Universität Leipzig

Abteilung für
Automatische Sprachverarbeitung

Anwendungen Linguistische Informatik

Abschlussbericht

Gruppe 8: Daten des Universitätsarchivs

Betreuer: Thomas Efer

Bearbeiter: Robert Noack
Stefan Schaub
Ramon Bernert

Inhaltsverzeichnis

1 Einleitung

In dieser Seminararbeit geht es um die Aufarbeitung von halbstrukturierten Daten. Bei den Daten handelt es sich um alte Personendaten aus dem Universitätsarchiv. Diese wurden zwar bereits durch die Verwendung von bestimmten Zeichenketten (beispielsweise .- für den Zeilenumbruch) strukturiert, diese Struktur wurde aber bei einem Bearbeitungsschritt leicht beschädigt und reicht für die Weiterverarbeitung der Daten nicht aus. Aus diesem Grund war es Ziel dieser Arbeit, die Daten aus dem einfachen .txt-Format in ein strukturiertes JSON-File zu überführen. In Kapitel 2 werden wir genauer auf die Daten und ihre interne Struktur eingehen und vorliegende Probleme aufzeigen. Im Anschluss wird in Kapitel 3 die Funktionalität unseres Codes erklärt und auf zusätzlich verwendete API's verwiesen. Zum Abschluss ist in Kapitel 4 ein Beispiel für unsere komplette JSON Struktur angegeben.

2 Datenherkunft und interne Struktur

2.1 Datenherkunft

Wie bereits erwähnt, handelt es sich bei den Daten um Personendaten des Universitätsarchives der Universität Leipzig. Die Personendaten stammen hauptsächlich aus dem 19. und 20. Jahrhundert. In dieser Zeit war es üblich, persönliche Dokumente die für ein Studium benötigt wurden bei der Universität einzureichen. Diese wurden dann so lange aufbewahrt, bis die Person ihren Abschluss erlangt hat und alle offenen Rechnungen bezahlt hatte. Neben den typischen Personendaten wie Name, Geburtstag, Geburtsort, etc. enthalten die uns vorliegenden Daten eine Liste mit den personenspezifischen Dokumenten. Diese wird derzeit aber nur als einfacher Fließtext dargestellt.

2.2 Interne Struktur

Die Grundstruktur der vorliegenden Daten ist relativ einfach gehalten. Sie besteht derzeit aus drei Bereichen die durch verschiedene Zeichenketten voneinander getrennt sind. Dabei handelt es sich um einen Namensbereich, persönliche Informationen und eine Liste der zuvor erwähnten Dokumente. Die am häufigsten auftretende Struktur ist die folgende:

- Namensbereich: persönliche Informationen-- Dokumente.-

Hierbei sieht man, dass die einzelnen Bereiche eindeutig durch die Zeichenketten ":", "--" und ".-" getrennt sind. Jedoch kann es vorkommen, dass bei

einzelnen Personen einer dieser Bereiche fehlt. Dadurch verändert sich auch die Struktur. Beispielsweise kann es dabei zu Strukturen wie

- Namensbereich: Dokumente.-

kommen. Unser Programm erkennt diese Probleme und konvertiert die Daten in die korrekte Form. Nicht vorhandene Bereiche werden somit in der späteren Struktur auch weggelassen. Ein Beispiel für eine komplette Struktur lässt sich auf Seite 7 finden. Zunächst werden wir aber auf die möglichen Inhalte der einzelnen Textbereiche eingehen.

2.2.1 Namensbereich

Der Namensbereich ist relativ simpel gehalten. Er hat fast immer die folgende Struktur:

- Nachname, Vorname_1 Vorname_2 ... Vorname_n:

2.2.2 Persönliche Informationen

Die persönlichen Informationen variieren da schon deutlich häufiger. Sie enthalten zunächst (falls vorhanden) den akademischen Grad der Person. Dieser ist durch das Kürzel "stud." einfach zu erkennen. Neben dem akademischen Grad werden auch Informationen zur Herkunft der Person angegeben. Dazu gehören das Geburtsdatum sowie der Geburtsort. Diese sind durch das Kürzel "geb." gekennzeichnet. Format und Reihenfolge dieser variiert aber stark. In manchen Fällen wird beispielsweise nur "aus Leipzig" angegeben. Folgende Kombinationen sind beispielsweise Möglich:

- stud. oecon., geb. Kiel 5.9.1897--
- geb. 5.7.1860 in Zöblitz--
- aus Saalfeld--

2.2.3 Dokumente

Die Dokumente sind meist nur eine einfache Auflistung mit der Struktur:

- Dokument_1, Dokument_2, ..., Dokument_n.-

Zu manchen dieser Dokumente wird außerdem das Ausstellungsjahr angegeben. Um jedoch eine bessere Analyse für die statistische Verteilung der Dokumententypen zu ermöglichen, weisen wir dem Dokument ein Tag mit

seinem Dokumententyp zu. So erhält das Dokument "Abschlusszeugnis" den zusätzlichen Tag "Zeugnis". Mögliche Tags wurden von uns manuell aus den vorliegenden Daten extrahiert. Dabei sind folgende mögliche Tags aufgetreten:

