Sowohl HDP als auch LDA geben ein gemeinsames Datenformat für die zu verarbeitenden Dokumente vor. Auch wenn für die meisten Forscher es kein Problem darstellen sollte, die nativen Dokumente in das LDA-C genannte Format zu überführen, soll dennoch in einer Vorbetrachtung das Toolset vorgestellt werden, mit dessen Hilfe für diese Arbeit die Zeitungsartikel der ZEIT in LDA-C umgewandelt wurden. Es soll hierbei allerdings nur um eine kurze Einführung in den Gebrauch dieser in den Sprachen Perl und Python verfassten Skripte gehen, da diese nicht zu dem eigentlichen Umfang an Werkzeugen von TopModHis zählen. Damit aber die Werkzeuge von TopModHis ihren vollen Funktionsumfang bereitstellen können, ist es auf Informationen angewiesen, die besonders das Pythonskript data_prepare.py zur Verfügung stellt. Da dieses Skript wiederum auf Datenstrukturen des Skripts Listenerstellen.PL aufbaut, bietet es sich an, für eine bessere Nachvollziehbarkeit beide Skripte mit im Respository bereitzustellen. Es ist darauf hinzuweisen, dass beide Skripte für den Einsatz auf einem nativen Windows Betriebssystem erstellt wurden und besonders das Perlskript für einen Einsatz unter MacOS oder Linux umgeschrieben werden muss. Da diese Arbeit einer Gebrauchsanweisung nahe kommen soll, werden wie bereits bei der Analyse der Skripte zur besseren Übersicht Zwischenüberschriften eingefügt, um mit einem Blick gesuchte Eigenschaften sehen zu können.

Vorbereitung

Listenerstellen.PL

Das Perlskript Listenerstellen.PL hat die Funktion, alle in reiner Textform vorliegenden Artikel eines vom Nutzer im Skript vordefinierten Zeitraums (Konstanten my $Jahr:Start und my $Jahr_Ende) mithilfe des Treetaggers von Helmut Schmid[1] nach Wortarten aufzuschlüsseln und die Nomen in einer gesonderten Datei abzulegen. Da dieses Skript für die Auswertung eines Zeitungskorpus verfasst wurde, geht es davon aus, dass jedes Dokumenten einen Jahrgang darstellt, der in Ausgabe, Ressort und Artikelnummer aufgeteilt wurde. Das Programm durchläuft daher der Reihe und der Hierarchie nach in Schleifen diese Ebenen und müsste für ein gemischtes Korpus oder ein Korpus aus Büchern leicht umgeschrieben werden.

Aufruf

Zum Aufrufen des Skripts wechselt man mit einer Konsole in den Ordner, in dem das Skript liegt und startet dieses über „perl Listenerstellen.PL“. Das Skript sucht daraufhin nach Dateien, die dem jeweiligen Jahr entsprechen. Startet das Skript also mit dem Jahr 1969, so sucht das Skript nach der Textdatei „1969.txt“, die sich im Ordner des Skriptes befinden muss. Die Textdatei ist dabei nach folgendem Muster aufgebaut:

Beginn-1969-1-1-1
 # Text des Artikels
Ende-1969-1-1-1
Beginn-1969-1-1-2
 # Text des Artikels

[1] Schmid, Helmut, TreeTagger - a part-of-speech tagger for many languages, online verfügbar unter: http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/. Zuletzt geprüft am: 28.02.2019.

Ende-1969-1-1-2
Beginn-1969-1-2-1
…

Während die Befehlswörter *Beginn* und *Ende* wohl keiner weiteren Erläuterung bedürfen, sollte kurz auf die Logik der Zahlenkette eingegangen werden. Die Zahlenkette „1969-1-1-1“ steht für den ersten Artikel des ersten Ressorts der ersten Ausgabe des Jahres 1969. Dementsprechend würde die Zahlenkette „1989-50-13-4“ für den vierten Artikel des dreizehnten Ressorts der fünfzigsten Ausgabe des Jahres 1989 stehen. Diese Zahlenketten ermöglichen es dem Nutzer und z.B. auch den Topic-Modeling Anwendungen einen Rückschluss vom Wort auf den das Wort enthaltenden Artikel.

Neben der Textdatei des jeweiligen Skriptes müssen im Ordner mindestens zwei weitere Dateien liegen. In der Comma-Seperated-Value (kurz CSV) Datei „Jahr_Ausgaben.csv“ muss hinterlegt werden, wie viele Ausgaben das Skript für einen jeweiligen Jahrgang zu erwarten hat. Der Aufbau ist dabei simpel:

1969,52;
1970,52;
1971,45;
…

Pro Zeile wird die Jahreszahl vermerkt und per Komma die Zahl der Ausgaben angehängt. Anstelle der Ausgaben eines Jahres könnte es sich hierbei auch um die in der Textdatei hinterlegten Dokumente eines Jahres handeln.

Der Dateiname der weiteren CSV Datei setzt sich aus der Jahreszahl und dem Suffix „_ressort“ zusammen. Hier werden die Schlüssel und die Bezeichnungen der Ressorts geordnet nach dem Jahr und der Nummer der Ausgabe hinterlegt. Der Aufbau der Datei „1985_ressort.csv“ wäre also:

1985,1,1,Politik
1985,1,2,Wirtschaft
1985,1,3,Wissen
…
1985,2,1,Politik
1985,2,2,Wirtschaft
…
1985,3,1,Politik
1985,3,2,Wirtschaft
…

Ergebnisse

Listenerstellen.PL erstellt pro Jahrgang zwei unterschiedliche Dateien mit zwei verschiedenen Wortlisten, daher auch der Name des Skripts. Für das Jahr 1969 würden die Dateinamen „1969_analysiert.txt“ und „1969-Nomen.csv“ lauten.

