Webscraping für MAXQDA mit Python

Thomas Zapf-Schramm, 26. August 2020

In diesem Dokument wird beschrieben, wie mithilfe von Python die Beiträge eines Internetforums heruntergeladen werden können, um diese in MAXQDA zu importieren. Weitere Hinweise zu dem hier vorgestellten Beispiel finden sich im PDF-Dokument "Webscraping für MAXQDA mit R".

Table of Contents

Warum Python?

Python ist eine der verbreitetsten und beliebtesten Programmiersprachen. Sie hat eine klare und strukturierte Syntax und ist (relativ) leicht erlernbar. Ähnlich wie die Sprache R stellt sie fertige Module für wissenschaftliche Aufgaben, insbesondere auch für sprachwissenschaftlich Ansätze und Methoden zur Verfügung (nltk, spaCy), aber auch für numerische Probleme (numpy) und Datentransformationen (pandas). Es gibt auch eine große Zahl fertiger Module für viele andere Bereiche der Datenverarbeitung und -analyse, z.B. für Webscraping.

Man kann Python auf verschieden Wegen beziehen und installieren. Für wissenschaftliche Zwecke ist die Distribution "anaconda" besonders interessant, da in ihr praktisch alles enthalten ist, was hier normalerweise benötigt wird. Fehlende Pakete können leicht nachinstalliert werden.

Pythonprogramme können unter Windows mit dem mitgelieferten Editor *"Idle"* geschrieben und ausgeführt werden, ansonsten auf allen gängigen Betriebssystemen mit jedem beliebigen Editor und dem Python-Interpreter. Komfortabeler ist die Nutzung einer kommerziellen Entwicklungsumgebung (wie *PyCharm*) oder -- wie hier -- *Jupyter Notebooks*. Jupyter ist eine Python-Entwicklungsumgebung, die im Webbrowser läuft und das Schreiben und schrittweise Ausführen von Code ermöglicht. Code und Ergebnisse erscheinen dabei gemischt in "Zellen" auf einer Seite. Dazwischen können Zellen mit beschreibendem Text im Markdown-Format eingefügt werden.

Einrichtung der benötigten Werkzeuge

Am Beginn unseres Python-Programms importieren wir alle Module, die wir benötigen werden. **request** enthält Methoden zum Abruf von Informationen aus dem Internet, **lxml**, **html** und **cssselect** helfen bei der Verarbeitung der heruntergeladenen Inhalte, **pandas** ist hier das zentrale Modul zur Verarbeitung unserer generierten Datentabellen, **locale** und **platform** brauchen wir zur Behandlung von Problemen mit unterschiedlichen Zeichencodierungen im Internet und auf den verschiedenen Betriebssystemen, **regex** stellt Funktionen zur Textsuche- und Manipulation bereit. Von **numpy** brauchen wir nur eine kleine Funktion für bedingte Berechnungen in Datentabellen ("where").

In [ ]:

```
from lxml import html  as htm
from html import escape
from pandas import DataFrame
from pandas import to_datetime
from pandas import read_excel
from numpy import where
import cssselect
import requests as rq
import locale
import platform
import regex as re
```

Datendownload

Download der Basisseiten des Forums

Im ersten Schritt legen wir die Wurzel-URL des MAXQDA Technikforums fest. Um ein anderes der USER-Foren zu laden müssen wir nur die Zahl am Ende dieser URL ändern (24 für Funktionswünsche, 21 für Forschung).

Danach erzeugen wir einen leeren DataFrame, d.h. eine leere Tabelle, mit dem Namen "df_base_pages".

In diese Tabelle fügen wir eine Spalte für die URLS der Basisseiten ein und füllen sie automatisch mit den entsprechenden URLs. Die Adresse einer Seite setzt sich hier aus der Adresse des Foruns und einem Offset zusammen. Jede Seite enthält 50 Threads, so dass bei den 8 vorhandenen Seite die Offsets 0, 50, 100, ... 400 lauten.

In die zweite Spalte dieser Tabelle speichern wir den HTML-Quelltext der Seiten, den wir mit **request.get().text** herunterladen.

Anmerkung für Interessierte: Die Befehle in den eckigen Klammern verarbeiten Listen in Schleifen und werden als *list comprehensions* bezeichnet.

Zuletzt drucken wir den gefüllten DateFrame am Bildschirm. Dabei wird er automatisch so gekürzt und aufgeteilt, dass der Bildschirm nicht überfüllt wird.

