AKADEMIA GÓRNICZO-HUTNICZA

im. Stanisława Staszica w Krakowie

WYDZIAŁ INŻYNIERII MECHANICZNEJ I ROBOTYKI

Praca dyplomowa
inżynierska

Wojciech Wróbel
Imię i nazwisko

Inżynieria Akustyczna
Kierunek studiów

Korpus językowy na potrzeby budowy głosu w syntezie TTS
Temat pracy dyplomowej

Dr inż. Jakub Gałka
Promotor pracy

……..
Ocena

Kraków, rok 2016/2017

Kraków, dnia.....

Imię i nazwisko : Wojciech Wróbel
Nr albumu : 269569
Kierunek studiów : Inżynieria Akustyczna
Profil dyplomowania : -

OŚWIADCZENIE

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (tj. Dz.U.z 2006 r. Nr 90, poz. 631 z późn.zm.) : „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie", a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust.1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (tj. Dz.U. z 2012 r. poz. 572, z późn.zm.) „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej „sądem koleżeńskim", oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem (-am) ze źródeł innych niż wymienione w pracy".

.....
podpis dyplomanta

Kraków, dn……..

Imię i nazwisko: Wojciech Wróbel

Nr albumu: 269569

Kierunek studiów: Inżynieria Akustyczna

Profil dyplomowania: -

OŚWIADCZENIE

Świadomy/a odpowiedzialności karnej za poświadczanie nieprawdy oświadczam, że niniejszą inżynierską pracę dyplomową wykonałem/łam osobiście i samodzielnie oraz nie korzystałem/łam ze źródeł innych niż wymienione w pracy.

Jednocześnie oświadczam, że dokumentacja oraz praca nie narusza praw autorskich
w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz. U. z 2006 r. Nr 90 poz. 631 z późniejszymi zmianami) oraz dóbr osobistych chronionych prawem cywilnym. Nie zawiera ona również danych i informacji, które uzyskałem/łam w sposób niedozwolony. Wersja dokumentacji dołączona przeze mnie na nośniku elektronicznym jest w pełni zgodna z wydrukiem przedstawionym do recenzji.

Zaświadczam także, że niniejsza inżynierska praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów wyższej uczelni lub tytułów zawodowych.

……..
podpis dyplomanta

Kraków, ……..

Imię i nazwisko : Wojciech Wróbel

Adres korespondencyjny : os. Kamionki 15, 32-566 Alwernia

Temat pracy dyplomowej inżynierskiej : Korpus językowy na potrzeby budowy głosu w syntezie TTS

Rok ukończenia : 2017

Nr albumu : 269569

Kierunek studiów : Inżynieria Akustyczna

Profil dyplomowania : -

OŚWIADCZENIE

Niniejszym oświadczam, że zachowując moje prawa autorskie , udzielam Akademii Górniczo-Hutniczej im. S. Staszica w Krakowie nieograniczonej w czasie nieodpłatnej licencji niewyłącznej do korzystania z przedstawionej dokumentacji inżynierskiej pracy dyplomowej, w zakresie publicznego udostępniania i rozpowszechniania w wersji drukowanej i elektronicznej[1].

Publikacja ta może nastąpić po ewentualnym zgłoszeniu do ochrony prawnej wynalazków, wzorów użytkowych, wzorów przemysłowych będących wynikiem pracy inżynierskiej[2].

Kraków, ...… ……..
data *podpis dyplomanta*

[1] Na podstawie Ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (Dz.U. 2005 nr 164 poz. 1365) Art. 239. oraz Ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz.U. z 2000 r. Nr 80, poz. 904, z późn. zm.) Art. 15a. "Uczelni w rozumieniu przepisów o szkolnictwie wyższym przysługuje pierwszeństwo w opublikowaniu pracy dyplomowej studenta. Jeżeli uczelnia nie opublikowała pracy dyplomowej w ciągu 6 miesięcy od jej obrony, student, który ją przygotował, może ją opublikować, chyba że praca dyplomowa jest częścią utworu zbiorowego."

[2] Ustawa z dnia 30 czerwca 2000r. – Prawo własności przemysłowej (Dz.U. z 2003r. Nr 119, poz. 1117 z późniejszymi zmianami) a także rozporządzenie Prezesa Rady Ministrów z dnia 17 września 2001r. w sprawie dokonywania i rozpatrywania zgłoszeń wynalazków i wzorów użytkowych (Dz.U. nr 102 poz. 1119 oraz z 2005r. Nr 109, poz. 910).

Kraków, dnia .....

AKADEMIA GÓRNICZO-HUTNICZA
WYDZIAŁ INŻYNIERII MECHANICZNEJ I ROBOTYKI

TEMATYKA PRACY DYPLOMOWEJ INŻYNIERSKIEJ
dla studenta IV roku studiów stacjonarnych

.....
imię i nazwisko studenta

TEMAT PRACY DYPLOMOWEJ INŻYNIERSKIEJ:
Korpus językowy na potrzeby budowy głosu w syntezie TTS

Promotor pracy: dr inż. Jakub Gałka

Recenzent pracy: prof. dr hab inż. Mariusz Ziółko *Podpis dziekana:*

PLAN PRACY DYPLOMOWEJ (Należy ustalić z Promotorem, np.:)
1. Omówienie tematu pracy i sposobu realizacji z promotorem.
2. Zebranie i opracowanie literatury dotyczącej tematu pracy.
3. Zebranie i opracowanie wyników badań.
4. Analiza wyników badań, ich omówienie i zatwierdzenie przez promotora.
5. Opracowanie redakcyjne.

Kraków, ...… ……..
data *podpis dyplomanta*

TERMIN ODDANIA DO DZIEKANATU: ..... 20..... r.

.....
podpis promotora

Akademia Górniczo-Hutnicza im. Stanisława Staszica Kraków, ...
Wydział Inżynierii Mechanicznej i Robotyki

Kierunek: Inżynieria Akustyczna

Profil dyplomowania: -

Wojciech Wróbel
Praca dyplomowa inżynierska
Korpus językowy na potrzeby budowy głosu w syntezie TTS
Opiekun: dr inż. Jakub Gałka

STRESZCZENIE

Przedmiotem niniejszej pracy jest przygotowanie korpusu językowego, stanowiącego podstawę do metod uczenia maszynowego, przy budowie automatycznego klasyfikatora, na potrzeby normalizacji tekstu w systemie syntezy mowy. Praca zawiera wstęp teoretyczny przedstawiający ogólny zarys działania systemu TTS oraz zagadnienia związane z normalizacją tekstu, oraz jej metodyką.

Część praktyczna nakierowana jest na analizę potrzeb normalizacji tekstu oraz przygotowanie odpowiednich narzędzi programowych, niezbędnych do zautomatyzowanej budowy korpusu. Przedstawione zostaną kryteria wyboru klas do korpusu, oraz szczegóły implementacji skryptów. W tej części opisano również zasoby i narzędzia, które wykorzystano do realizacji pracy.

Ostatnia część to analiza przygotowanego korpusu oraz podsumowanie wszystkich działań. Znajduje się tam opis zawartości korpusu oraz ocena jego przydatności.

