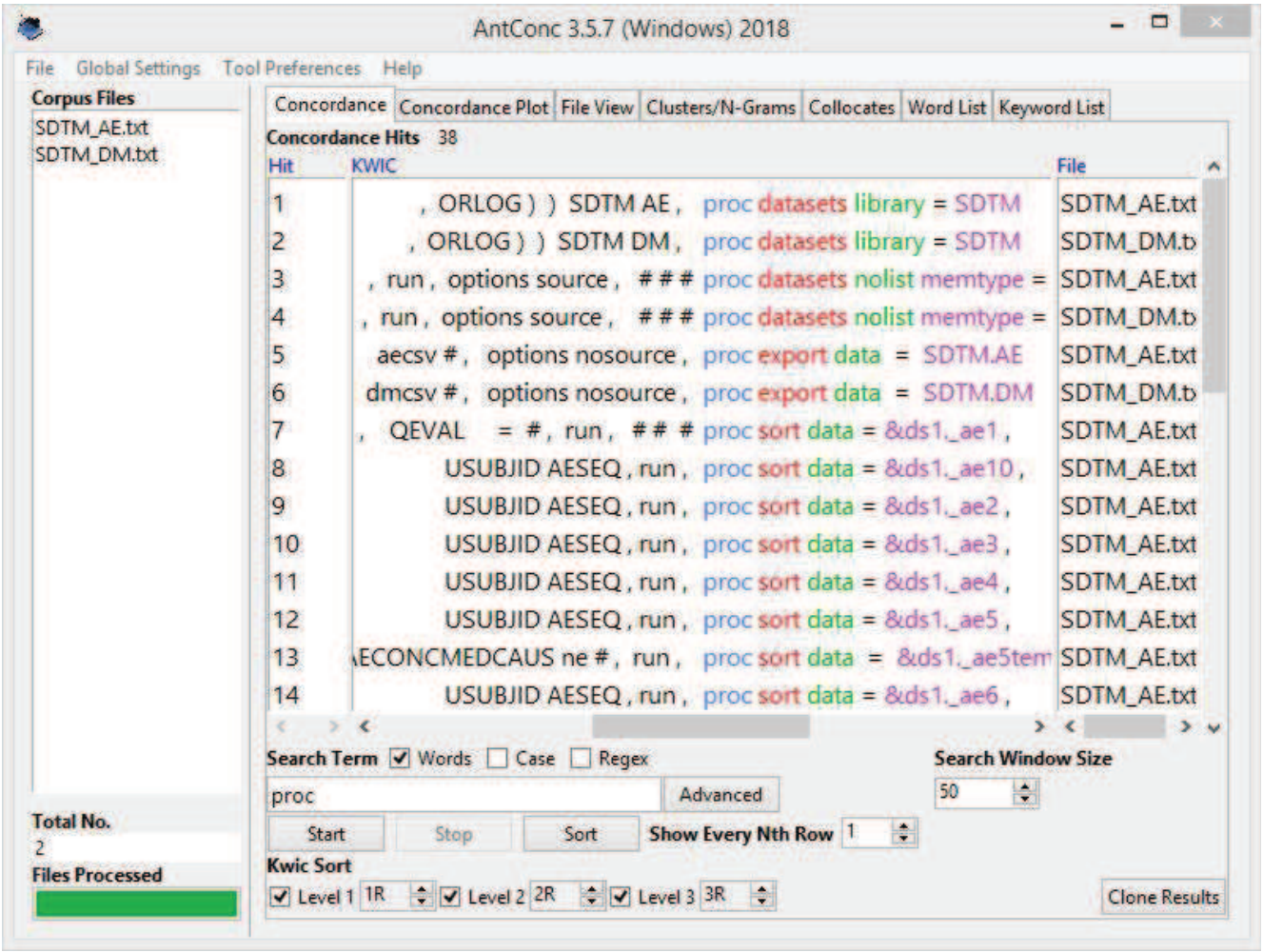

Abbildung 3: AntConc: Hauptfenster mit den Vorkommen für das Wort PROC

4.4 Nutzung individueller Wortlisten

Der Abgleich gegen individuelle Wortlisten erfolgt sinngemäß wie im vorigen Abschnitt. Statt der Liste L2 wird die individuelle Wortliste benutzt. In AntConc lassen sich Wordlisten manuell wortweise erstellen oder aus einer Datei einlesen.

