Indeks słów przy pomocy NLP Toolkit

Wstęp

Celem niniejszej pracy jest porównanie trzech zestawów narzędzi służących do przetwarzania języka naturalnego. Na początku zostaną przedstawione i krótko omówione narzędzia wybrane do testu. Następnie znajdzie się opis instalacji każdego z poszczególnych zestawów oraz proponowane rozwiązanie problemu.

Przy pomocy każdego zestawu zostanie rozwiązany problem stworzenia indeksu słów danego tekstu. Na podstawie pliku wejściowego należy przetworzyć każde słowo, sprowadzić je do formy podstawowej i nadać jemu odpowiedni indeks. Należy uwzględnić możliwość występowania powtórzeń, wówczas każde kolejne wystąpienie powinno uzyskać indeks nadany pierwszemu wystąpieniu.

Narzędzia

Do testu zostały wybrane trzy zestawy narzędzi: PSI-Toolkit, Natural Language Toolkit oraz Stanford NLP Toolbox. Wybierając narzędzia starałem się je dobrać w taki sposób aby nie były do siebie zbyt podobne i aby każde z nich miało swoje unikalne charakterystyki.

PSI-Toolkit - jest to narzędzie stworzone na Uniwersytecie Adama Mickiewicza w Poznaniu i zostało wydane na licencji LGPL. Projekt można pobrać z repozytorium na Github pod adresem https://github.com/filipg/psi-toolkit. Z oprogramowania można korzystać na dwa sposoby, pierwszym jest użycie interfejsu webowego, drugim uruchomienie PSI-Toolkit z poziomu konsoli linuxowej. Celem projektu było umożliwienie korzystania z tego typu narzędzi osobom bez szerokiej wiedzy informatycznej.

Natural Language Toolkit - jest to zestaw bibliotek przeznaczony do języka Python. Umożliwiają przetwarzanie tekstu w programach stworzonych w języku Python. Zestaw został również wydany jako projekt o otwartym kodzie źródłowym a obecnie rozwijany jest przez jego społeczność.

Stanford NLP Toolbox - projekt stworzony na uniwersytecie Stanforda, zrealizowany w języku Java, wydany na licencji GPL. Może zostać wykorzystywany do projektów niekomercyjnych. Narzędzia są dostarczone w postaci paczek jar i mogą zostać wykorzystywane jako biblioteki w innych programach lub mogą być uruchomione jako samodzielna aplikacja z poziomu linii poleceń.

Instalacja

PSI-Toolkit

Instalacja polega na pobraniu wcześniej przygotowanej paczki instalacyjnej ze strony http://psi-toolkit.amu.edu.pl/download.html. Przykładowo instalacji paczki w systemie Ubuntu polega na wydaniu jednego polecenia w terminalu:

```
$ sudo dpkg -i psi-toolkit_0.3.0-1_amd64.deb
```

Następnie możemy uruchomić serwer aplikacji

```
$ psi-server
```

lub uruchomić aplikację bezpośrednio w terminalu

```
$ psi-pipe
```

Natural Language Toolkit

Z faktu, że NLTK jest przeznaczona dla środowiska Python do jego instalacji wykorzystamy setuptools.

1) Instalujemy PIP czyli Python Package Index

```
sudo easy_install pip
```

2) Instalujemy NLTK wraz z wymaganą zależnością

```
sudo pip install -U pyyaml nltk
```

Stanford NLP Toolbox

Tak jak było to podkreślone wczęsniej Stanford udostępnia swoje narzędzie jako paczka jar, więc w takim wypadku wystarczy pobrać odpowiednie archiwum ze strony http://nlp.stanford.edu/downloads/corenlp.shtml. Biblioteka została również dodana do głównego serwera Maven, więc istnieje możliwość dodania Stanford NLP jako zależność w naszym projekcie.

Rozwiązanie problemu

Z uwagi na to, że jedynie PSI-Toolkit wspiera język polski, przygotowany plik wejściowy zawiera tekst w języku angielskim. Tak jak zostało to opisane wcześniej problem przed jakimi stoją zestawy polega na stworzeniu indeksu słów. Ogólny schemat działania przedstawia się następująco:

- wczytaj dane wejściowe
- poddaj tekst tokenizacji i lematyzacji
- utwórz listę zawierającą formy podstawowe
- dla każdej formy podstawowej przypisz indeks
- wyświetl rezultat

PSI-Toolkit

Zaczniemy od sprowadzenia wejścia do listy wyrazów w formie podstawowej. W tym celu wykonujemy polecenie:

```
cat input.txt | psi-pipe lemmatize --lang en ! write-simple --tags lemma >
output.txt
```

Przekazujemy zawartość pliku input.txt potokiem do programu psi-pipe, który najpierw poddaje wyrazy lematyzacji a następnie wyświetla rezultat działania, który zostaje prekazany do pliku output.txt.

output.txt zawiera dane w następującej postaci

```
#head output.txt
1 stater
2 Paul
3 from
4 contend|contending
5 for
6 a
7 berth
8 on
9 the
10 U.S.
```

Jak widzimy w linii 4 zostały zwrócone dwie formy podstawowe. Postanowiłem, że w przypadku takiej sytuacji zawsze będę wybierał pierwszą formę czyli w tym przypadku contend.
Następnie stworzyłem prosty skrypt w bashu, którego zadaniem jest wczytanie listy zwróconej przez PSI-Toolkit, przypisanie każdemu wyrazowi indeksu a następnie wyświetlenie rezultatu.