AGH University of Science and Technology Kraków, the.....
Faculty of Mechanical Engineering and Robotics
Field of Study: Acoustic engineering
Specialisations: -

Wojciech Wróbel
Engineer Diploma Thesis
Language corpus for voice building for TTS synthesis
Supervisor: Ph. D. Jakub Gałka

SUMMARY

The subject of this work is to prepare a language corpus for machine learning purposes for creating automatic text classifier in text normalization modules in Text-to-Speech systems. Theoretical introduction is focused on TTS basics and issues related to the normalization of the text, and its methodology.

Practical part is focused on text normalization needs and preparation of appropriate software tools necessary for the automated construction of the corpus. The criteria of classes selection and details of script implementation will be explained. This section also describes resources and tools that were used in this work.

The last part is the analysis of the prepared language corpus and a summary of all steps. There is a description of the results and an estimation of output sentences usefulness.

Spis treści

1. Wstęp

Technologia mowy staje się coraz bardziej popularnym tematem na świecie i znajduje swoje zastosowanie w wielu aplikacjach. Automatyczne rozpoznawanie mowy ASR (ang. *Automatic Speech Recognition*) jak i synteza mowy TTS (ang. *Text-to-Speech*) to zagadnienia często mianowane pojęciem innowacji, dlatego że ich głównym celem jest diametralna zmiana sposobu interakcji człowieka z maszyną – poprzez interfejs głosowy.

Jak wskazuje temat pracy, dotyczy ona obszaru technologii mowy jakim jest synteza TTS, czyli konwersji tekstu na sygnał mowy. Jest to złożony proces na który składa się wiele etapów. Jednym z nich jest normalizacja tekstu odpowiedzialna za jego przygotowanie do odczytania przez system TTS tj. wykrycie i werbalizację np. liczb, dat oraz skrótów. Korpus językowy, o którym mowa w temacie, jest zasobem który może okazać się przydatny właśnie na tym etapie.

Podsumowując można stwierdzić, że temat syntezy TTS jest zagadnieniem jak najbardziej aktualnym i interesującym we współczesnym świecie, lecz nadal pozostaje w nim wiele aspektów wymagających udoskonalenia, dlatego też w niniejszej pracy podjęto próbę jego rozwoju.

1.1 Cel i zakres pracy

Głównym celem pracy jest utworzenie korpusu językowego, składającego się ze zdań opisanych tzw. tagami (szczegółowy opis tagów w podrozdziale 3.2.3), czyli oznaczeniami, że dany wycinek zdania należy do określonej klasy wymagającej późniejszej normalizacji w procesie TTS. Korpus ten będzie bazą do uczenia automatycznego taggera w przyszłości projektu TTS. Celem pośrednim, niezbędnym do budowy korpusu, jest zaprojektowanie skryptu, który umożliwi sprawne poruszanie się po strukturze NKJP (Narodowy Korpus Języka Polskiego) oraz zautomatyzowane pozyskiwanie danych. Narzędzie to będzie miało na celu wyszukiwanie odpowiednich zdań do korpusu oraz ich opis za pomocą właściwego tagu. Niezbędnym etapem pracy będzie również analiza potrzeb normalizacji tekstu.

Pierwsza część pracy nakierowana jest na zagadnienia teoretyczne ściśle związane z tematem. Omówione zostanie przeznaczenie normalizacji tekstu i dlaczego jest ona

tak istotna w systemach TTS. Poruszona będzie również kwestia licznych problemów przy projektowaniu modułów normalizacji oraz różne podejścia ich eliminacji.

Druga część pracy to opis wykorzystanych narzędzi oraz zasobów tekstowych przy budowie korpusu językowego. Zostaną również opisane kroki jakie zostały podjęte w realizacji niniejszej pracy i szczegóły implementacji skryptów.

Kolejnym elementem jest opis i analiza produktu, czyli gotowego korpusu językowego. Oceniona zostanie jego przydatność oraz zostanie poddany analizie statystycznej.

W zakończeniu zebrane zostaną wszystkie wnioski oraz przemyślenia, które nasunęły się w trakcie całego procesu realizacji pracy. To miejsce na podsumowanie wszystkich działań.

2. Wprowadzenie

Idea działania systemu syntezy mowy TTS jest prosta - dokonać konwersji wprowadzonego tekstu na sygnał mowy czyli sygnał dźwiękowy. Pomysł zaprojektowania maszyny zamieniającej tekst na mowę istniał już od dawna. Od ponad 50 lat naukowcy zmagają się z próbą odtworzenia zjawisk fizycznych jakie mają miejsce w torze głosowym, podczas generacji mowy przez człowieka [1]. Podejście to nosi nazwę artykulacyjnej syntezy mowy i opiera się na artykulacyjnych modelach ludzkiego traktu głosowego. Sygnał mowy uzyskany tym sposobem był zazwyczaj nienaturalny i nieakceptowalny do użytku w rzeczywistych zastosowaniach. Pod koniec lat siedemdziesiątych powstała idea generowania mowy poprzez łączenie wcześniej nagranych jednostek mowy (najczęściej difonów). Taka technika nosi nazwę konkatenacyjnej syntezy mowy i dzięki niej udało się wygenerować mowę o dużym stopniu zrozumiałości, jednak nadal była bardzo nienaturalna. Przełom w rozwoju syntezy mowy miał miejsce pod koniec lat osiemdziesiątych. Odkryto wówczas że kiedy baza jednostek mowy zawiera więcej niż tylko jedną wersję tego samego difonu, możliwe jest uzyskanie mowy która brzmi naturalniej. Dzieje się tak, ponieważ możemy dokonać wtedy selekcji najbardziej odpowiednich jednostek mowy (ang. *unit selection*), które nigdy nie są identyczne w brzmieniu – zależy ono od umiejscowienia w słowie lub zdaniu oraz innych czynników. Ten typ syntezy nosi nazwę korpusowej syntezy mowy i jest modyfikacją syntezy konkatenacyjnej. Istotnym problemem jest tutaj konwersja surowego tekstu na sekwencję fonemów z dołączoną informacją

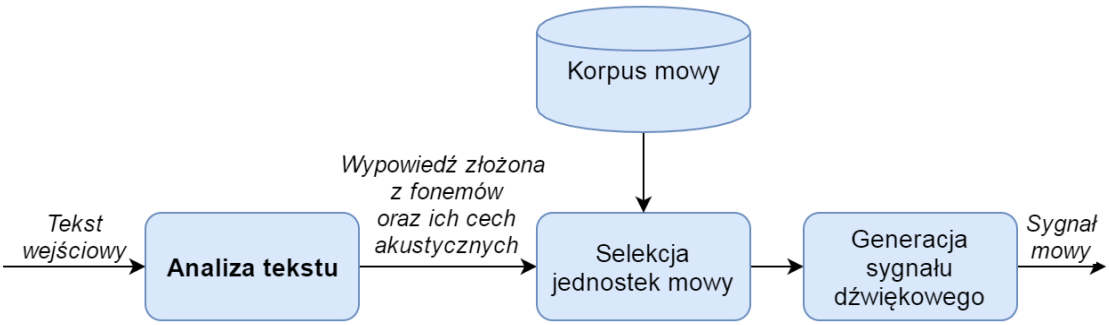
prozodyczną o czasie trwania dźwięku, głośności, akcencie, wysokości, pauzach i innych aspektach, które mają wpływ na wymowę [2].

