Fully automated online language resource detection

Yves Bugge
Marcel Jacob
J. Nathanael Philipp
Paul Röwer

1 Einleitung

Durch die weltweite Nutzung des Internet in allen Lebensbereichen von privaten Blogs, über Online-Shops bis hin zu einer Online-Präsenz von fast jedem Unternehmen, finden sich Texte in zahlreichen Sprachen. Das Finden von Texten in einer bestimmten Sprache kann sich als sehr schwierig erweisen. Vor allem wenn es sich um eine Sprache mit sehr wenigen Sprechern handelt.

Im folgenden beschreiben wir einen Ansatz um mit Hilfe der Google-Suche Webseiten einer Sprache zu finden. Die Idee ist es, eine Suchanfrage mit Wörtern der Sprache an Google zu senden, die man finden möchte. Als Ergebnis der Suchanfrage erhält man Webseiten, die die Wörter enthalten. Diese Vorgehensweise wird auch als BootCat bezeichnet.

Wenn man nach einer Sprache sucht, zu deren Wörtern man keine Übersetzung hat und man nachdem Zufallsprinzip Wörter auswählt, können Wortkombinationen zustande kommen, die es nicht gibt oder nur sehr selten sind. Diese Suchanfragen liefern entsprechend schlechte oder gar keine Ergebnisse.

2 Implementierung

Zum Auffinden der Webseiten benutzen wir Google, allerdings beschränkt Google die Seitenzugriffe. In Abhängigkeit von der benutzten API ist auch die Ergebnismenge beschränkt.

Es gibt eine JSON API, diese liefert pro Suchanfrage maximal 32 Ergebnisse, über die Websuche bekommt man 1000 Ergebnisse. Aus diesem Grund haben wir uns für die Websuche entschieden.

Ein weiteres Problem ist, das nach einer gewissen Menge an Suchanfragen Google, den Zugriff beschränkt, weswegen wir die Möglichkeit implementiert haben über Proxies die Anfragen zu stellen.

Unser Algorithmus erwartet als Eingabe eine Wortliste einer Sprache, diese wird als Grundlage für die Suchanfragen benutzt.

Wenn die Benutzung von Proxies ausgewählt wurde, wird als erstes wird eine Liste von Proxies angelegt, diese werden von `https://hidemyass.com/proxy-list/` geparsed.

Anschließend werden die Suchanfragen erstellt. Dazu werden zufällig Wörter aus der Wortliste ausgewählt. Dabei wird die angegebenen Einstellungen berücksichtigt. Danach werden die Ergebnisse von Google geparsed. Dabei wird solange ein Proxy benutzt, bis es zu einem Timeout kommt, danach wird der nächste Proxy benutzt. Wenn alle Proxy benutzt wurden wird eine neue Liste abgerufen.

Anschließend wird jede Ergebnisseite geparsed und auf die Sprache überprüft, dazu wird das Tool JLani von der ASV benutzt.

Unser Quellcode ist auf Github unter `https://github.com/jnphilipp/faolrd` zu finden.

3 Usage

Im `config`-Verzeichnis liegt eine Vorlage zur Konfiguration. Im einzelnen:

`wordlist.file`: Pfad zu Wortliste, relativ zum `data`-Verzeichnis

`wordlist.has_header`: gibt an, ob Wortliste eine Kopfzeile hat

`wordlist.language`: Sprache der Wortliste in ISO 639-3 Code

`query.average_lentgh`: Durchschnittliche Länge der Anfragen

`query.max_length`: Maximale Länge der Anfragen

`query.max_queries`: Maximale Anzahl an Anfragen

`faolrd.jlani.check_results`: gibt an, ob die Ergebnisseiten mit JLani getestet werden sollen

`faolrd.use_proxies`: gibt an, ob die Anfragen über Proxies gehen sollen

`faolrd.save_all_results`: gibt an, ob auch Ergebnisseiten die von JLani als aussortiert wurden, in die Ergebnisdatei gespeichert werden sollen

Das Programm kann entweder über Maven mit `mvn exec:java` oder über die jar-File gestartet werden. Der jar-File kann mit `mvn assembly:assembly` erstellt werden.

Für JLani wird ein Config-File im Verzeichnis `config/jlani` erwartet und die Wortlisten unter `resources/jlani`.

4 Ergebnisse

Die Ergebnisse werden automatisch in eine Datei im CSV-Format gespeichert, die im selben Verzeichnis wie die Wortliste liegt, es wird lediglich der Suffix `_result.csv` angehängt. Die erste Spalte bildet der Suchbegriff, die zweite der Titel der Ergebnisseite, anschließend die URL, gefolgt vom Vorschautext von Google. Die letzten beiden Spalten enthalten die von JLani bestimmte Sprache und den Konfidenzwert.

Wir haben unseren Ansatz an mehreren kleinen Sprachen getestet:

- Arifama-Miniafia (aai)
- Abau (aau)
- Abkhaz (abk)
- Abelam (abt)
- Achuar-Shiwiar (acu)

Als Quellen für die Wortlisten der Sprachen haben wir den Wortschatz-Korpus der ASV benutzt. Die Wortlisten sind dabei jeweils aus Texten der Bibel erstellt worden. Ziel wäre es nun Webseiten zu finden, die Texte in der entsprechende Sprache enthalten und dabei keine Bibeltexte sind.

So hat zum Beispiel für die Sprache Arifama-Miniafia die Anfrage `naya'iy hikartanen wawan` 11 mal zu der selben Ergebnisseite geführt, die das Neue Testament enthält.

Bei Abau haben wir die anschließende Filterung mit JLani nicht durchgeführt. Das führt dazu, das die Ergebnisdatei wesentlich länger ist. Allerdings sind viele der Seiten nicht in der entsprechenden Sprache. So ist zum Beispiel das Wort `korey` ein sehr beliebter Name.

Der Testlauf für Abkhaz führte zu 184 Ergebnissen, allerdings sind es nur zwei verschiedene URLs und dies bei 20 verschiedenen Suchanfragen.

Die Ergebnisse für Abelam haben nur verschiedene Seiten mit Bibeltexten geliefert.

Der Testlauf für Achuar-Shiwiar ergab keinerlei Ergebnisse, es wurden 100 Suchanfragen mit einer maximal Länge von fünf und einer Durchschnittslänge von 1 gestellt.

5 Zusammenfassung

Was die Ergebnisse soweit angeht kann man sagen, das man zwar Texte für die Sprachen findet, diese aber meist auch nur Bibeltexte sind.

Für eine bessere und allgemeinere Beurteilung des Ganzen, müsste man jedoch das Programm für wesentlich mehr Sprachen und mit einer größeren Menge an Anfragen, mindestens dem drei- oder vierfachem der Anzahl der Wörter in der Wortliste laufen lassen. Dies sprengt allerdings den Rahmen dieser Arbeit.