In [13]:

```
root_url = 'https://www.maxqda.de/support/forum/viewforum.php?f=23'

df_base_pages = DataFrame()
df_base_pages['base_page_url'] = [root_url + '&start=' + str(x) for x in range(0, 400,
50)]
df_base_pages['html'] = [rq.get(x).text for x in df_base_pages.base_page_url]
print(df_base_pages)
```

```
base_page_url  \
0  https://www.maxqda.de/support/forum/viewforum....
1  https://www.maxqda.de/support/forum/viewforum....
2  https://www.maxqda.de/support/forum/viewforum....
3  https://www.maxqda.de/support/forum/viewforum....
4  https://www.maxqda.de/support/forum/viewforum....
5  https://www.maxqda.de/support/forum/viewforum....
6  https://www.maxqda.de/support/forum/viewforum....
7  https://www.maxqda.de/support/forum/viewforum....

                                                html
0  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 S...
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 S...
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 S...
3  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 S...
4  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 S...
5  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 S...
6  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 S...
7  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 S...
```

Download der Threadseiten

Jede der 8 geladenen Basisseiten (bis auf die Letzte) enthält Links zu 50 Unterseiten, die die Threads des Forums enthalten. Die Adressen dieser Seiten müssen -- zusammen mit einigen Basisinformationen zu den Threads -- extrahiert werden und in eine neue Tabelle namens "df_threads" eingefügt werden. Diese Tabelle wird am Ende also so viele Zeilen haben, wie Threads im Forum existieren.

- Wir können leider keine absolute URL der Threads auf den Basisseiten finden, sondern nur relative URLs zur Basis des USER Forums. Deshalb müssen wir die gefundenen URL-Fragmente um die "base_url" erweitern.
- Auf den base_pages finden sich neben den Thread-URLs auch Angeben zu den Titeln der Themen, zum Umfang der jeweiligen Diskussionen und zu den Abrufzahlen der Threads. Auch diese Informationen extrahieren wir und schreiben sie in unsere Tabelle. Die Extraktion erfolgt mithilfe von "CSS-Selektoren", wie sie manchen MAXQDA-Nutzer*innen vielleicht aus der Gestaltung eigener Websites bekannt sind.
- Außerdem fügen wir noch die URL der Basisseite hinzu, auf der wir den Thread gefunden haben.
- Als letztes fügen wir das Wichtigste in die Tabelle ein: Den kompletten HTML-Quelltext der Thread-Seite.

Um zm gewünschten Ergebnis zu gelangen, gehen wir folgenderma0en vor:

- Wir erzeugen zunächst wieder eine leere Tablle "df_threads".
- Dann durchlaufen wir unsere acht base_pages (mit iterrows).
- Für jede Seite legen wir eine Zwischentabelle "df_tmp" an, in die wir die auf der Seite gefundenen Informationen eintragen.
- Zuletzt fügen wir den Inhalt der temporären Tabelle an die Tabelle "df_threads" an.

Wenn die 8 "base_pages" durchlaufen sind ist die Tabelle "df_threads" mit 354 Thread-Zeilen gefüllt.

Weil innerhalb dieser Prozedur die 354 Thread-Seiten heruntergeladen werden, dauert sie -- je nach Bandbreite der Internetverbindung -- einige Minuten. Alle anderen Teile des Programms laufen innerhalb von Sekunden. Um die Kontrolle über den Stand der Programmausführung zu erhalten, drucken wir jeweils die URL der gerade bearbeiteten Basisseite am Bildschirm aus.