Stosowanie systemów typu TTS motywuje to, że mowa jest najbardziej naturalnym sposobem komunikowania się ludzi. Z tego względu sygnał mowy, w przyszłości może okazać się najlepszym medium do komunikacji człowieka z maszyną. Synteza mowy ma coraz szersze zastosowanie we współczesnym świecie. Przykładowo technologię tę wykorzystuje się w IVR (ang. *Interactive Voice Response*), czyli systemach telekomunikacyjnych, umożliwiających interaktywną obsługę osób dzwoniących [3]. Kolejną dziedziną są portale głosowe oraz wirtualni doradcy, którzy reprezentują określoną bazę danych i mają ułatwić klientowi dostęp do interesujących go informacji. Ważnym aspektem syntezy mowy jest również ułatwienie dostępu do wiadomości tekstowych dla osób niewidomych, bądź z wadami wzroku, a także przekaz komunikatów w relacji samochód – kierowca. Poza wymienionymi przykładami istnieje jeszcze niezliczona ilość dziedzin, w których zastosowanie technologii TTS może okazać się interesujące.

3. Podstawy teoretyczne

3.1 Korpusowa synteza mowy

Poniżej przedstawiono ogólny schemat działania korpusowej syntezy mowy TTS (Rys. 3.1). Niniejsza praca skupia się wyłącznie na pierwszym z modułów takiego systemu jakim jest analiza tekstu, dlatego dalsze etapy zostaną jedynie zaznaczone bez wgłębiania się w ich szczegóły.



Rys. 3.1 Schemat ideowy korpusowej syntezy mowy

3.2 Analiza tekstu

3.2.1 Przebieg analizy tekstu

Pierwszym etapem analizy tekstu jest jego wstępne przetwarzanie (ang. *pre-processing*). Jego celem jest rozwiązanie problemów typu: identyfikacja rodzaju tekstu, kodowanie znaków oraz obecność wyrazów obcych. Następnie tekst dzielony jest na zdania, gdyż będą one podstawowym zakresem dalszej analizy lingwistycznej. Dokonuje się również podziału tekstu na tokeny, czyli najczęściej słowa, dla których można określić prawidłową wymowę np. za pomocą słownika. Dla wielu języków (w tym języka polskiego) podział na tokeny będzie odbywał się w miejscu spacji w surowym tekście [1]. Kolejnym etapem jest właściwa analiza tekstu składająca się z dwóch podstawowych części: klasyfikacji semiotycznej oraz werbalizacji. Klasyfikacja semiotyczna ma za zadanie przyporządkować każdemu tokenowi jedną z klas semiotycznych, czyli zadecydować czy badany token jest zwykłym słowem, skrótem, liczbą, datą lub należy do innej klasy. Z kolei werbalizacja sprowadza się do konwersji elementów nie należących do języka naturalnego na słowa [2]. Następnym etapem jest usunięcie jakichkolwiek dwuznaczności spowodowanych występowaniem homografów, czyli wyrazów o identycznej pisowni, ale innej wymowie i znaczeniu [2][4]. Ostatnim procesem analizy tekstu jest predykcja prozodii, czyli odpowiednika melodii w mowie. Wynikiem tego procesu będzie zbiór cech akustycznych jednostek mowy, z których system będzie korzystał w następnym module TTS czyli selekcji jednostek mowy.

3.2.2 Zarys zagadnienia normalizacji tekstu

Jednym z zadań pierwszego modułu systemu TTS jest sprowadzenie wejściowego tekstu do postaci języka naturalnego, aby nadawał się do dalszego przetwarzania lingwistycznego [2]. Warto rozważyć poniższy przykład, aby lepiej zrozumieć idee działania takiego modułu. Założono że na wejście trafia następujący ciąg znaków:

„*11.11.2016 o godz. 17:30 z peronu 4 odjeżdża pociąg nr. 8.*”

Aby odpowiednio odczytać takie zdanie, należy poprawnie zinterpretować symbole i liczby które się w nim pojawiły. Wydaje się to proste, gdyż mózg człowieka

automatycznie dobiera odpowiednie słowa które są sensowne dla danej wypowiedzi tj. dokonuje decyzji czym jest dany element, wybiera odpowiednią formę gramatyczną pasującą do kontekstu oraz dokonuje werbalizacji, czyli tworzy słowa. Tym sposobem powyższy przykład zostanie automatycznie odczytany przez człowieka jako:

„*jedenastego listopada dwa tysiące szesnaście o godzinie siedemnastej trzydzieści z peronu czwartego odjeżdża pociąg numer osiem*”

Warto zwrócić uwagę, że oprócz werbalizacji liczb odbyła się tutaj również werbalizacja skrótów (*godz.* ; *nr.*). W przeciwieństwie do łatwości czytania tekstu przez człowieka, implementacja tego procesu na komputerze jest bardzo złożona i problem ten nie został jeszcze w pełni rozwiązany we współczesnych systemach TTS. Główną trudnością nie jest tutaj werbalizacja, ale samo dokonanie decyzji do jakiej klasy semiotycznej należy każdy z tokenów [2].

3.2.3 Tagi do normalizacji tekstu

W celu ułatwienia pracy na etapie normalizacji tekstu zdefiniowano zestaw tagów do opisu tokenów, które nie są przedstawicielami języka naturalnego i wymagają werbalizacji. Na początku należało zdecydować jakie tagi podstawowe można wyróżnić dla języka polskiego. Na podstawie analizy tekstów różnego pochodzenia, zadecydowano że należy otagować odrębnymi tagami: liczby całkowite, liczby całkowite ze znakiem, liczby rzeczywiste, liczby porządkowe, liczby rzymskie, ułamki, sekwencje cyfr, jednostki, waluty, godziny, daty, zakresy liczbowe, skróty, literowce, kody pocztowe, telefony, e-maile, adresy internetowe oraz identyfikatory. Przyjęto następujący format tagowania tekstu przypominający nieco język znaczników XML (ang. *Extensible Markup Language*):

<tag>token</tag>

Kompletne zestawienie tagów oraz ich przykłady znajdują się w tabeli (tab. 3.1).

