Grupa projektowa: **WT 9:15** Wrocław, 22 maja 2016 r.

Piotr Chmiel, 200608
Kamil Machnicki, 200752
Łukasz Matysiak, 200646
Michał Polański, 200852
Maciej Stelmaszuk, 200654
Jakub Zgraja, 200609

Deeplearning – tagger
(POS, lematyzacja)

DOKUMENTACJA PROJEKTOWA

Kurs „Zastosowania informatyki w gospodarce”
Rok akad. 2015/2016, kierunek INF, studia II stopnia

PROWADZĄCY:
dr inż. Tomasz Walkowiak

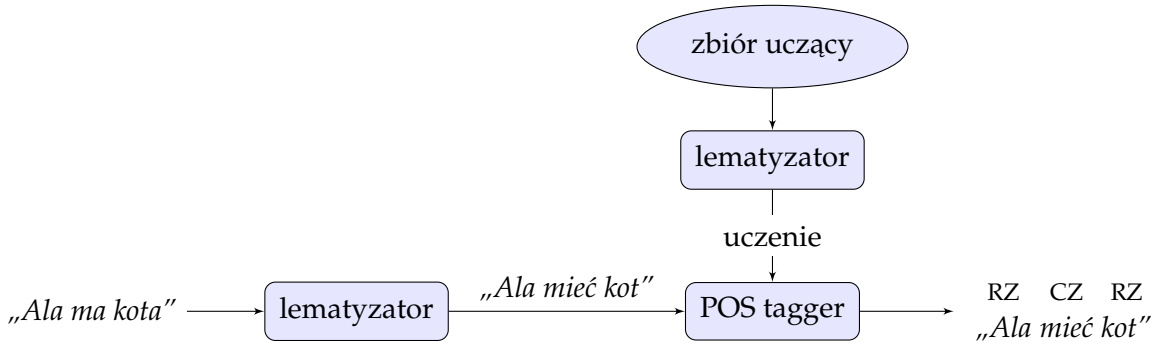
WYDZIAŁ ELEKTRONIKI
POLITECHNIKA WROCŁAWSKA

Spis treści

1 Cel projektu

Temat projektu: *Deeplearning – tagger (POS, lematyzacja).*

Zadaniem realizowanego przedsięwzięcia jest stworzenie programu głębokiego uczenia (ang. *deep learning*) z zakresu przetwarzania języka naturalnego. Jego celem jest przypisywanie każdemu wyrazowi w tekście wejściowym odpowiadającej mu części mowy (ang. *part-of-speech*, *POS*). Wykonanie zadania tagowania zostanie poprzedzone procesem lematyzacji tekstu.



Rysunek 1: Uproszczony schemat działania programu.

Realizacja powinna posiadać formę aplikacji desktopowej lub skryptu. Cel projektu zostanie osiągnięty, jeśli przygotowane oprogramowanie (wyposażone w odpowiedni zbiór uczący) będzie w stanie przetwarzać tekst w języku polskim w czasie i dokładności, które zostaną sprecyzowane przez prowadzącego.

2 Koszty

Szacunkowy czas wykonania

Biorąc pod uwagę zakres tematyczny projektu szacuję się, że do jego wykonania potrzebnych jest 400 godzin roboczych. Około 1/8 czasu poświęcone będzie na zebranie i analizę materiałów dotyczących projektu, ogólne poznanie możliwości i technologii. Największa część czasu zużyta zostanie na implementację projektu (stworzenie lematyzatora oraz taggera). Szacuje się, że będzie to 6/8 czasu. Pozostała część przeznaczona jest na kontakty z prowadzącym oraz testy zaimplementowanego rozwiązania.

Szacunkowy koszt projektu

Mając na uwadze poziom skomplikowania zadania projektowego stawka za godzinę pracy wynosi 30 zł. Przy szacowaniu kosztów projektu nie uwzględniamy dodatkowych wydatków, które trzeba będzie ponieść w przypadku, gdy np. zajdzie potrzeba wykupienia domeny WWW. Nie są uwzględnione także koszty wdrożenia projektu w środowisku produkcyjnym. W związku z powyższym szacunkowy koszt wykonania projektu jest równy 12 000 zł netto.

Usługi, o których mowa w poprzednich punktach opodatkowane są 23% stawką podatku VAT (podatek od towarów i usług). Szacowana cena brutto wynosi **14 760 zł**.

Tabela 1: Kosztorys.

Nazwa	Jedn.	Ilość	Cena jedn.	Wart. netto	Stawka	Podatek	Wart. brutto
Robocizna	r-g	400	30	12 000	23 %	2 760 zł	14 760 zł
						Razem:	**14 760 zł**

3 Terminy i harmonogram projektu

Granicznym terminem realizacji projektu jest **7 czerwca 2016 r**. W tym dniu nastąpi prezentacja rezultatów projektu wraz z przekazaniem jego pełnej dokumentacji.

Tabela 2: Harmonogram projektu.

Data	Opis
23.02.2016	Wybór tematu projektu.
1.03.2016	Określenie zakresu tematycznego projektu.
8.03.2016	Przekazanie specyfikacji projektu (wstępna funkcjonalność, określenie kamieni milowych).
9.03.2016 – 06.06.2016	Konsultacja wyników pracy po osiągnięciu kolejnych kamieni milowych projektu.
7.06.2016	Prezentacja rezultatów projektu i przekazanie dokumentacji projektowej.

Tabela 3: Kamienie milowe.