In [15]:

```
base_url = 'https://www.maxqda.de/support/forum'

df_threads = DataFrame()
for index, row in df_base_pages.iterrows():
    print(row.base_page_url)
    doc = htm.document_fromstring(row.html)
    df_tmp = DataFrame()
    df_tmp['thread_url'] = [base_url + e.get('href')[1:] for e in doc.cssselect('li.row
dt a.topictitle')]
    df_tmp['title'] = [e.text for e in doc.cssselect('li.row dt a.topictitle')]
    df_tmp['posts'] = [e.text for e in doc.cssselect('li.row dd.posts')]
    df_tmp['views'] = [e.text for e in doc.cssselect('li.row dd.views')]
    df_tmp['base_page_url'] = row.base_page_url
    df_tmp['html'] = [rq.get(x).text for x in df_tmp.thread_url]
    df_threads = df_threads.append(df_tmp)
print(df_threads.head())
```

```
https://www.maxqda.de/support/forum/viewforum.php?f=23&start=0
https://www.maxqda.de/support/forum/viewforum.php?f=23&start=50
https://www.maxqda.de/support/forum/viewforum.php?f=23&start=100
https://www.maxqda.de/support/forum/viewforum.php?f=23&start=150
https://www.maxqda.de/support/forum/viewforum.php?f=23&start=200
https://www.maxqda.de/support/forum/viewforum.php?f=23&start=250
https://www.maxqda.de/support/forum/viewforum.php?f=23&start=300
https://www.maxqda.de/support/forum/viewforum.php?f=23&start=350
                                          thread_url  \
0  https://www.maxqda.de/support/forum/viewtopic....
1  https://www.maxqda.de/support/forum/viewtopic....
2  https://www.maxqda.de/support/forum/viewtopic....
3  https://www.maxqda.de/support/forum/viewtopic....
4  https://www.maxqda.de/support/forum/viewtopic....

                                              title posts views  \
0  Spalten von Excel-tabelle werden nicht komplet...     1   276
1  Codeliste wird nicht angezeigt im Creative Coding     3    35
2  MAXQDA erkennt den Text in einigen PDF-Dateien...     2   156
3                  Nicht codierte Passagen anzeigen     2   126
4      Zusammenarbeit (Teamwork) beim Transkribieren     1   125

                                       base_page_url  \
0  https://www.maxqda.de/support/forum/viewforum....
1  https://www.maxqda.de/support/forum/viewforum....
2  https://www.maxqda.de/support/forum/viewforum....
3  https://www.maxqda.de/support/forum/viewforum....
4  https://www.maxqda.de/support/forum/viewforum....

                                               html
0  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 S...
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 S...
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 S...
3  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 S...
4  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 S...
```

Extraktion der Postings

Nachdem die HTML-Daten für die Threads heruntergeladen sind und sich in der "df_threads"-Tabelle befinden, müssen aus ihnen die einzelnen "postings", d.h. die Fragen und Antworten der Nutzer, extrahiert werden und in eine neue Tabelle überführt werden, die eine Zeile für jedes Posting enthält.

Wir gehen dabei im Prinzip genauso vor, wie im letzten Schritt.

- Wir legen eine leere Zieltabelle "df_postings" an.
- Wir durchlaufen die 354 Zeilen von "df_treads" mit "iterrows".
- Für jede Zeile legen wir eine Zwischentabelle "df_tmp" an, in der wir die Informationen zu allen Postings dieses Threads einfügen.
- Bevor die nächste Threadzeile bearbeitet wird, wird die Zwischentabelle an "df_threads" angefügt.

Wenn alle Threads durchlaufen sind hat "df_postings" 1143 Zeilen.

Da hier allerdings keine Downloads erfolgen, sondern alles offline passiert, ist der Abschnitt in wenigen Sekunden bewältigt.

Von jedem Posting sammeln wir -- wieder mit Hilfe von CSS-Selektoren -- Den "header", das Datum, den Autor nzw. die Autorin, und den Textinhalt ein. Außerdem werden als Kontextinformation die Daten aus der Tabelle "df_threads" eingefügt, die sich auf den umgebenden Thread beziehen ("thread_url", "title", "post", "views"). Außerdem halten wir anhand des "class"-Attributs der Posting-Überschrift fest, ob es sich um das erste Posting im Thread handelt.

Vor der Extraktion der Postings ändern wir in der doppelten Schleife

"**for** cp in postings: ... **for** br in cp.cssselect: ... "

den HTML-Quelltext der Postings in einer Weise, dass nachher in der Contentspalte korrekte Zeilenumbrüche enthalten sind.

Das Ergebnis dieses Schrittes wird wieder am Bildschirm ausgedruckt.