Tab. 3.1 Tagi do normalizacji tekstu, ich oznaczenia oraz przykłady

Nazwa	Tag	Przykład
Liczba całkowita	**cardinal**	<cardinal>10000</cardinal>
Liczba całkowita ze znakiem	**signed**	<signed>+5</signed>
Liczba rzeczywista	**real**	<real>4,5</real>
Liczba porządkowa	**ordinal**	<ordinal>21</ordinal>
Liczba rzymska	**roman**	<roman>LI</roman>
Ułamek	**fraction**	<fraction>1/5</fraction>
Sekwencja cyfr	**digits**	<digits>0123</digits>
Jednostki	**unit**	<unit>15 dB</unit>
Waluty	**currency**	<currency>10 zł</currency>
Czas	**time**	<time>2:00</time>
Data	**date**	<date>12.05.2001</date>
Zakres	**range**	<range>3 - 5</range>
Skróty	**abbrev**	<abbrev>np.</abbrev>
Literowce	**initialism**	<initialism>HTML</initialism>
Kod pocztowy	**postal**	<postal>32-566</postal>
Telefon	**phone**	<phone>663123423</phone>
E-mail	**email**	<email>abc@gmail.com</email>
Adres internetowy	**web**	<web>www.techmo.pl</web>
Identyfikator	**ident**	<ident>z126p</ident>

3.2.4 Dodatkowe atrybuty tagów

W odróżnieniu do języka angielskiego, w języku polskim do jednoznacznej i prawidłowej werbalizacji tokenu, oprócz wiedzy o klasie semiotycznej do której przynależy, niezbędna jest również znajomość formy gramatycznej, czyli na potrzeby TTS najczęściej przypadku, rodzaju oraz liczby. W tym celu do podstawowego formatu tagowania dodano opcjonalne atrybuty opisujące formę gramatyczną na podstawie

oznaczeń z systemu znaczników morfosynktatycznych w korpusie IPI PAN [5]. Atrybuty te zestawiono w tabeli (tab. 3.2).

Tab. 3.2 Dodatkowe atrybuty opisujące formę gramatyczną tokenów

Kategoria gramatyczna	**Wartość**	**Oznaczenie**
Liczba (num)	pojedyncza	sg
	mnoga	pl
Przypadek (case)	mianownik	nom
	dopełniacz	gen
	celownik	dat
	biernik	acc
	narzędnik	inst
	miejscownik	loc
	wołacz	voc
Rodzaj (gen)	męski osobowy	m1
	męski zwierzęcy	m2
	męski rzeczowy	m3
	Żeński	f
	nijaki zbiorowy	n1
	nijaki zwykły	n2
	przymnogi osobowy	p1
	przymnogi zwykły	p2
	przymnogi opisowy	p3

Ze względu na duże znaczenie formy gramatycznej przy rozwijaniu niektórych tagów zmodyfikowano format tagowania na poniższy:

<tag kategoria_gramatyczna=”value”>token</tag>

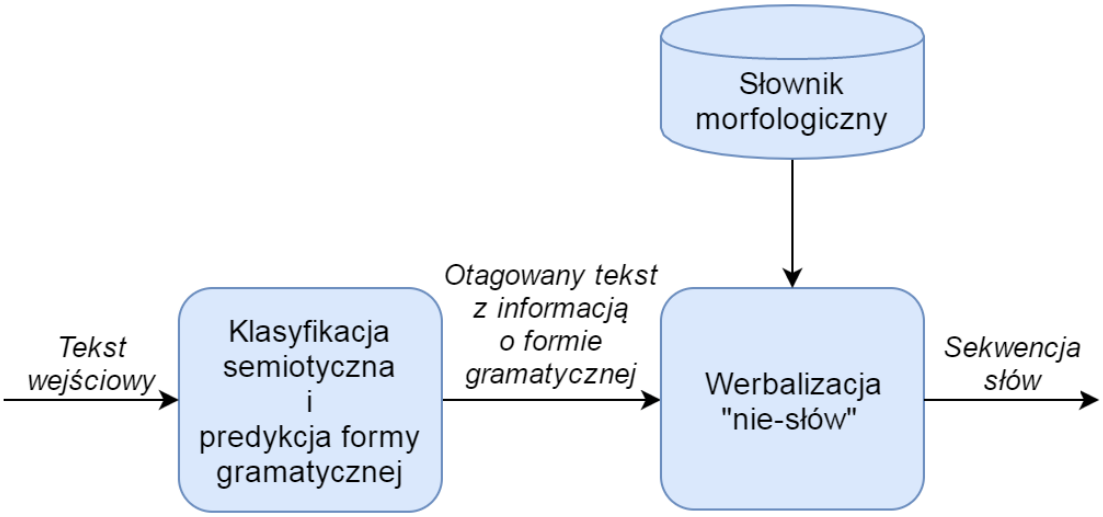
Jak wspomniano wcześniej atrybuty określające formę gramatyczną tokenu są opcjonalne, gdyż nie dla wszystkich klas semiotycznych mają one znaczenie przy werbalizacji. Tak więc w praktyce otagowany token może zawierać jeden, dwa lub trzy atrybuty, bądź nie zawierać ani jednego, jeśli forma gramatyczna nie ma znaczenia. Poniżej, dla lepszego przedstawienia idei tagowania, podano przykład prawidłowo otagowanego tokenu według ustalonego formatu:

<cardinal case=”inst” gen=”f”>8</cardinal>

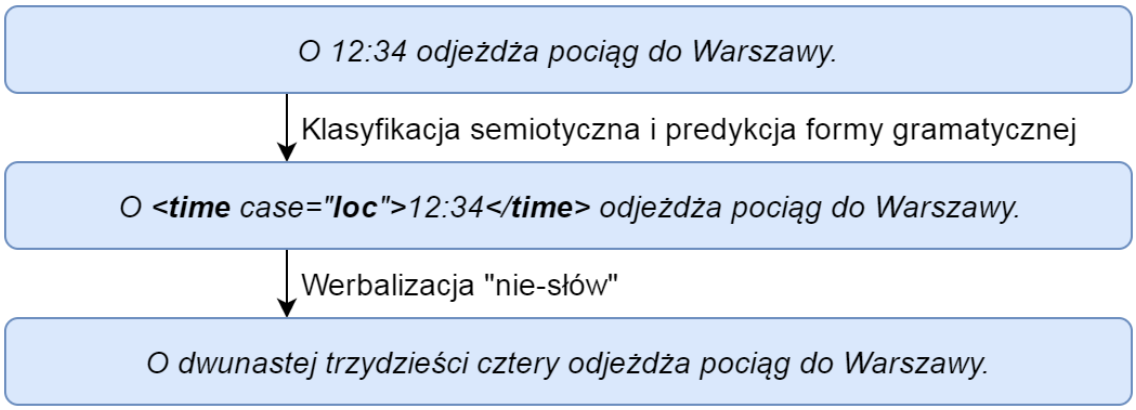
Moduł werbalizacji dostając taką wiadomość otrzymuje informację, że symbol ‘8’ należy traktować jako liczbę całkowitą (tab. 3.1) oraz że przypadek tej liczby to narzędnik i jest ona rodzaju żeńskiego (tab. 3.2). Szczegółowa analiza potrzeb tagowania, jeśli chodzi o dobór koniecznych atrybutów znajduje się w rozdziale 4.

3.2.5 Przebieg normalizacji tekstu

Odnosząc się do poprzednich rozdziałów proces normalizacji tekstu można sprowadzić do dwóch głównych etapów: klasyfikacji semiotycznej oraz werbalizacji [2]. Ze względu na charakter języka polskiego przed werbalizacją należy dodatkowo określić formę gramatyczną badanych tokenów, która jest niezbędna do odnalezienia odpowiedniego słowa w słowniku morfologicznym podczas werbalizacji. Słownik morfologiczny zawiera zbiór wszystkich słów występujących w danym języku wraz z opisem ich formy gramatycznej. Dla języka polskiego takim zasobem może być PoliMorf [6]. Poniżej przedstawiono schemat ogólny przebiegu normalizacji tekstu (rys. 3.2) oraz jej kolejne etapy na przykładowym tekście wejściowym (rys. 3.3).



