# PyNER - Dokumentacja

Konrad Delong Antek Piechnik

17 lutego 2009

## Spis treści

# 1 Wstęp

Projekt został zrealizowany w ramach przedmiotu Inżynieria Oprogramowania. Głównym zadaniem projektu będzie utworzenie systemu typu Named Entity Recognition - wyszukiwanie konkretnego typu słów w tekście stron WWW (takich jak nazwiska, imiona, czy pseudonimy).
Projekt domyślnie ma być integralną częścią systemu SWAT. Sercem systemu jest algorytm łączący zestawy reguł językowych do utworzenia optymalnych kombinacji rozpoznających encje danego typu w tekście. Dzięki takiemu podejściu nie tylko wykorzystuje podane reguły do stworzenia mozliwie najlepszej kombinacji, ale również pozwoli na łatwe dodanie nowych reguł do aplikacji.

# 2 Struktura systemu

System składa się z następujących części:

## 2.1 Tagger tekstów do testów

Skrypt mający na celu skanowanie przykładowych tekstów a następnie generowania specyficznych raportów odnośnie ilości wystąpień poszczególnych encji w tekście.
Dzięki skryptowi będzie możliwe przygotowanie tekstów odpornych na konkretne reguły, oraz zwiększenie możliwości testowania aplikacji.
Tagger bazować będzie na liście danych encji (np. bazie nazwisk), z której nie będziemy korzystać w pracy Rdzenia aplikacji.

## 2.2 Generator reguł n-gramowych

W związku z chęcią implementacji reguł bazujących na najpopularniejszych n-gramach utworzony został generator reguł który zbiera najczęściej występujące n 3,4,5 gramy występujące w tekstach przed nazwiskami oraz po, które posłużą za wykładnię do łączenia reguł.
Do tworzenia list n-gramów potrzebny jest odpowiedni korpus zawierający nazwiska, jak i również ich baza, dzięki której można je wykrywać w tekście.
Generator, w celu uniknięcia zbierania nic nie znaczących n-gramów porównywał będzie nie tylko jego zawartość ale również pozycję w słowie.

### 2.3 Rdzeń

W tej części aplikacji znajduje się zaimplementowany algorytm który aplikuje przygotowane i wygenerowane reguły do wyszukiwania nazwisk w tekście.
Przygotowanych została konkretna liczba reguł, oraz ich przykładowe kombinacje dzięki którym będzie można porównywać ich efektywność na tekstach przykładowych.
Rdzeń pobiera również odpowiednie listy n-gramów o różnej długości, wygenerowane uprzednio przez Generatora reguł n-gramowych.
Rdzeń został napisany w języku Python, dzięki czemu wyjątkowo proste jest ewentualne dodanie dodatkowych reguł.

## 3 Opis działania aplikacji

### 3.1 Zastosowane reguły (przykłady)

### 3.2 Reguły kontekstowe

- Słowo ‘doktor‘ poprzedza nazwisko
- Słowo ‘mgr‘ poprzedza nazwisko
- Słowo ‘mgr inż.‘ poprzedza nazwisko
- Reszta tytułów naukowych.
- Słowo ‘lek.‘
- Słowo ‘mec.‘
- Słowo ‘arch.‘
- Reszta tytułów zawodowych.
- Słowo ‘pan‘ poprzedza nazwisko
- Słowo ‘pani‘ poprzedza nazwisko
- Czasownik w trzeciej osobie liczby pojedynczej następuje po nazwisku (czasownik w formie męskiej czy tez damskiej)
- Imiesłowy takie jak ‘zamieszkały‘, ’urodzony‘.

### 3.3 Reguły bazujące na innych encjach

- Imię poprzedza nazwisko
- Przezwisko znajduje się na 2 gim miejscu w wyrażeniu trójsłownym (np. Antoni ‘Tosiek‘ Piechnik)
- Przezwisko znajduje się na 3 cim miejscu w wyrażeniu trójsłownym (np. Antoni Piechnik ‘Tosiek‘)

### 3.4 Reguły bazujące na regułach ortograficznych

- Nazwisko zaczyna się dużą literą
- Nazwisko jest częścią dwu lub trzy wyrazowego wyrażenia w których wszystkie elementy są pisane wielką literą.

### 3.5 Reguły n-gramowe

- Nazwisko kończy się charakterystycznym sufiksem (np. ‘’-ski‘’, ‘’-icz‘’)
- Pozostałe reguły generowane poprzez system

Dzięki zastosowaniu systemu reguł oraz ich kombinacji, bez problemu będzie można do systemu wstawiać dodatkowe reguły co usprawni jego prace oraz uświetni wyniki osiągane przez algorytm.
Wielką zaletą zastosowanego algorytmu jest fakt, iż reguły tak naprawdę nie muszą dokładnie precyzować wystąpień danych encji (tu nazwisk), ale jedynie sprzyjające temu warunki (które w połączeniu z innymi warunkami mogą definiować reguły).