In [16]:

```
df_postings = DataFrame()
for index, row in df_threads.iterrows():
    # print(row.thread_url)
    thread = htm.document_fromstring(row.html)
    postings = [e for e in thread.cssselect('div.post')]
    for cp in postings:
        for br in cp.cssselect('br'):
            br.tail = "\n" + br.tail if br.tail else "\n"
    df_tmp = DataFrame()
    df_tmp['header'] = [cp.cssselect('h2')[0].text_content().strip() for cp in postings
]
    df_tmp['datum'] = [cp.cssselect('p.author')[0].text_content().strip() for cp in pos
tings]
    df_tmp['autor'] = [cp.cssselect('dl.postprofile dt')[0].text_content().strip() for
cp in postings]
    df_tmp['content'] = [cp.cssselect('div.content')[0].text_content().strip() for cp i
n postings]
    df_tmp['first_in_thread'] = [(cp.cssselect('div.postbody h2')[0]).get('class') ==
'first' for cp in postings]
    df_tmp['base_page_url'] = row.base_page_url
    df_tmp['thread_url'] = row.thread_url
    df_tmp['title'] = row.title.strip()
    df_tmp['posts'] = int(row.posts)
    df_tmp['views'] = int(row.views)
    df_postings = df_postings.append(df_tmp)
print(df_postings.head())
```

```
header                datum  \
0  Spalten von Excel-tabelle werden nicht komplet...  29 Jun 2020, 20:25
1  Re: Spalten von Excel-tabelle werden nicht kom...  30 Jun 2020, 16:17
0  Codeliste wird nicht angezeigt im Creative Coding  28 Jul 2020, 11:44
1  Re: Codeliste wird nicht angezeigt im Creative...  29 Jul 2020, 11:52
2  Re: Codeliste wird nicht angezeigt im Creative...  29 Jul 2020, 12:38

        autor                                              content  \
0  Student726  Hallo, \n\nich habe eine dringende Frage zum D...
1    Maren_de  Hallo Student726,\n\nder häufigste Grund für e...
0  Vera TAube  Hallo zusammen,\n\nich habe folgendes Problem:...
1    Maren_de  Hallo Vera TAube, \n\nDanke für den Beitrag! N...
2  Vera TAube  Vielen Dank, das war tatsächlich das Problem!\...

   first_in_thread                                    base_page_url  \
0             True  https://www.maxqda.de/support/forum/viewforum....
1            False  https://www.maxqda.de/support/forum/viewforum....
0             True  https://www.maxqda.de/support/forum/viewforum....
1            False  https://www.maxqda.de/support/forum/viewforum....
2            False  https://www.maxqda.de/support/forum/viewforum....

                                        thread_url  \
0  https://www.maxqda.de/support/forum/viewtopic....
1  https://www.maxqda.de/support/forum/viewtopic....
0  https://www.maxqda.de/support/forum/viewtopic....
1  https://www.maxqda.de/support/forum/viewtopic....
2  https://www.maxqda.de/support/forum/viewtopic....

                                          title  posts  views
0  Spalten von Excel-tabelle werden nicht komplet...      1    276
1  Spalten von Excel-tabelle werden nicht komplet...      1    276
0  Codeliste wird nicht angezeigt im Creative Coding      3     35
1  Codeliste wird nicht angezeigt im Creative Coding      3     35
2  Codeliste wird nicht angezeigt im Creative Coding      3     35
```

In [ ]:

```
```

Nach Abschluss des letzten Schrittes haben wir alle wesentlichen Informationen, die wir brauchen, eingesammelt. Diese Daten müssen jetzt noch für MAXQDA gebrauchsfertig angepasst werden. Es ist sinnvoll, den Zwischenstand hier zu sichern, damit wir nicht jedes Mal, wenn wir bei der Nachbearbeitung einen Fehler machen und wieder den "Rohzustand" unserer Daten haben wollen, den zeitraubenden Download wiederholen müssen. Wir speichen sie in einer Excel-Tabelle, die wir im Bedarfsfall mit der auskommentierten Programmzeile wieder zurücklesen können.

In [18]:

```
df_postings.to_excel("postings_save.xlsx")

# df_postings = read_excel("postings_save.xlsx")
```

Datenaufbereitung

Aufbereitung für beide Projektvarianten

Bevor wir mit der Datenaufbereitung für MAXQDA beginnen können, müssen wir ein paar Vorkehrungen treffen, damit Python die in den Forumsdaten enthaltenen Datumsangaben korrekt lesen kann. Diese Vorkehrungen sind für Windows und MacOS unterschiedlich.