Rys. 3.2 Schemat ogólny normalizacji tekstu w systemie TTS



Rys. 3.3 Kolejne etapy przetwarzania przykładowego tekstu w module normalizacji tekstu

Otrzymując prawidłowo otagowany tekst z pierwszego modułu, jego werbalizacja przy wykorzystaniu słownika, bądź określonych reguł werbalizowania danej klasy semiotycznej jest zadaniem dosyć prostym do realizacji. Głównym wyzwaniem jest pierwszy etap czyli klasyfikacja semiotyczna i predykcja formy gramatycznej i to na tym zagadnieniu skupia się niniejsza praca w stopniu największym.

3.3 Klasyfikacja tekstu

3.3.1 Metody klasyfikacji tekstu

Zarówno w klasyfikacji semiotycznej jak i ustalaniu prawidłowego znaczenia homografów celem jest przypisanie do każdego z tokenów jednej z wcześniej zdefiniowanych etykiet. Proces ten można przedstawić jako funkcję C z przestrzeni cech F do przestrzeni etykiet L [2]:

$$C : F \mapsto L \tag{3.1}$$

gdzie

$$L = \{l_1, l_2, ...l_N\} \tag{3.2}$$

Warto rozważyć prosty przykład dla lepszego zrozumienia działania funkcji. Założono dwa przykładowe teksty wejściowe:

1. „*O godzinie 8 rano zaczynają się zajęcia.*”
2. „*Osoba nr. 8 proszona jest o kontakt.* ”

Założono również że w tym przypadku zestaw predefiniowanych etykiet (3.2) będzie reprezentowany poprzez 2 klasy: $L = \{l_1 = 8 - \text{GODZINA}, l_2 = 8 - \text{LICZBA}\}$. Analizując powyższy przykład można stwierdzić że słowa w otoczeniu symbolu, który należy sprowadzić do języka naturalnego, dają wskazówki co do jego prawidłowej postaci rozwinięcia. Takie wskazówki nazywa się cechami (ang. *features*) [2]. Zbiór cech oznacza się symbolem F (3.1). Obserwacje odnoszące się do przykładu zestawiono w tabeli (tab. 3.3).

Tab. 3.3 Zestawienie cech i możliwych wartości przy rozważaniu przykładowych tekstów

Oznaczenie	**Cecha**	**Możliwe wartości**
F_1	Czy poprzedzający token to 'godzinie' ?	tak, nie
F_2	Czy następny token to 'rano' ?	tak, nie
F_3	Czy poprzedzający token to 'nr.' ?	tak, nie

Na podstawie tabeli (tab. 3.3) działanie algorytmu klasyfikacji można sprowadzić do następującej postaci (3.3):

$$C(F_1, F_2, F_3) \mapsto l_1 \quad lub \quad l_2 \tag{3.3}$$

Opisywany przykład można zaliczyć do trywialnych, gdyż w rzeczywistości teksty wejściowe zwykle są o wiele bardziej złożone, bądź nie występują w nich charakterystyczne cechy umożliwiające klasyfikację bez ogólnego zrozumienia sensu zdania. Istnieje wiele różnych metod rozwiązania tego problemu. Mogą to być ręcznie projektowane reguły, jednak jest ich na tyle dużo, że byłoby to praktycznie nie możliwe do wykonania. Innym popularnym podejściem są metody bazujące na uczeniu maszynowym (ang. *machine learning*). Opiera się ono na analizie przez komputer obszernej bazy z przykładami i tworzeniu pewnego modelu klasyfikacji. Baza takich odpowiednio opisanych przykładów to właśnie korpus językowy którego dotyczy ta praca.

3.3.2 Wyrażenia regularne

Niektóre z klas semiotycznych mogą być dosyć łatwo rozpoznawane poprzez aparat wyrażeń regularnych (ang. *regular expressions*). Są to pewnego rodzaju wzorce opisujące łańcuchy symboli. Jeśli istnieje możliwość jednoznacznej klasyfikacji tokenu za pomocą analizy wyrażeń regularnych, nie trzeba wtedy korzystać z metod

wymienionych w podrozdziale 3.3.1 co zwiększa efektywność modułu normalizacji. W nawiasach kwadratowych takich wyrażeń podaje się dozwolone symbole poszukiwanej składni. Przykładowo wyrażenie: [0-9]+ będzie pasować do dowolnego ciągu liczb całkowitych. Plus po prawej stronie wyrażenia oznacza dowolną ilość wystąpień symboli w sekwencji. W tabeli (tab 3.4) przedstawiono podstawowe wyrażenia regularne, które mogą być przydatne przy klasyfikacji semiotycznej.

Tab. 3.4 Podstawowe wyrażenia regularne do klasyfikacji semiotycznej

Klasa	Wyrażenie regularne
Język naturalny	$[A-Za-z]+$
Liczba całkowita	$[0-9]+$
Liczba rzymska	$[IVXLCDM]+$
Godzina	$[0-9] \quad \vert \quad 1[0-9] \quad \vert \quad 2[0-4]$
Minuta	$[0-5][0-9]$

Nawet już przy podstawowych wyrażeniach regularnych (tab 3.4) pojawiają się konflikty oznaczeń, gdyż klasy dla danego wyrażenia mogą się pokrywać (np. godzina równocześnie jest liczbą całkowitą). Stanowią one duży problem przy klasyfikowaniu tekstu.

4. Analiza potrzeb normalizacji tekstu

4.1 Cel analizy

Celem analizy potrzeb normalizacji tekstu jest decyzja, które z klas semiotycznych mogą być w prosty sposób rozpoznane poprzez aparat wyrażeń regularnych, a które będą wymagać zastosowania bardziej złożonych metod (3.3.1), ze szczególnym nastawieniem na metody wykorzystujące uczenie maszynowe. Analiza ta określi jakie klasy mają znaleźć się w treningowym korpusie językowym, o którym mowa w tytule pracy.

4.2 Kryteria wyboru klas do korpusu

Można wyróżnić dwa główne kryteria do oceny przydatności umieszczenia danej klasy semiotycznej w korpusie treningowym. Jednym z nich jest to, czy forma gramatyczna ma wpływ na to w jaki sposób token zostanie zwerbalizowany. Jeśli

znajomość formy jest konieczna do wygenerowania prawidłowego słowa, należy wtedy odwołać się do bardziej zaawansowanych metod (3.3.1), w celu jej predykcji. Drugim kryterium jest niejednoznaczność aparatu wyrażeń regularnych. Jeśli składnia danej klasy pokrywa się z innymi, to za pomocą samych wyrażeń regularnych nie możliwa jest jej właściwa klasyfikacja i podobnie jak przy kryterium pierwszym, należy odwoływać się wtedy do innych metod klasyfikacji tekstu (3.3.1). W tabeli zestawiono wstępną analizę klas pod względem wyżej wymienionych kryteriów (tab. 4.1).