Wykonanie skryptu wygląda następujące

```
$ ./word_index output.txt
```

Skrypt

```
 1 #!/bin/bash
 2
 3 ARRAY=()
 4
 5 function get_index {
 6     for ((i = 0; i < ${#ARRAY[@]}; i++)); do
 7         if [ "${ARRAY[$i]}" = $1 ]; then
 8             index=$(($i+1))
 9             echo "${index} - ${ARRAY[$i]}"
10             break
11         fi
12     done
13 }
14
15 while read line; do
16     ARRAY+=("$(echo $line | awk -F"|" '{print $1}')")
17 done < $1
```

```
18
19 for ix in ${!ARRAY[*]}
20 do
21     get_index "${ARRAY[$ix]}"
22 done
```

Natural Language Toolkit

W przypadku biblioteki NLTK cały proces tworzenia indeksu słów odbywa się przy użyciu jednego skryptu. Najpierw następuje tokenizacja każdej linii wczytanego tekstu, jest ona przeprowadzona przy pomocy metody word_tokenize. Następenie każdy otrzymany wyraz zostaje sprowadzony do formy podstawowej poprzez wykorzystanie lematyzatora WordNetLemmatizer, jednego z dostępnych w bibliotece NLTK.

```
1  #!/usr/bin/env python
2  #-*- coding: utf-8 -*-

3  import nltk
4  from nltk.stem.wordnet import WordNetLemmatizer

5  file_name = 'input.txt'

6  lemmatizer = WordNetLemmatizer()
7  indexes = {}
8  index = 1

9  f = open(file_name, 'r')

10 for line in f.readlines():
11   tokens = nltk.word_tokenize(line)
12    for token in tokens:
13       lemma = lemmatizer.lemmatize(token)
14        if lemma not in indexes:
15            indexes[lemma] = index
16             index += 1

17        print str(indexes[lemma]) + " - " + token
```

Stanford NLP Toolbox

Ostatnim testowanym narzędziem jest Stanford NLP toolbox. Rozwiązanie zostalo przygotowane w języku Java. Zaczynamy od stworzenia obiektu StanfordCoreNLP i ustawienia jakim operacjom powinien zostać poddany tekst.

```
Properties props = new Properties();
```

```
props.put("annotators", "tokenize, ssplit, pos, lemma");
this.pipeline = new StanfordCoreNLP(props);
```

Tak jak poprzednio poddajemy tekst tokenizacji i lematyzacji, dodatkowo, musimy dodać annonatory ssplit i pos, które są zależnościami lemma.

Następnie przechodzimy do tworzenia listy wyrazów w formie podstawowej

```
        Annotation document = new Annotation(documentText);
        // i uruchamiamy na nim po kolei wszystkie annonatory
        this.pipeline.annotate(document);
        // Iterujemy po każdym zdaniu
        List<CoreMap> sentences = document.get(SentencesAnnotation.class);
        for (CoreMap sentence : sentences) {
            // Poddajemy każde zdanie tokenizacji
            for (CoreLabel token : sentence.get(TokensAnnotation.class)) {
                // i dla każdego tokenu tworzymy jego formę podstawową
                lemmas.add(token.get(LemmaAnnotation.class));
            }
        }
```

Na koniec przekazujemy na wyjście programu stworzony indeks słów. Wypisywanie samego indeksu odbywa się w zbliżony sposób do wersji programu używającej NLTK, ze względu na obszerność kodu ten fragment został tutaj pominięty.

Rezultat

Każde z przedstawionych rozwiązań zwróciło podobny do rezultat, które prezentował się następująco:

# head -25 1 - stater 2 - Paul 3 - from 4 - contend 5 - for	6 - a 7 - berth 8 - on 9 - the 10 - U.S.	11 - cup 12 - team 13 - after 14 - win 15 - first	16 - tour 17 - event 18 - last 19 - year 20 - stay	21 - within 22 - three 23 - stroke 24 - of 9 - the

Różnice wynikały z zastosowania różnych bibliotek do tokenizacji i lematyzacji. Przykładowo wyrażenie “It’s” raz było rozdzielane na “It” i “‘s” a w drugim przypadku tokenizator zwracał “It’s”.

Warto podkreślić, że jedynie PSI-Toolkit oferował wsparcie języka polskiego. Dodatkowo przeprowadzanie lematyzacji było bardo prostym procesem. Jednak już samo wygenerowanie indeksu wymagało dodatkowego skryptu Bash. Kolejną zaletą tego narzędzia jest dobra dokumentacja wraz z przykładami oraz przeglądarkowa wersja zestawu.

Rozwiązanie towarzyszące bibliotece NLTK, cechowało się dużą prostotą i zwięzłością. Wymagane było napisanie skryptu w języku Python, jednak sam skrypt nie jest skomplikowany i nawet mniej doświadczona osoba powinna sobie z nim poradzić. Biblioteka świetnie sobie radzi z językiem angielskim, brakuje wsparcia dla polskiego.

Rozwiązanie oparte o Stanford NLP Toolbox jest najbardziej skomplikowane ze wszystkich. Wymagany program znacznie przewyższa objętościowo wersję z NLTK. Z kolei użycie narzędzia z poziomu konsoli jest bardzo nieintuicyjne i trudne w stosowaniu.

Podsumowując porównanie w momencie gdy chcemy pracować z NLP bez konieczności tworzenia własnych programów powinniśmy wybrać PSI-Toolkit, które świetnie nada się do prostych prac. W przypadku chęci stworzenia własnych skryptów przetwarzających tekst warto skupić się na NLTK, która oferuje dobrze przygotowaną bibliotekę o sporych możliwościach.