In [19]:

```
if platform.system() == "Windows":
    locale.setlocale(locale.LC_ALL, 'German_Germany.1252')
    df_postings.datum = df_postings.datum.str.replace('Mär', 'Mrz')
else:
    locale.setlocale(locale.LC_ALL, 'de_DE')
```

Zunächst nehmen wir eine Reihe von Transformationen an allen Postings vor,

- Eine der aus den Postings extrahierten Spalten heißt 'datum', sie hat aber den Datentype 'string' und ist somit keine echte Datumsvariable. Das ändern wir, indem wir sie mit **'to_datetime()'** in eine Datum/Zeit-Variable umwandeln. Anschließend können wir mit Hilfe der Funktion "strftime" (string from time) wieder eine Reihe neuer Textspalten generieren, die sich besser zur Auswertung in MAXQDA eignen als der ursprüngliche Zeitstempel (Jahr, Monat, Jahr_Monat, Tag, Wochentag, Stunde).
- Neben den Zeittransformationen berechnen wir noch die Länge des Posting-Textes.
- Aus dem Postingtext ziehen wir in den Fällen. wo dies angegeben ist, mithilfe von "regulären Ausdrücken" die Betriebssystem-Version der Poster*innen und die benutzte MAXQDA-Version heraus.

Anschließend gruppieren wir die Postings auf ihre Thread-URL und berechnen drei Spalten, die für alle Postinngs des Threads gleich sind:

- "thread_datum",
- "thread_zeit",
- "first_poster".

Mithilfe dieser Informationen können wir schließlich, nachdem wir die Gruppierung wieder aufgehoben haben, für die Einzelpostings einen Dokumentgruppennamen, einen Dokumentnamen und einen Posting-Typ ("Frage/Antwort") berechnen.

Das Ergebnis unserer Bemühungen speichern wir sicherheitshalber wieder in einer Excel-Datei.

In [20]:

```
df_postings['datum_zeit'] = to_datetime(df_postings.datum, format="%d %b %Y, %H:%M")
df_postings['post_datum'] =df_postings.datum_zeit.dt.strftime("%Y-%m-%d")
df_postings['uhrzeit'] = df_postings.datum_zeit.dt.strftime("%H:%M:%S")
df_postings['time_stamp'] = df_postings.datum_zeit.dt.strftime("%Y-%m-%d %H:%M:%S")
df_postings['jahr'] = df_postings.datum_zeit.dt.strftime("%Y")
df_postings['monat'] = df_postings.datum_zeit.dt.strftime("%m %B")
df_postings['jahr_monat'] = df_postings.jahr + "-" + df_postings.monat
df_postings['tag'] = df_postings.datum_zeit.dt.strftime("%d")
df_postings['wtag'] = df_postings.datum_zeit.dt.strftime("%u %A")
df_postings['stunde'] = df_postings.datum_zeit.dt.strftime("%H")
df_postings['size'] = df_postings.content.str.len()
df_postings['maxqda_ver'] = df_postings.content.str.extract(r"(?<=^Version: )(.*$)", re
.MULTILINE, expand=False)
df_postings['os_ver'] = df_postings.content.str.extract(r"(?<=^System: )(.*$)", re.MULT
ILINE, expand=False)

df_postings['thread_datum'] = df_postings.groupby('thread_url').post_datum.transform('f
irst')
df_postings['thread_zeit'] = df_postings.groupby('thread_url').datum_zeit.transform('fi
rst')
df_postings['first_poster'] = df_postings.groupby('thread_url').autor.transform('first'
)

df_postings['typ']=where(df_postings.autor == df_postings.first_poster, "Frage", "Antwo
rt")
df_postings['Dokumentgruppe']= (df_postings.thread_zeit.dt.strftime("%Y-%m\\%d") + "-"
+ df_postings.title).str[:63]
df_postings['Dokumentname']= df_postings.time_stamp + "_" + df_postings.autor

df_postings.to_excel("postings_modified.xlsx")
```

In [11]:

```
df_postings[['Dokumentgruppe', 'Dokumentname', 'typ', 'first_in_thread','os_ver', 'maxqda_ver']].head(10)
```

Out[11]:

	Dokumentgruppe	Dokumentname	typ	first_in_thread	os_ver	maxqda_ver
0	2020-06\29-Spalten von Excel-tabelle werden ni...	2020-06-29 20:25:00_Student726	Frage	True	Windows 10	MAXQDA 2020
1	2020-06\29-Spalten von Excel-tabelle werden ni...	2020-06-30 16:17:00_Maren_de	Antwort	False	NaN	NaN
0	2020-07\28-Codeliste wird nicht angezeigt im C...	2020-07-28 11:44:00_Vera TAube	Frage	True	Windows 10	MAXQDA 2020
1	2020-07\28-Codeliste wird nicht angezeigt im C...	2020-07-29 11:52:00_Maren_de	Antwort	False	NaN	NaN
2	2020-07\28-Codeliste wird nicht angezeigt im C...	2020-07-29 12:38:00_Vera TAube	Frage	False	NaN	NaN
3	2020-07\28-Codeliste wird nicht angezeigt im C...	2020-07-30 15:28:00_Maren_de	Antwort	False	NaN	NaN
0	2020-07\21-MAXQDA erkennt den Text in einigen ...	2020-07-21 16:29:00_alphabeta	Frage	True	Windows 10	MAXQDA 2020
1	2020-07\21-MAXQDA erkennt den Text in einigen ...	2020-07-29 11:45:00_Maren_de	Antwort	False	NaN	NaN
2	2020-07\21-MAXQDA erkennt den Text in einigen ...	2020-07-30 09:39:00_alphabeta	Frage	False	NaN	NaN
0	2020-07\24-Nicht codierte Passagen anzeigen	2020-07-24 12:33:00_stefan2607	Frage	True	Windows 10	MAXQDA 2020

Aufbereitung für das Projekt "Einzelposting in EXCEL-Datei"

Um unsere erste Importdatei für MAXQDA zu erzeugen, kopieren wir den "df_posting"-Dataframe.

In der Kopie sortieren wir die Spalten in eine sinnvolle Reihenfolge und speichern das Ganze als "postings_maxqda.xlsx" in Excel-Datei, die MAXQDA importieren kann.

In [21]:

```
df_postings_maxqda = df_postings.copy()
df_postings_maxqda = \
df_postings_maxqda.reindex(columns= \
                           ['Dokumentgruppe',
                            'Dokumentname',
                            'base_page_url',
                            'thread_datum', 'thread_zeit',
                            'posts','views','title',
                            'post_datum', 'time_stamp',
                            'tag', 'wtag', 'stunde', 'size',
                            'autor', 'header', 'content', 'typ',
                            'maxqda_ver', 'os_ver',
                            'first_in_thread', 'first_poster'
                           ]).copy()

df_postings_maxqda.to_excel("postings_maxqda.xlsx")
```

Aufbereitung für das Projekt "Threads mit Präprozessor"

Zum Export der zweiten Projetvariante, Threadtexte aus Textdatei mit mehrfach-vorcodierten Postings, sind einige weitere Datenmodifikationen notwendig.

Als erstes wollen wir verhindern, dass alle Einzelausprägungen der kategorialen Informationen -- wie Monate, Wochentage etc. -- als einzelne Codes nach MAXQDA übertragen werden. Stattdessen sollen sie über Obercodes zusammengefasst werden. Statt "Mittwoch" soll "wtag\3 Mittwoch", statt "Dezember" soll "monat\12 Dezember" in die Exportdatei geschrieben werden.

Die strukturierte Textdatei, die wir an MAXQDA übergeben wollen, soll auch keine nackte, unformatierte Textdatei sein, sondern eine Datei mit hervorgehobenen Überschriften. Dies können wir erreichen, wenn wir eine HTML-Datei exportieren. Dieses bedarf aber wieder einiger kleinerer Modifikationen (Missing Values durch 'n.a.' ersetzen, bestimmte Zeichen "escapen", "
"-Tag für Zeilenumbruch.

In [22]:

```
df_postings = df_postings.fillna('n.a.')

cat_var = ['thread_datum', 'post_datum', 'jahr', 'monat', 'jahr_monat', 'tag', 'wtag',
'stunde',
           'maxqda_ver', 'os_ver', 'typ']

for col in cat_var:
    df_postings.loc[:, col] = col + "\\" + df_postings.loc[:, col]

df_postings.content = df_postings.content.str.replace("\n", "<br>")
df_postings.content = escape(df_postings.content)
```

Anschließend kann die Ausgabe in eine HTML-Datei erfolgen, wozu die Postings auf den Dokumentgruppennamen gruppiert werden. Bei jeder Dokumentgruppe beginnt ein neuer Text (**#TEXT** wird in die Datei eingefügt).