Tab. 4.1 Analiza klas pod kątem przydatności do korpusu

Klasa	**Tag**	**Czy forma gramatyczna ma znaczenie przy werbalizacji?**	**Czy składnia pozwala na jednoznaczną klasyfikacje?**
Liczba całkowita	**cardinal**	tak	nie
Liczba całkowita ze znakiem	**signed**	nie	tak
Liczba rzeczywista	**real**	nie	tak
Liczba porządkowa	**ordinal**	tak	nie
Liczba rzymska	**roman**	tak	tak
Ułamek	**fraction**	nie	tak
Sekwencja cyfr	**digits**	nie	nie
Jednostki	**unit**	tak	tak
Waluty	**currency**	tak	tak
Czas	**time**	tak	nie
Data	**date**	tak	nie
Zakres	**range**	nie	tak
Skróty	**abbrev**	nie	nie
Literowce	**initialism**	nie	nie
Kod pocztowy	**postal**	nie	tak
Telefon	**phone**	nie	nie
E-mail	**email**	nie	tak
Adres internetowy	**web**	nie	tak
Identyfikator	**ident**	nie	tak

Dodatkowym czynnikiem wpływającym na stopień przydatności klasy w korpusie jest prawdopodobieństwo jej wystąpienia w tekście wejściowym przy standardowym użytku. Należy pamiętać, że skuteczność systemu syntezy mowy TTS jest w dużym stopniu uwarunkowana odpowiednim zaprojektowaniem go pod określoną dziedzinę.

Zazwyczaj projektuje się je do typowych, z góry założonych zastosowań. Im dziedzina jest szersza, tym system będzie generował więcej błędów [1].

4.3 Klasy do uwzględnienia w korpusie

Z tabeli (tab. 4.1) wynika że w korpusie należałoby uwzględnić następujące klasy (spełnienie co najmniej jednego kryterium): liczba całkowita, liczba porządkowa, liczba rzymska, sekwencja cyfr, jednostki, waluty, czas, data, skróty, literowce, telefon. Niektóre klasy spełniają obydwa kryteria, przez co należałoby je uwzględnić w szczególności. Są nimi: liczba całkowita, liczba porządkowa, czas oraz data. Wiele z klas może mieć taką samą składnię jak liczba całkowita, co uniemożliwia poprawne rozpoznanie metodą wyrażeń regularnych. Należą do nich: liczba porządkowa, sekwencja cyfr, czas, data oraz telefon. Założono że liczba całkowita to klasa, która może wystąpić z największym prawdopodobieństwem, dlatego projektując algorytmy klasyfikacji będzie się ją prawdopodobnie traktowało jako klasa wyjściowa, a metody dodatkowe będzie się stosowało do rozpoznawania innych klas o tej samej składni. Jednak z drugiej strony klasy takie jak sekwencja cyfr oraz numer telefonu w małym stopniu zależą od słów w ich otoczeniu. Z tego względu zdania z takimi przykładami niekoniecznie nadają się do umieszczenia w korpusie językowym. Ponadto rzadko kiedy występują one w tekście ciągłym do tłumaczenia, szczególnie jeśli docelowo system TTS ma pracować na potrzeby IVR. W systemach IVR werbalizacja wiadomości typu: sekwencja cyfr (np. PESEL), dane kontaktowe (adres, kod pocztowy, telefon, e-mail) czy adresy internetowe powinna być ułatwiona poprzez odpowiednie zaprojektowanie przebiegu specyficznych dialogów ze świadomością na którym etapie zajdzie potrzeba wypowiedzenia konkretnej klasy. Na podstawie analizy potrzeb zdecydowano stworzyć korpus stanowiący bazę zdań z datami, godzinami oraz liczebnikami porządkowymi.

5. Budowa korpusu językowego

5.1 Zasoby i narzędzia

5.1.1 Narodowy Korpus Języka Polskiego

Najważniejszym zasobem przy budowie korpusu do treningu automatycznego klasyfikatora tekstu jest NKJP (Narodowy Korpus Języka Polskiego). Jest to największy, morfologicznie anotowany zbiór danych języka polskiego [7]. Do tworzenia korpusu będącego przedmiotem tej pracy wykorzystano ręcznie anotowany milionowy podkorpus NJKP, dostępny na licencji GNU GPL v.3. Podkorpus ten liczy nieco ponad milion tradycyjnie rozumianych słów i zawarto w nim oznaczenia typu: segmentacja tekstu na zdania i słowa, znakowanie morfosynktatyczne, znakowanie sensami słów (w ograniczonym zakresie), powierzchowne znakowanie składniowe (anotacja słów składniowych oraz prostych grup składniowych) oraz znakowanie jednostek nazewniczych [7]. Teksty NKJP pochodzą z różnych źródeł, ich zestawienie wraz z udziałem procentowym przedstawiono w tabeli (tab. 5.1).

Tab. 5.1 Kategorie tekstów NKJP (Źródło: [7])

Kategoria	Udział w korpusie
Dzienniki	25,5%
Pozostałe periodyki	23,5%
Książki publicystyczne	1,0%
Literatura piękna	16,0%
Literatura faktu	5,5%
Typ informacyjno-poradnikowy	5,5%
Typ naukowo-dydaktyczny	2,0%
Internetowe interaktywne (blogi, fora, Usenet)	3,5%
Internetowe nieinteraktywne (statyczne strony, Wikipedia)	3,5%
Quasi-mówione (protokoły sesji parlamentu)	2,5%
Mówione medialne	2,5%
Mówione konwersacyjne	5,0%
Inne teksty pisane	3,0%
Książka niebeletrystyczna nieklasyfikowana	1,0%

5.1.2 Struktura NKJP

Oznaczenia zawartości NKJP wymienione wcześniej (5.1.1) mają formę oddzielnych plików, które z kolei umieszczono w katalogach nazwanych zgodnie

z tytułem źródła tekstów. Każdy z nich ma postać dokumentu w formacie XML. Wyróżniono następujące pliki:

- header.xml - nagłówek zawierający informacje o pochodzeniu tekstu,
- text.xml - właściwy tekst, wraz z informacją o jego strukturze,
- ann_segmentation.xml - reprezentacja segmentacji tekstu na zdania i segmenty,
- ann_senses.xml - informacja o znaczeniach wybranych słów wieloznacznych,
- ann_morphosyntax.xml - interpretacje morfosynktatyczne poszczególnych segmentów,
- ann_words.xml - słowa składniowe i ich interpretacje morfosynktatyczne,
- ann_groups.xml - grupy składniowe,
- ann_named.xml - jednostki nazewnicze [7]

5.1.3 Python

Rys. 5.1 Logo Python (Źródło: python.org)

Python jest językiem programowania wysokiego poziomu ogólnego przeznaczenia. Dobrze nadaje się do przetwarzania danych lingwistycznych oraz jest bardzo prosty w użyciu [8]. Z tego względu zdecydowano się na implementację skryptu do automatycznego gromadzenia zdań do korpusu właśnie w tym języku.

5.1.4 Beautiful Soup

Beautiful Soup to biblioteka działająca w języku Python, która służy do pozyskiwania danych z plików typu HTML lub XML. Ułatwia nawigację, przeszukiwanie oraz modyfikację takich plików [9].

