1. Entwurf

Als Grundlage wird die URL Liste der Personenseiten genommen, die im Moment im HTML-Format gespeichert sind und die letztendlich anhand eines Python-Skripts in .pickle Dateien umgewandelt werden.

Die Konvertierung erfolgt in mehreren Schritten. Inzwischen werden also noch andere Zwischenformate erzeugt, damit die Dateien von den unnötigen HTML-Tags gesäubert werden, indem man aber auf den inhaltlichen Teil achtet - das heißt, dass keine Informationen verloren gehen und dass jeder einzelne Textabschnitt immer noch der gleichen Kategorie zugeordnet wird.

Jede Kategorie wird mit einem Zeilenumbruch gefolgt, was später die Aufgabe des Parsers erleichtert. Die so vorbereitete Datei wird als .txt gespeichert und im nächsten Schritt in dem Parser weiterverwendet.

Die Aufgabe des Parsers besteht darin, entweder nach einer Zahl bzw. einem Symbol oder nach einem bestimmten Buchstaben, gefolgt von einem Zeilenumbruch zu suchen.

Da die Dateien nicht einheitlich formatiert sind, stößt man natürlich gleich auf unterschiedliche Anomalien, die zum Beispiel zu einer falschen Zuordnung der M-Buchstaben führen. Es muss eine Fallunterscheidung vorgenommen werden, weil sowohl die Kategorie von Mutter als auch die von Mitgliedschaft mit M bezeichnet wurde. Man kann also nicht gleich nach dem M suchen, die Zusatzbedingung überprüft noch, ob mindestens eine von 3 anderen Kategorien (WL, A, B) davor steht. Sollte das der Fall sein, handelt es sich um die Mitgliedschaft, sonst um die Mutter.

Einen solchen Sonderfall gibt es auch bei den Geburtsdaten, wo auch die Ortsangabe nicht einheitlich gehalten wurde. Es treten unterschiedliche Variationen der Angabe auf, entweder ist das Datum und der Ort durch ein Komma getrennt oder mit unterschiedlichsten Präpositionen bzw. noch weiteren Details.

Die durch den Parser konvertierten Daten werden in Dictionaries als .Pickle gespeichert, so dass deren Umwandlung in das TTL-Format leichter ist.

2. Dokumentation

Das HTML-Konvertierungstool besteht aus *drwHelper.py* und *openhtml.py*, die sich auf die Datei mit HTML-Personenseiten beziehen: *listOfAllHTMLSites.pickle*.

2.1 openhtml.py

Mithilfe der DrwHTMLParser Klasse können die Daten geparst werden. Die wichtigste Funktion saveName beinhaltet alle Bedingungen zu der richtigen Extraktion der Daten.

Alle Kategorien werden in Dictionary *namenDict* gespeichert.
ID und Namen sind einfach zu finden, es reicht aus, nur die entsprechende Zeilen zu durchlaufen und diese dann unter dem entsprechenden Dictionary-Eintrag zu speichern

```
ID = zeilen[0][:-1].strip()
```

Transliterierte Namen bzw. deren unterschiedliche Schreibweisen tauchen immer nach dem Label *Namensvariationen* auf, falls es solche gibt.

```
        for j in range(3,10):
            if ((zeilen[j][:5].strip() == "Namen") or (zeilen[j][:5].strip() ==
            "Name")):
                        if (zeilen[j][-1:] == "\n"):
                                namensVar += zeilen[j][19:-1].strip()
                        else:
                                namensVar += zeilen[j][19:].strip()
```

Die meisten Informationen bzw. Textabschnitte lassen sich auf ähnliche Weise zuordnen, weil die durch eine vorangehende Buchstabe und einen Zeilenumbruch gekennzeichnet sind, z.B. Dictionary-Eintrag von Vater:

```
for n in range(len(zeilen)):
        if (zeilen[n] == kat + "\n") and (ende == -1):
                start = n+1
elif ((zeilen[n][1:] == "\n") or (zeilen[n][2:] == "\n") or (zeilen[n][3:] ==
"\n") or (n+1 == len(zeilen)) ) and (n > start) and (start != -1):
                ende = n
                break
        if (start != -1) and (ende != -1):
                for o in range(start, ende):
                        ergebnis += zeilen[o]
                ergebnis = ergebnis[:-1]

        return ergebnis
```

Nicht eindeutige Benennung der Kategorien, wie bei M, die sowohl für die Kategorie Mutter als auch für die Mitgliedschaft steht, braucht komplexere Lösung. Einfacher Ansatz wäre, nur die Reihenfolge der Daten zu überprüfen, dass das erste M der Mutter zugeordnet sein soll und der Rest der Mitgliedschaft. Da aber manchmal die Reihenfolge nicht eingehalten ist, muss man sich in der if-Bedingung auf andere Kategorien beziehen. In diesem Fall führen wir die zusätzliche Variable wl_b_a ein, die auf einen Wert gesetzt wird, falls die Kategorien schon aufgetaucht sind.

```
for r in range(len(zeilen)):
                if ((zeilen[r] == "WL\n") or (zeilen[r] == "B\n")
                or (zeilen[r] == "A\n")):
                        wl_b_a = 1
                if (zeilen[r] == "M\n"):
                        if (wl_b_a != 1):
                                m_start = r+1
                if ((zeilen[r][1:] == "\n")
                or (zeilen[r][2:] == "\n")
                or (zeilen[r][3:] == "\n"))
                and (r > m_start) and (m_start != -1):
                        m_ende = r
                        break
```

Soll das nicht der Fall sein, bedeutet das – es ist die Mutter

2.2 drwHelper.py

In *drwHelper* gibt es Funktionen, die HTML-Daten ins andere Format umgewandelt mit Rücksicht auf deren Formatierung.

convertHTMLChars ist für die Konvertierung von String in utf-8 zuständig, *save* dagegen ermöglicht Speicherung der Daten unter einem angegebenen Pfad, einem Namen und mit der entsprechenden Endung.

3. Testsuite

- Überprüfen, ob alle HTML-Dateien konvertiert worden sind
- Richtigkeit der Zuordnung von Mutter und Mitgliedschaft für die ersten 10 Datensätze – einfacher Ansatz: falls der Text länger als 1 Zeile ist, sollte diese Information unter der Mitgliedschaft stehen
- ID und Vater – ist ID eine Nummer, ist Vater ein String