Data	Opis
29.03.2016	Instalacja środowiska
19.04.2016	Stworzenie lematyzatora
10.05.2016	Stworzenie taggera
31.05.2016	Stworzenie instrukcji użytkowania
07.06.2016	Prezentacja projektu

4 Przygotowanie środowiska

Wymagany Python 3.5.

4.1 Wirtualne środowisko (opcjonalnie)

```
$ python3 -m venv env               # instalacja venv
$ source env/bin/activate           # aktywacja venv

  ... praca w wirtualnym środowisku ...

$ deactivate                        # deaktywacja venv
```

4.2 Instalacja pakietów

Podczas instalacji pakietów pod Linuksem, potrzebny jest kompilator `gcc-fortran`.

```
$ pip3 install -r requirements.txt
```

4.3 Instalacja korpusów

Należy uruchomić skrypt instalacyjny, który rozpakowywuje korpus PWr z pliku kpwr-1.2.6-disamb.7z a korpus polski pobiera ze źródła.

```
$ ./installCorpuses.sh
```

4.4 Instalacja NLTK

Do tokenizacji słów w podanym tekście użyto NLTK. Dwie opcje instalacji:

1. Interaktywna instalacja w interpreterze:

   ```
   >>> import nltk
   >>> nltk.download()
   ```

2. Instalacja poprzez linię komend:

   ```
   $ sudo python -m nltk.downloader -d /usr/local/share/nltk_data all
   ```

   Dane NLTK zostaną zainstalowane w katalogu `/usr/local/share/nltk_data`.

4.5 Instalacja CUDA

1. Pobranie paczki instalującej repozytorium Nvidii ze strony: https://developer.nvidia.com/cuda-downloads, testowana wersja: Linux Ubuntu 14.04, architektura x86_64, paczka deb (local).
2. Instalacja repozytorium w systemie:

   ```
   $ dpkg -i cuda-repo-ubuntu1404-7-5-local_7.5-18_amd64.deb‘
   ```

3. Instalacja sterowników i środowiska CUDA:

   ```
   $ apt-get update && apt-get install -y cuda
   ```

4. Restart maszyny w celu załadowania sterowników Nvidii zamiast Nouveau.
5. Instalacja Nvidia cuDNN (biblioteki wspomagające sieci neuronowe): należy umieścić zawartość archiwum `cudnn-7.0-linux-x64-v4.0-prod.tgz` w folderze `/usr/local/cuda`. Do ściągnięcia ze strony https://developer.nvidia.com/rdp/form/cudnn-download-survey.
6. Instalacja modułu tensorflow dla Pythona.

5 Korzystanie z programu

5.1 Konfiguracja środowiska

Należy dodać folder zawierający projekt do zmiennej środowiskowej `PYTHONPATH`. Jeśli akurat się w nim znajdujemy (jest on katalogiem bieżącym), można to zrobić np. poprzez:

```
$ export PYTHONPATH="${PYTHONPATH}:${PWD}"
```

Aby wykonać powyższą komendę wraz z włączeniem `venv`, wystarczy pobrać zawartość pliku `prepare` do shella poprzez:

```
$ source prepare
```

5.2 Ustawienia ścieżek

Skrypty korzystają ze ścieżek konfigurowalnych za pomocą pliku src/settings.py.

5.3 Konwersja plików xml do csv

Projekt zawiera skrypt umożliwiający konwersję plików xmlowych do formatu csv:

```
$ python3 src/scripts/csv_creator.py -h
usage: csv_creator.py [-h] (--use_pwr | --use_national) [--extract_base_words]
optional arguments:
-h, --help            show this help message and exit
--use_pwr             Use pwr corpus
--use_national        Use national corpus
--extract_base_words  Save words with their base form
```

5.4 Tworzenie bazy końcówek

Do stworzenia bazy końcówek służy plik src/scripts/suffix_creator.py.

5.5 Tagger - uczenie klasyfikatorów

```
$ python3 src/tagger_trainer.py -h  # więcej o ustawianiu liczby rdzeni
$ python3 src/tagger_trainer.py
```

Domyślnie podczas uczenia używane są wszystkie rdzenie procesora - jeden rdzeń na algorytm. Logi związane z uczeniem zapisywane są do pliku `tagger_factory.log`.

5.6 Testowanie taggera

```
$ python3 src/tagger_tester.py  # pojedyncze słowo
$ python3 src/text_tagger.py    # tekst
```

Skrypt zapyta się o słowo/tekst do klasyfikacji.

5.7 Uruchomienie benchmarka

```
$ python3 src/benchmark.py
```

6 Wyniki pomiarów działania programu

6.1 Dokładność modeli w bazie

Nazwa algorytmu	Dokładność [%] - korpus uczący PWr	Dokładność [%] - korpus uczący national
Support Vector Machine	56.468	54.949
Decision Trees	57.246	55.671
Stochastic Gradient Descent	53.627	53.121
Logistic Regression	55.571	55.307
Naive Bayes	54.141	52.813
K Neighbors	50.168	50.255
Neural Networks	55.801	54.692

6.2 Czas uczenia

Nazwa algorytmu	Korpus uczący PWr `(h:m:s)`	Korpus uczący national `(h:m:s)`
Support Vector Machine	2:05:03	45:46:28
Decision Trees	0:00:30	0:02:24
Stochastic Gradient Descent	0:00:29	0:02:09
Logistic Regression	0:01:02	0:05:41
Naive Bayes	0:00:21	0:01:51
K Neighbors	0:00:23	0:01:49
Neural Networks	0:35:20	2:21:40

Pomiar czasowy wykonany na procesorze Intel(R) Core(TM) i5-2540M CPU @ 2.60GHz.