- Zeugnis
- Buch
- Schein
- Protokoll
- Genehmigung
- Zuweisung
- Diplom
- Quittung

3 Verwendete Methoden

3.1 Parsing der Daten

- Sequenzielle Abarbeitung des Dokuments

Die zunächst unformatierten Daten der Datei "personendaten2.txt", werden Zeilenweise extrahiert. Diese sind recht unhandlich und weisen Längen von bis zu 18760 Zeichen auf. Die Funktion "check_for_begin" sucht nach dem jeweils ersten Datensatz und setzt die Startposition an die gefundene Stelle. Danach folgt die statische Unterteilung, durch die Trennzeichen: ".-", in die jeweiligen Datensätze. Die Funktion "get_data_sets" ist dafür zuständig. Die Sets werden einzeln eingelesen. Zur besseren Lesbarkeit der endgültigen Daten werden alle Attribute, welche im folgenden gefunden werden, in ein geordnetes Dictionary ("OrderedDict()" aus der Klasse "collections") geschrieben.

Nun wird als erstes das Set in 2 listen durch den Begrenzer ':' geteilt. Dies geschieht in der Funktion "getting_name". Jener Teil, der vor dem Begrenzer steht, dient der Namenszuweisung. Weiter wird durch "get_surname" und "get_prename" der Nachname bzw. der Vorname ermittelt. Diese sind wieder durch ',' voneinander getrennt. Bsp: "Abb, Edmund: [...]". Zum Schluss werden die gefundenen Attribute in das Dictionary gespeichert.

Die Funktion "get_birth_place_and_academic_title" ermittelt den Geburtsort sowie die erlangten akademischen Titel. Jene befinden sich in der Sequenz, die mit ":" eingeleitet und "--" beendet wird. Anschließend wird

die Funktion "check_date" aufgerufen. Hier wird nach der Kombination von mindestens einer Ziffer, einem Punkt,mindestens einer Ziffer,einem Punkt gefolgt von mindestens zwei Ziffern gesucht. Dies ist als regulärer Ausdruck formuliert: "[0-9]+\.[0-9]+\.[0-9][0-9]+". Wird ein solches Datum gefunden, gehen wir davon aus, dass es sich um das Geburtsdatum der Person handelt. Der restliche String wird an die aufrufende Funktion zurückgegeben. Mit der Funktion "get_academic_title" wird nun nach einem potentiellen akademischen Titel gesucht. Das geschieht wieder mit einem regulärem Ausdruck der Form: "(stud\.)+|(dr\.)+". Wird ein solcher gefunden geht das Tool davon aus, dass die Spezifikation des akademischen Titels (z.B. Chirurg) mit einem "," beendet wird. Falls dies nicht er Fall ist wird nach dem Wort "aus" gesucht, welches als Einleitung für den Ort angenommen wird und stellvertretend den Informationsabschnitt beendet. Bsp: "[...]: stud. Philol., geb. 9.8.1857 in Bautzen--". Als nächstes werden Bindewörter wie "geb", "aus", "in" aus dem String entfernt. Dieser wird mit einer Liste der Leipziger Ortsteile verglichen ("liste_leipziger_vororte_clean"). Wird ein solcher erkannt nimmt das Programm im folgenden als Geburtsort Leipzig an. Dies soll mögliche Komplikationen beim Abgleich mit der Geodatenbank verhindern, da Leipziger Vororte zwar geschildert, andere Städte aber ohne weitere Unterteilung genannt werden. Falls es sich nicht um einen Vorort von Leipzig handelt, wird der String an die Methode "Location.getLocation" übergeben (siehe Ortsbestimmung). Liefert der Aufruf ein Ergebnis, wird dieses im Dictionary abgelegt.

Die folgende Funktion "get_certificate" sucht nach den Zertifikaten, welche mit "--" vom restlichen String getrennt sind. Wird das Programm fündig behandelt es jeden, durch ',' abgetrennten, Abschnitt wie folgt: Es wird nach dem Datum mit dem regulären Ausdruck: ".*(\d{4}([/]\d{2})*).*" gesucht. Danach wird die Stadt ermittelt, wo das Zertifikat erworben wurde. Hierzu wird das vorletzte Wort aus dem String genommen und überprüft, ob es korrekt ist. Folgende Syntax ist hier sehr häufig anzutreffen: [Typ] [Institution] [Ort] [Jahr] z.b. "[...] Abgangszeugnis Universitätsgericht Leipzig 1842". Somit erwies sich dieses Muster als sehr Treffsicher. Zum Schluss wird der (Zeugnis-) Typ ermittelt, indem einfach nach "zeugnis", "diplom", ... im String gesucht wird. Anschließend werden alle gefundenen Attribute in einer Liste gespeichert und jene in das Dictionary abgelegt.