In Abbildung 4 wurden drei Wörter eingegeben und in Abbildung 5 werden die zugehörigen Häufigkeiten angezeigt. Bei Doppelklick auf eines der Suchergebnisse wird eine gleichartige Ergebnisliste wie in Abbildung 3 angezeigt.

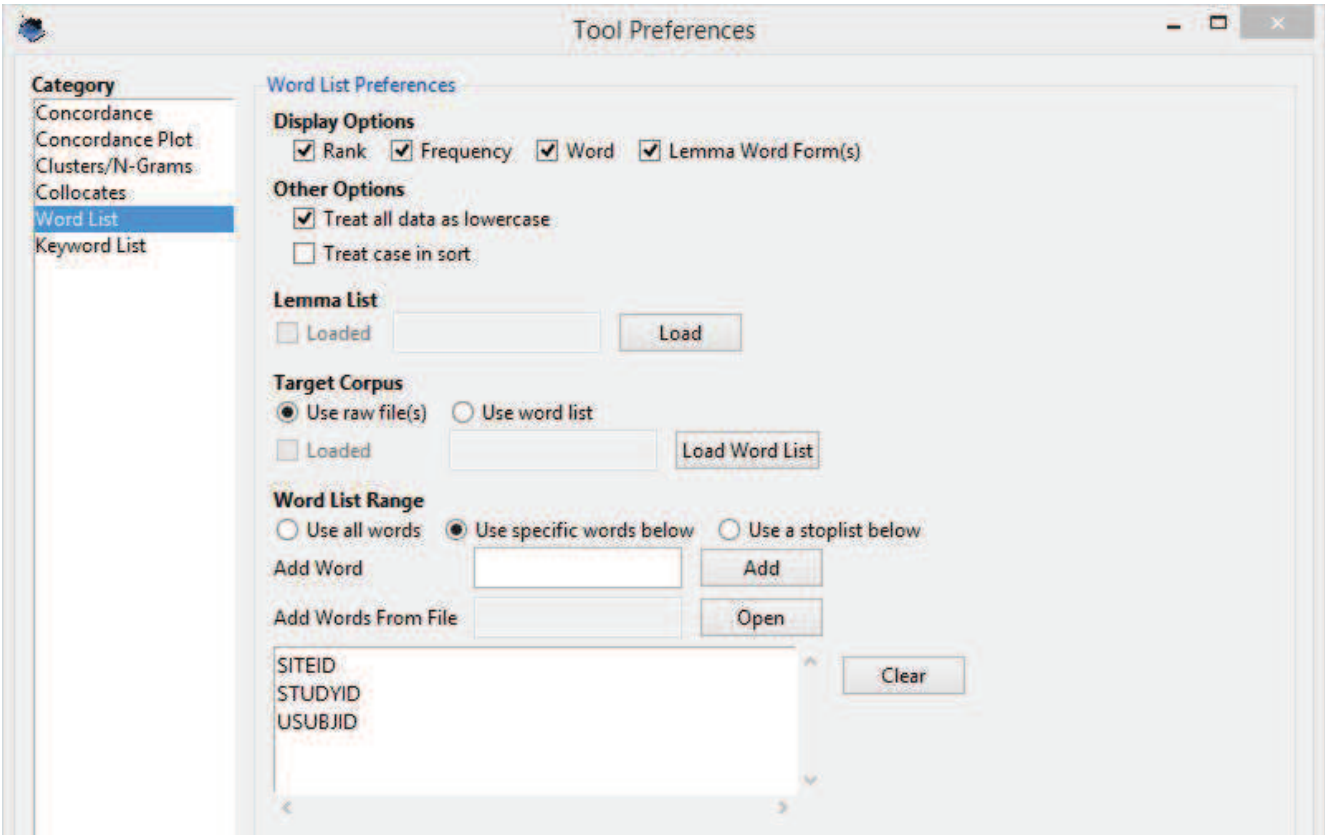

Abbildung 4: AntConc: Voreinstellungen mit drei manuell eingegebenen Wörtern

4.5 Konstruktion benutzter n-Gramme

Die n-Gramme der Dateien im Korpus werden konstruiert, indem über die Liste L1 ein Leseraster der Länge n gelegt wird.

L1: w1 w2 w3 w4 w5 w6 …

Das erste 3-Gramm ist w1w2w3, das zweite 3-Gramm ist w2w3w4, das vierte 3-Gramm ist w3w4w5 usw.