## 4 Zastosowane technologie

Do implementacji znacznej części aplikacji wykorzystano język Python, ze względu na jego perfekcyjne przystosowanie do prac nad przetwarzaniem języka naturalnego. W związku z faktem, iż projekt SWAT jest napisany w języku Java, należało wykorzystać swojego rodzaju pomost pomiędzy naszą częścią aplikacji a dotychczasowymi interfejsami wyszukiwania encji w tekstach. W tym celu wykorzystaliśmy Jythona, czyli implementację języka Python napisaną w języku Java.
Poza Jythonem korzystaliśmy z funkcji wbudowanych w język Python, związanych

z przetwarzaniem tekstu oraz konwersją między różnego rodzaju kodowaniem (pozwalająca na komunikację oraz transfer danych z poziomu Javy do Pythona i na odwrót).

# 5 Opis zastosowanych narzędzi

## 5.1 Jython

Implementacja języka Python w Javie pozwalający na transparentna komunikacje miedzy Pythonem a klasami Javy. Okazał się niezastąpiony przy wiązaniu aplikacji Rdzenia z dotychczasowymi interfejsami projektu SWAT.
Dzięki wykorzystaniu zewnętrznych bibliotek związanych m.in. z kodowaniem udało się bez najmniejszych problemów wywoływać klasy napisane w Rdzeniu (w Pythonie) z poziomu Javy, jak również importować wszelakie pakiety projektu SWAT do kodu aplikacji w Pythonie.
Wiecej informacji na stronie Jythona: www.jython.org

## 5.2 encodings

Moduł Pythona zawierający zbiór najważniejszych i najpopularniejszych kodowań oraz metod z nimi związanych.
Dzięki niemu udało się rozwiązać problem związany z komunikacją (w szczególności przesyłaniem polskich znaków z obiektów Javy do obiektów Pythona)

## 5.3 egothor

Silnik full-text search z którego korzystaliśmy przy tworzeniu systemu.
Wiecej informacji na stronie: http://www.egothor.org/

## 5.4 Morfologik

Analizator morfologiczny, słownik morfologiczny, korektor gramatyczny.
Wiecej informacji na stronie: http://morfologik.blogspot.com/

# 6 Opis powiazania aplikacji z Interfejsem IQuantumDetector z projektu SWAT

Należy zaimportować odpowiednie pakiety do kodu w Javie a następnie wykorzystać metodę getJythonObject klasy JythonFactory. Informacje odnośnie składni znajdują się na stronie projektu Jython.
Z kolei w kodzie Pythona importujemy wszystkie klasy Javy które będą nam potrzebne (zarówno te z projektu SWAT), a następnie korzystamy z udostępnianych przez nie metod, jak gdyby były klasami Pythona.

## 6.1 Kod od strony aplikacji w Javie

```
import org.ppbw.agh.swat.hoover.smith.quantum.detection.IQuantumDetector;
import jyinterface.factory.JythonFactory;
..

public class Main {
  public static void main(String[] args) throws Exception {
    String shortName = "org.ppbw.agh.swat.hoover.smith.quantum.detection.IQuantumDete
    Object obj = JythonFactory.getJythonObject(shortName, "test.py", "InternetDetecto
    IQuantumDetector detector = (IQuantumDetector) obj;
    ..
    for (IContentSegment s : resourceModel.getLeafSegments()) {
      System.out.println(detector.detectQuantums(s).toString());
    }
  }
}
```

## 6.2 Kod przykładowy implementujący test z przykładowego pliku w Pythonie

```
from java.io import ByteArrayInputStream
from java.util import ArrayList
..
from org.ppbw.agh.swat.hoover.smith.quantum import QuantumType

#from unittest import main, TestCase
```

```
class InternetDetector(IQuantumDetector):
  ..
  def detectQuantums(self, leafSegment):
    dq = []
    for word_id in range(leafSegment.getWordsCount()):
      if leafSegment.compareWord(word_id, "Internet"):
        dq.append(DetectedQuantum(leafSegment.getWordToken(word_id), QuantumType.NICK
        print leafSegment.getWordToken(word_id).tokenContent
      return ArrayList(dq)
```

# 7 Opis struktury folderów/plików w systemie

- io-projekt - zawiera dane tymczasowe etc.
  - build.xml - plik build.xml dla ANTa
  - data - folder zawierający dane na których pracuje system.
  - doc - folder zawierający tę dokumentację w formacie TeX
  - java-lib - folder zawierający wykorzystywane biblioteki Javy
  - java-src - kody źródłowe Javy
  - pyner - źródła systemu (Rdzenia, Taggera oraz Generatora)
  - python-lib - biblioteki wykorzystywane przez źródła Pythona

# 8 Misc

## 8.1 Repozytorium

Repozytorium publiczne projektu znajduje się na serwerze GitHub pod adresem http://github.com/tph/io-projekt/tree

## 8.2 Kontakt

Kontakt z autorami systemu:

- Konrad Delong - konryd ¡at¿ gmail.com

- Antoni Piechnik - antek.piechnik ¡at¿ gmail.com