5.2 Implementacja skryptu

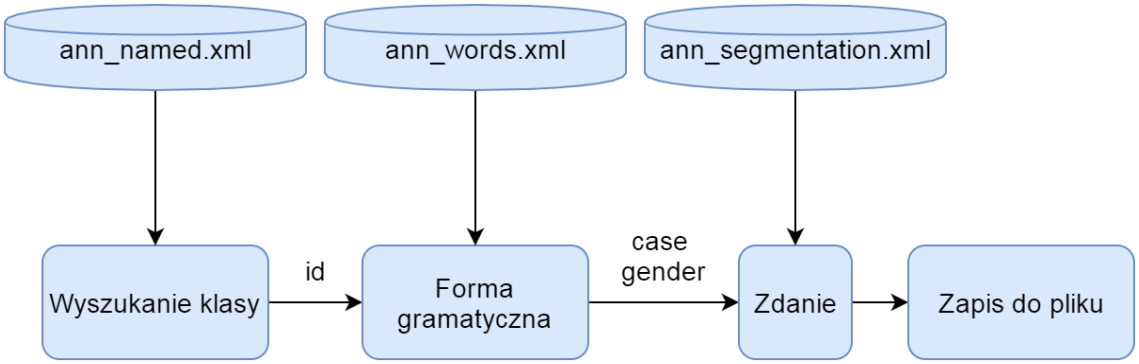
5.2.1 Założenia projektowe

Głównym przeznaczeniem skryptu jest wydobycie i odpowiedni opis wszystkich zdań z NKJP, w których występuje co najmniej jedna z klas wyznaczonych

na etapie analizy potrzeb (4.3). Sprowadza się to do przefiltrowania całego NKJP w celu wyszukania zdań z datami, godzinami oraz liczebnikami porządkowymi (4.3). Oprócz samego pozyskania zdań równie ważny jest opis ich formy gramatycznej według ustalonego wcześniej formatu (3.2.4).

5.2.2 Opis działania skryptu

W ogólnym przypadku skrypt wykorzystuje trzy spośród ośmiu podstawowych plików NKJP. Idea jego działania polega na znalezieniu odpowiedniej klasy i pobraniu id słowa lub słów które ją reprezentują (*ann_named.xml*), następnie odszukaniu informacji o formie gramatycznej (*ann_words.xml*) oraz wydobyciu całego zdania (*ann_segmentation.xml*). Zdarzają się katalogi w których brakuje pliku *ann_words.xml*, należy wtedy poszukiwać formy w *ann_morphosyntax.xml*. Po uzyskaniu wszystkich koniecznych informacji wydobyte zdania są modyfikowane poprzez dodanie do nich odpowiedniego tagu wraz z opisem formy gramatycznej. Taki proces powtarzany jest dla każdego katalogu w NKJP, czyli 3889 razy. W rzeczywistości jednak skrypty do wyciągania poszczególnych klas nieco się różnią z powodu różnic w ich opisie przez NKJP. Zmodyfikowane zdania zapisywane są w postaci pliku tekstowego i stanowią korpus który jest przedmiotem niniejszej pracy. Dodatkową funkcjonalnością skryptu jest pomiar czasu działania oraz licznik wystąpień danej klasy. Zasadę działania skryptu przedstawiono na schemacie (rys. 5.2), a różnice przy jego implementacji dla różnych klas, w kolejnych podrozdziałach.



Rys. 5.2 Schemat ogólny skryptu

5.2.3 Jednostki nazewnicze w NKJP

Niektóre z klas (czas, data) znajdują są w grupie jednostek nazewniczych (ang. *named entities*) reprezentowanych przez pliki *ann_named.xml* w każdym katalogu [10]. Pliki te zazwyczaj są niewielkich rozmiarów i pobór id odpowiedniej klasy zajmuje mniej czasu niż filtrowanie wszystkich słów na podstawie analizy danych morfologicznych. Liczebniki porządkowe nie są jednak uwzględnione w tej grupie i potrzebne będzie zastosowanie innych metod gromadzenia zdań do korpusu dla tej klasy – bez wykorzystania pliku *ann_named.xml*.

5.2.4 Podkorpus z oznaczeniami godzin

Jako pierwszy do implementacji wybrano skrypt gromadzący do korpusu zdania, w których wystąpiły godziny. Na początku zaimplementowano skrypt pomocniczy *examineTime.py* gromadzący wyłącznie formę ortograficzną wszystkich elementów opisanych jako czas w NKJP, w celu wstępnej analizy. Poniżej przedstawiono fragment pliku *ann_named.xml*, który zawiera informację o typie klasy (*type*) oraz jej formie ortograficznej (*orth*) :

```
▼<f name="type">
   <symbol value="time"/>
 </f>
▼<f name="orth">
   <string>godzinie 9.00</string>
 </f>
▼<f name="when">
   <string>09:00:00</string>
 </f>
▼<f name="certainty">
   <symbol value="high"/>
 </f>
```

Oprócz dwóch wymienionych wcześniej informacji równie istotną jest czas w zapisie uniwersalnym HH:MM:SS umieszczony poniżej tagu *"when"*. Okaże się to przydatne w dalszym procesie implementacji skryptu. Przeanalizowano wynik działania skryptu pomocniczego i okazało się, że postać formy ortograficznej czasu jest bardzo zróżnicowana. Pojawiły się zarówno godziny w formie tekstowej, zapisie liczbowym oraz zapisie mieszanym takie jak: '*godzinie dziewiątej*', '*9*', '*godz. 9.00*'. Fakt ten utrudnił implementacje skryptu, gdyż pojawił się problem z którego słowa pobierać formę gramatyczną oraz jaki dokładnie fragment zdania powinien pojawić się wewnątrz tagu. Stwierdzono że najlepszym rozwiązaniem będzie tagowanie całego wyrażenia

ortograficznego opisanego jako *orth* w NKJP, gdyż zminimalizuje to ilość błędów przy budowie korpusu, a nie skomplikuje w znaczącym stopniu modułu werbalizacji w systemie TTS. Jeśli chodzi o formę gramatyczną przy werbalizacji czasu (godzin) to istotna tylko jest forma liczby reprezentującej godzinę (minuty zawsze odczytuje się tak samo – jako liczby całkowite w formie podstawowej). W celu stworzenia jak największego korpusu ze zdaniami, postanowiono dokonać konwersji słów na liczby, w przypadku wyłącznie słownych form ortograficznych. Było to możliwe właśnie dzięki informacji o czasie w zapisie uniwersalnym. Zdecydowano się również na zapis tej informacji oraz pełnej formy gramatycznej wszystkich znalezionych klas – być może będą to cenne wiadomości przy rozwijaniu projektu TTS.

Główny skrypt modyfikowano kilkadziesiąt razy, testując go na coraz obszerniejszej części NKJP. Za każdym razem analizowano błędy w wynikowych zdaniach. Były to zarówno błędy związane z kodowaniem niektórych znaków (ostatecznie zdecydowano się na kodowanie utf-8) jak i błędną formą gramatyczną (na wskutek wyboru id z niewłaściwego elementu). W końcowej fazie implementacji udało się usunąć większość błędów, które generował skrypt. Zmodyfikowany skrypt znajduje się w pliku *getTimeSentences.py*.