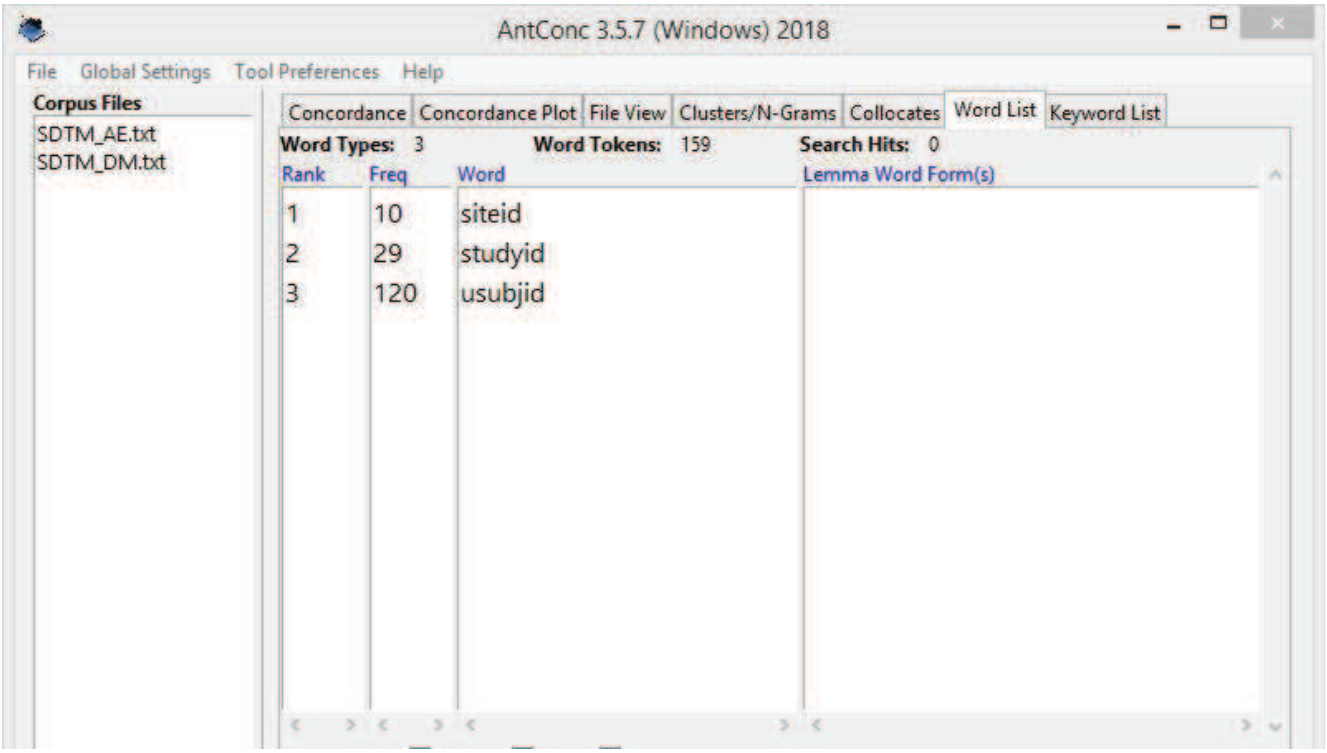

Abbildung 5: AntConc: Häufigkeiten für die drei vorgegeben Wörtern

```
def create_2_gramm(grep_file):
    with open(grep_file, 'r') as mysasfile:
        lineno = 0
        w1 = ""
        w2 = ""
        for line in mysasfile:
            lineno = lineno + 1
            if lineno == 1:
                w2 = line.strip()
            elif lineno > 1:
                w1 = w2
                w2 = line.strip()
                print(w1, w2, end = '\r\n')
```

Die n-Gramme werden bei AntConc auf dem Reiter Clusters/N-Grams dargestellt. Das n lässt sich im unteren Teil der Dialogbox einstellen.

Abbildung 6 zeigt die 2-Gramme in alphabetischer Reihenfolge. Es werden vier verschiedene PROCs genutzt, wobei PROC SORT am häufigsten vorkommt.