Wurden alle Werte in allen Sets gelesen werden diese mittels der "json.dump" Methode in der "to_json"-Funktion in ein JSON-File geschrieben.

3.2 Ortsbestimmung

Die vorliegenden Personendaten enthalten zahlreiche Ortsangaben, z.B. den Geburtsort oder den Ausstellungsort der Dokumente. Um die Korrektheit dieser Ortsnamen zu überprüfen, verwendet die API den Webservice "geonames.org". Der Webservice kann nach einer Registrierung kostenlos verwendet werden. Als Eingabe erwartet dieser einen Orts- und den Nutzernamen. Die Ausgabe besteht aus einer Auflistung aller gefundenen Orte mit zahlreichen Eigenschaften, wie das Land, oder den Längen- und Breitengrad. Der Webservice wird über einen HTTP-GET-Request nach folgendem Schema angesprochen:

- http://api.geonames.org/searchJSON
  ?q={Ortsname}& username={Nutzer}

Die Ausgabe erfolgt im JSON-Format, wobei jeder Ort ein eigenes Objekt mit den dazugehörigen Eigenschaften darstellt. Die folgende Anfrage

- http://api.geonames.org/searchJSON
  ?q=leipzig& username=asv2015

liefert beispielsweise diese Ausgabe:

```
{
  "totalResultsCount":941,
  "geonames":[
  {
    "countryId":"2921044",
    "countryName":"Germany",
    "countryCode":"DE",
    "lng":"12.37129",
    "name":"Leipzig",
    "geonameId":2879139,
    "lat":"51.33962"
  },
  ...]
}
```

Dabei stellt jedes Element der Eigenschaft "geonames" einen zu der Suchanfrage passenden Ort dar. Da diese Auflistung nach der Relevanz sortiert ist, verwendet unsere API nur den ersten Eintrag und behandelt diesen als korrektes Ergebnis. Um den Webservice innerhalb der API einfach verwenden zu können, implementiert diese die Klasse "Location". Diese Klasse besitzt die

Methoden "getGeonameId", "getName", "getLng", "getLat" und "getUrl". Diese Methoden repräsentieren die entsprechenden Werte des Ergebnisses einer Suchanfrage. Dabei gibt "getUrl" die Google-maps-Url zu dem Ort zurück. Zusätzlich enthält die Klasse noch die statische Methode "getLocation", welche als Parameter den zu suchenden Ortsnamen erwartet und eine Instanz von "Location" zurück gibt. Die Methode erstellt anhand der Webservice-Url und dem übergebenen Ortsnamen einen HTTP-Request, welcher an den Webservice gesendet wird. Aus der daraus resultierenden Antwort wird ein JSON-Objekt erzeugt und die enthaltenden Werte auf die Eigenschaften einer neuen "Location"-Instanz übertragen. Anschließend wird diese Instanz zurück gegeben. Ein großer Nachteil des Webservices ist, dass dieser Zugriffsbeschränkt ist. Das heißt, innerhalb einer Stunde dürfen nur 2000 Anfragen getätigt werden und diese Anzahl reicht nicht aus um jeden Ortsnamen erneut bei dem Webservice abzurufen. Um diese Einschränkung zu umgehen, besitzt die "Location" ein statisches "Dictionary", welches zu jedem über die Methode "getLocation" angefragten Ort die dazugehörige "Location"-Instanz speichert. Somit schaut "getLocation" vor dem Absenden des Requests in diesem "Dictionary" ob der angefragte Ort schon vorhanden ist. Wenn das der Fall ist, wird diese gecachte Instanz zurück gegeben.

4 Beispielstruktur


```
{
  "surname": "Hahmann",
  "prename": "Friedrich August Adolf",
  "adcademic title": stud. Jur.,
  "birthplace": {
    "source": "Ilfeld",
    "name": "Ilfeld",
    "url": "https://maps.google.de/maps?q=51.5757,10.78469",
    "latitude": "51.5757",
    "longitude": "10.78469",
    "geonameId": 2896592
  },
  "additional_information_of_birthplace": "10.2.1848",
  "certificate": [
    {
      "name": "Reifezeugnis",
      "type": "Zeugnis",
      "year": "1868"
    },
    {
      "source": "Abgangszeugnis Universität Leipzig 1874",
      "name": "Abgangszeugnis Universität Leipzig",
      "location": {
        "name": "Leipzig",
        "latitude": "51.33962",
        "geonameId": 2879139,
        "longitude": "12.37129",
        "url": "https://maps.google.de/maps?q=51.33962,12.371
           29"
      },
      "type": "Zeugnis",
      "year": "1874"
    }
  ],
  "object_under_investigation": "Hahmann, Friedrich August
     Adolf: stud. Jur., geb. Ilfeld 10.2.1848-- Reifezeugnis
     1868, Abgangszeugnis Universität Leipzig 1874"
}
```