Jedes Posting wird von den Markups **#CODE** und **#ENDCODE** umschlossenn. Vor jedes Posting wird eine ganze Batterie von Codes in die Datei geschrieben (separiert durch "**&&**").

Autor*innen-Namen und Posting-Datum werden durch **""**-Tags hervorgehoben.

In [23]:

```
postings_grouped = df_postings.groupby('Dokumentgruppe')

html_file = open("threads_maxqda.html", "w", encoding="UTF-8")

html_file.write("""<!DOCTYPE html>
<html lang="de">
<head>
<meta charset="UTF-8"/>
</head>
<body>
""")

for Dokumentgruppe, df_group in postings_grouped:
    html_file.write("#TEXT " + Dokumentgruppe + "<br>\n")
    for index, row in df_group.iterrows():
        html_file.write("#CODE ")
        html_file.write(
            row.thread_datum + "&&" +
            row.post_datum + "&&" +
            row.jahr + "&&" +
            row.monat + "&&" +
            row.jahr_monat + "&&" +
            row.tag + "&&" +
            row.wtag + "&&" +
            row.stunde + "&&" +
            row.maxqda_ver + "&&" +
            row.os_ver + "&&" +
            row.typ
        )
        html_file.write("<br>\n<strong>")
        html_file.write(row.time_stamp + "<br>\n")
        html_file.write(row.autor + "<br>\n")
        html_file.write("</strong><br>\n")
        html_file.write(row.content)
        html_file.write("<br>\n#ENDCODE")
        html_file.write("<br><br>\n\n")

html_file.write("""
</body>
</html>
""")

html_file.close()
```

Die Ergebnisdatei "threads_maxqda.html" kann jetzt als strukturierter Text in MAXQDA importiert werden.

Dieses Projektformat ist jedoch wenig geeignet um numerische Informationen zu verarbeiten. Deshalb haben wir die Anzahl der "views" und "posts" nicht in die strukturierte Textdatei übernommen. Ebenso wenig haben wir die Angabe über den Zeichenumfang der Postings genutzt. Wir können unserern Textdokumenten auf der Thread-Ebene -- ähnlich wie im vorherigen Projekt -- Eine Dokumentvariablentabelle zuordnen, die in der Lage ist, solche Variablen in sinnvoller Weise aufzunehmen.

- Zu diesem Zweck erzeugen wir eine auf Threadebene aggregierte Tabelle mit den relevanten Variablen.
- Anschließend benennen wir Spalte "Dokumengruppe" in "Dokumentname" um
- und berechnen eine neue Dokumentgruppenspalte. Diese Spalte füllen wir mit einem konstanten Inhalt ("Forumthreads").
- Am Ende schreiben wir die Tabelle in eine Exceldatei.

In unserer MAXQDA-Projektdatei müssen wir den automatisch erstellten Dokumentgruppennamen ändern, und zwar zu demselben Text, den wir in die Dokumentgruppe-Spalte geschrieben haben ("Forumthreads"). Der Name ist prinzipiell beleibig, er muss nur hier im Programm und im MAXQDA-Projekt identisch sein.

Die Exceldatei kann mit der Funktion "Dokumentvariablen importieren" dem MAXQDA-Projekt hinzugefügt werden.

In [24]:

```
df_thread_numeric_vars = (df_postings[["Dokumentgruppe","views","posts","size"]]
                     .groupby("Dokumentgruppe")
                     .agg({"views":"first","posts":"first","size":"sum"}))

df_thread_numeric_vars['Dokumentname'] = df_thread_numeric_vars.index
df_thread_numeric_vars["Dokumentgruppe"] = "Forumthreads"
df_thread_numeric_vars=df_thread_numeric_vars.reindex(columns=['Dokumentgruppe', 'Dokum
entname', 'posts', 'views', 'size'])

df_thread_numeric_vars.to_excel("threads_numeric_maxqda.xlsx", index=False)
```

In [16]:

```
df_thread_numeric_vars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 354 entries, 2011-12\09-an Anfang springen beim Textretrieval-Codie
ren verhi to 2020-07\28-Codeliste wird nicht angezeigt im Creative Coding
Data columns (total 5 columns):
Dokumentgruppe    354 non-null object
Dokumentname      354 non-null object
posts             354 non-null int64
views             354 non-null int64
size              354 non-null int64
dtypes: int64(3), object(2)
memory usage: 16.6+ KB
```