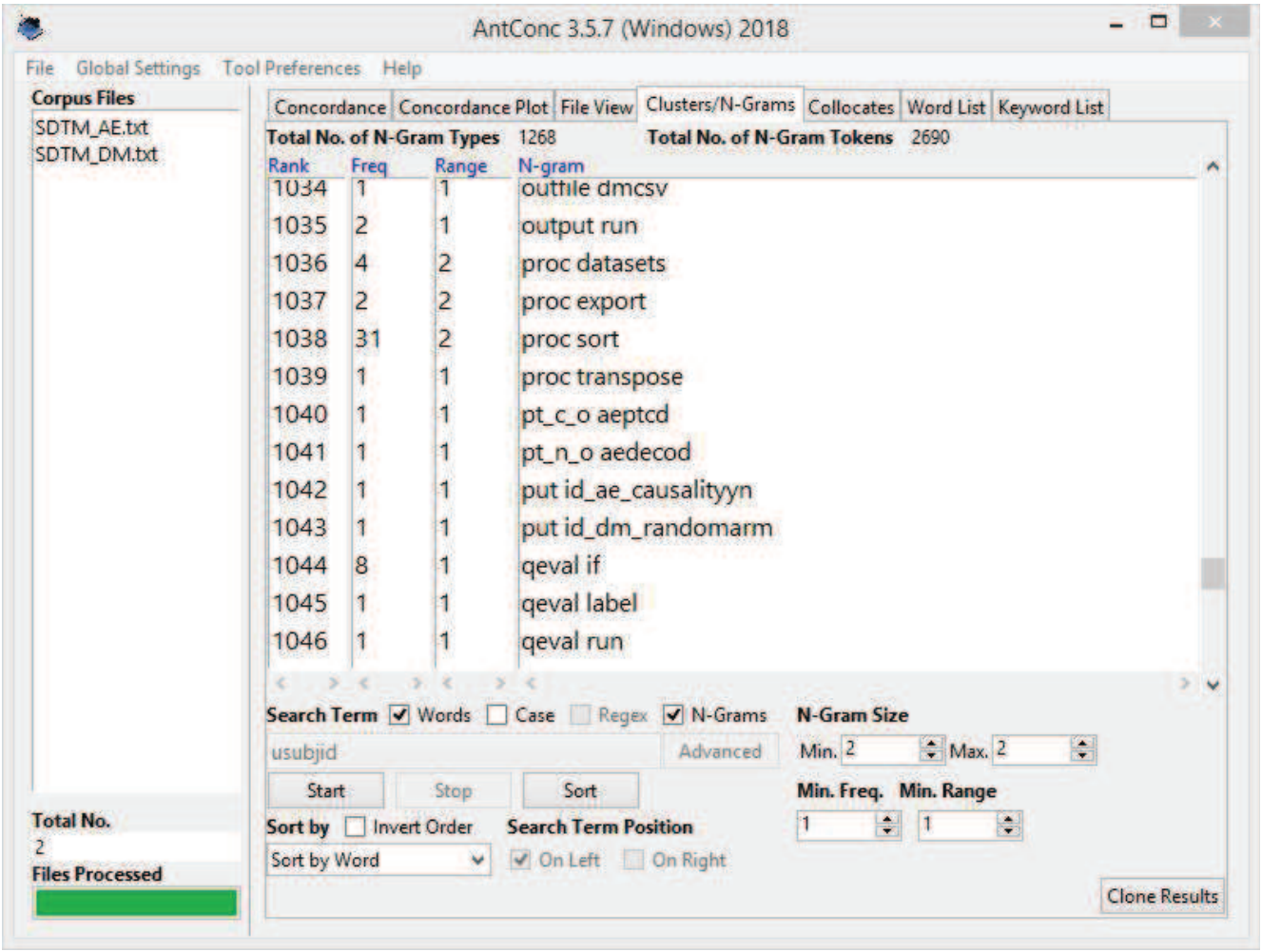

Abbildung 6: AntConc: 2-Gramme zeigen die Nutzung vier verschiedene PROCs

5 Zusammenfassung

Verschiedene Text mining Verfahren lassen sich auf SAS-Programme anwenden. Die Ergebnisse der einzelnen Verfahren erscheinen sinnvoll im Bereich Fehlersuche und Dokumentation und können eine Bereicherung des Werkzeugrepertoires des SAS-Programmierers darstellen. (Danksagung: Die Wiedergabe der Screenshots erfolgt mit freundlicher Genehmigung von Professor L. Anthony, Tokio)

Literatur

[1] L. Anthony (2018): AntConc (Version 3.5.7). Tokyo, Japan: Waseda University. http://www.laurenceanthony.net/software/antconc/, letzter Zugriff 15.01.2019.

[2] Notepad++, https://notepad-plus-plus.org/, letzter Zugriff 15.01.2019.

[3] TextFX, Plugin für Notepad++, https://sourceforge.net/projects/npp-plugins/files/TextFX/, letzter Zugriff 15.01.2019.