…_analysiert.txt

Bei der Textdatei mit dem Suffix *_analysiert* handelt es sich um das Ergebnis des TreeTaggers, der in dem Skript per PIPE Befehl aufgerufen wird. Der Inhalt der Datei hat folgende Struktur:

Beginn-1969-1-1-1 ADJA Beginn-1969-1-1-1
Admiral NN Admiral
auf APPR auf
Springers NN Springer
Flaggschiff NN Flaggschiff
…

Zwei Eigenschaften der Ausgabe sind mit Blick auf das Ziel, ein Topic Modeling mit den Daten durchzuführen, besonders interessant. Zum einen behält der TreeTagger den Artikelbezeichner *Beginn-1969…* bei, zum anderen reduziert das Programm erkannte Worte auf seine Grundform.

…-Nomen.csv

Diese beiden Informationen werden in der CSV Datei mit dem Suffix Nomen zusammengeführt. Alphabetisch geordnet werden hier die Grundformen der erkannten Nomen abgelegt. Die Struktur der Datei gestaltet sich wie folgt:

…
ABM,1971,31,Gesellschaft,6
ABMS,1971,22,Gesellschaft,5
ABMs,1971,29,Politik,14
ABR,1971,33,Reisen,2
…
ABS,1971,2,Wirtschaft,15
ACI,1971,8,Reisen,1
…

Wie dem aufmerksamen Leser beim Anblick der Struktur aufgefallen ist, führt die Datei Artikel und Ressortnamen wieder zusammen, damit für weitere Analyseschritte nicht noch weitere Dateien mitgetragen werden müssen.

Data_prepare.py

Das nach dem Python 3 Standard verfasste Skript hat die Aufgabe, um Informationen angereicherte Wortlisten in das LDA-C Format zu überführen. Im Gegensatz zu Listenerstellen.PL erwartet es nicht, dass sich die Dateien im selben Ordner wie das Skript befinden. Stattdessen muss der Anwendung der Pfad zu jeder Datei mitgeteilt werden. Bei dem Inhalt der Datei darf es sich nicht um eine reine Textdatei handeln, sondern eine nach der Logik von „…-Nomen.csv“ strukturierten Datei der Art:

Wort,Metainformationen

Bei den Metainformationen sollte es sich um einen Schlüssel handeln, der es dem Skript erlaubt, die Wörter verschiedenen Dokumenten zuzuordnen. Eine Struktur nach Jahrgang, Ausgabe und Ressort

wird empfohlen, ist aber nicht notwendig. Da es sich bei dem Skript um eine Klassendefinition handelt, kann seine Funktionalität in anderen Skripten über die Import Funktion eingebunden werden:

```
from data_prepare import Meta_POS_to_LDA
```

Aufruf

Möchte man das Skript direkt verwenden und nicht in einem eigenen Programm verwenden, so muss man zunächst innerhalb einer Konsole zu dem Ordner navigieren, in dem sich das Skript befindet. Der Aufruf lautet dann:

```
python data_prepare.py "Pfad_zur_Datei1[,Pfad_zur_Datei2, …]"
```

Im Gegensatz zu Listenerstellen.PL sind keine weiteren Programme oder Dateien notwendig. Möchte man mehrere Dateien gleichzeitig umwandeln und dadurch in einer Datei bündeln, so erfolgt dies durch die Aneinanderkettung der Dateipfade, separiert mit einem Komma. Es empfiehlt sich, den Pfad in Anführungsstrichen zu setzen, da Zeichen wie das Leerzeichen zu einer falschen Verarbeitung der Informationen führen können.

Da es für einige Anwendungsfälle notwendig ist, die Schlüssel eines vorhandenen Vokabulars zu verwenden, ist es möglich, dem Programm den Pfad zu einer bereits existierenden Vokabeldatei zu übergeben. Die Pfadangabe kann dabei relativ oder absolut erfolgen, es ist allerdings immer, d.h. auch unter Windows, die Unix Notation zu verwenden, d.h. die einzelnen Glieder der Pfadangabe werden mit „/“ (Slash) und nicht mit „\“ (Backslash) voneinander getrennt. Der Aufruf gestaltet sich dann:

```
python data_prepare.py "Pfad_zur_Datei1[, …]" "Pfad_zur_Vokabeldatei"
[True/False]
```

Das Optionale True oder False teilt dem Skript mit, ob das Programm nach Wörtern suchen soll, die eventuell noch nicht in der Liste sind, um diese dann dem Vokabular hinzuzufügen (True). Die Option ist standardmäßig auf False gestellt, da der Rechenaufwand je nach Größe des Korpus groß ist.

Ergebnis

Das Skript erstellt drei Dateien aus der Wortliste. Die Datei „Wortliste_LDA.txt“ ist für die Verwendung in LDA oder HDP vorgesehen und enthält die von diesen Programmen geforderte Schreibweise. In der Datei „Dokumente_sortiert.txt“ findet man die sortierte Liste der Dokumentennamen bzw. die den Wörtern angehängten Metainformationen. Die Zeilennummer des Dokuments entspricht hierbei der späteren Dokumenten ID in den Ergebnissen von LDA und HDP. Die Datei „word_id.txt“ schlussendlich enthält den Wörterschlüssel des LDA Formats und das entsprechende Wort, stellt also das Vokabular dar. Zur besseren Arbeit mit den Tools von TopModHis wurden den Zeichenketten des Wortes der LDA Schlüssel mit angehängt, dies kann aber bei Bedarf ausgestellt werden, damit nur die reine Auflistung der Wörter ohne vorangestellten Schlüssel erfolgt.