5.2.5 Podkorpus z oznaczeniami dat

Podobnie jak w przypadku godzin, daty są umieszone w grupie jednostek nazewniczych, więc metoda tworzenia korpusu dat będzie zbliżona do metody ogólnej. Na początku również zaimplementowano skrypt pomocniczy *examineDate.py* oraz przeanalizowano jego wynik. Po wstępnej analizie form ortograficznych zaobserwowano, że ich postać jest jeszcze bardziej zróżnicowana niż w przypadku czasu. W skład elementów opisanych jako data w NKJP wchodzą: pełne daty, wieki, lata, miesiące, dni oraz ich dowolne kombinacje w postaci słów lub liczb. Zdecydowano pominąć wieki (liczby rzymskie to oddzielna klasa do rozpoznania) oraz izolowane miesiące (w zapisie używa się tylko formy słownej, więc ich zamiana na liczby nie ma sensu). Zakres tagowania przyjęto podobny jak w przypadku godzin, czyli zgodny z formą ortograficzną danej klasy według NKJP. Do typowych skrótów występujących w otoczeniu dat w zdaniach z NKJP można zaliczyć: *'r.','dn.','bm.'*. Do poprawnej werbalizacji tej klasy konieczna jest tylko znajomość przypadku, który jest zawsze ten sam dla dnia i roku w danym zdaniu. Z kolei w praktyce miesiące nie występują

w innym przypadku niż w dopełniaczu, co jest istotną informacją przy projektowaniu modułu werbalizacji. W skład oznaczeń dat w grupie jednostek nazewniczych często wchodziły dodatkowe słowa typu: *'roku', 'dnia', 'latach'*. Ich forma gramatyczna pokrywa się z formą daty, więc możliwa była ekstrakcja przypadku właśnie z tych słów. Końcowy skrypt do gromadzenia i tagowania zdań z datami znajduje się w pliku *getDateSentences.py*.

5.2.6 Podkorpus z oznaczeniami liczb porządkowych

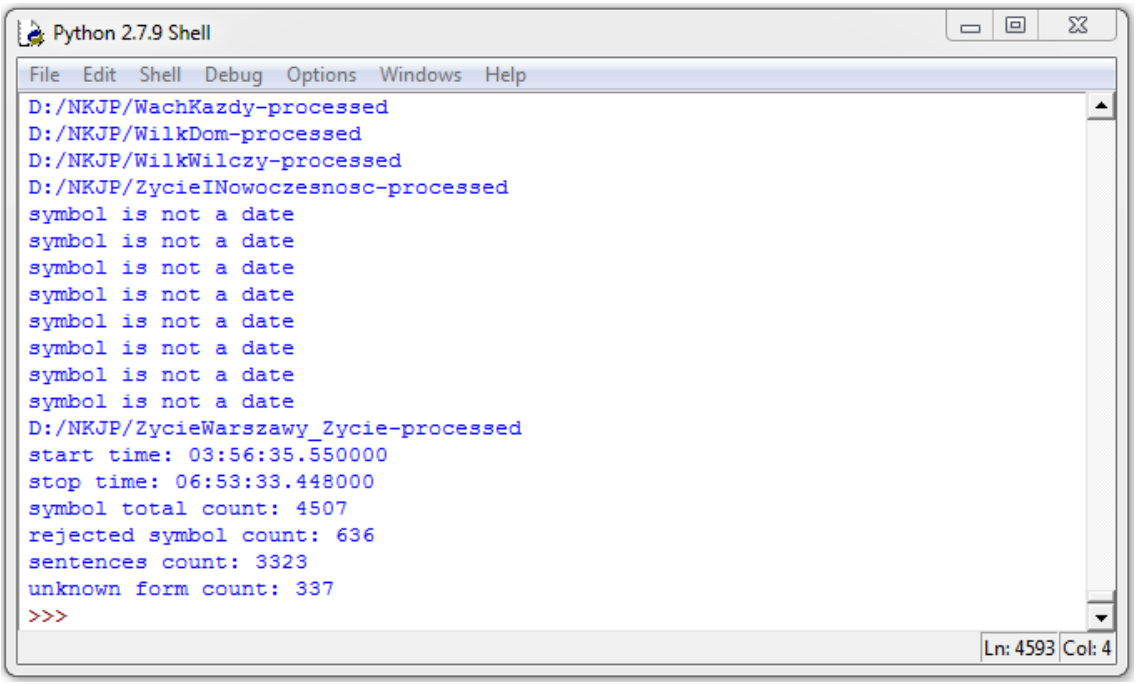
Liczby porządkowe nie są wprost oznaczone w żadnym z oznaczeń NKJP. Ich wydobycie wymaga zatem innego podejścia niż skorzystanie z jednostek nazewniczych, jak w przypadku dat lub godzin. Według konwencji NKJP liczby porządkowe są opisane jako przymiotniki *'Adj'*. Na początku z pomocą wyrażeń regularnych filtruje się plik *ann_words.xml* pobierając wszystkie formy ortograficzne w których występują liczby. Następnie sprawdza się część mowy badanych elementów – jeśli są one przymiotnikami to znaczy, że należą do liczb porządkowych i można pobrać z nich identyfikator (*id*) oraz formę gramatyczną. Dalsze działanie skryptu jest takie, jak przedstawiono w jego ogólnym schemacie (5.2.2). Istotnymi parametrami formy gramatycznej liczebników porządkowych ze względu na ich prawidłową werbalizacje są: przypadek oraz rodzaj. Z tego względu wewnątrz wszystkich tagów tej klasy zostały umieszczone atrybuty: *case* i *gen*.

W języku polskim po cyfrach arabskich oznaczających liczebniki porządkowe, według zasad pisowni i interpunkcji, powinno się stawiać kropki. Mogą być one jednak opuszczone, jeśli z kontekstu jednoznacznie wynika że użyty został liczebnik porządkowy. Z tej przyczyny klasyfikacja liczby porządkowej staje się problematyczna, gdyż w tekście może mieć postać identyczną jak liczba całkowita. W opisywanym skrypcie zaimplementowano prosty licznik liczebników porządkowych z kropką, w celu sprawdzenia jak często występują w języku pisanym. Gotowy skrypt nosi nazwę *getOrdinalSentences.py*.

5.3 Obsługa programu

Sposób użycia przygotowanych skryptów jest bardzo prosty. Użytkownik modyfikuje tylko ścieżkę katalogu w którym znajduje się NKJP (zmienna *nkjp_path*) oraz ścieżkę katalogu docelowego, w którym ma się znaleźć wynik skryptu (zmienna *output_path*). Do prawidłowego działania skryptu wymagany jest Python

z zainstalowaną biblioteką Beautiful Soup. Podczas działania program generuje proste komunikaty w interpreterze, czy dany katalog został pomyślnie przetworzony (*-processed*), czy brakuje w nim niezbędnych plików (*-unable to get sentences*). Na końcu generowane jest również krótkie podsumowanie zawierające czas rozpoczęcia i zakończenia działania skryptu (*start time*, *stop time*), liczbę symboli oznaczonych jako szukana klasa w NKJP (*symbol total count*), liczbę odrzuconych symboli (*rejected symbol count*), liczbę uzyskanych zdań (*sentences count*) oraz liczbę tagów, dla których nie odnaleziono formy gramatycznej (*unknown form count*). Poniżej dla przykładu przedstawiono okno programu po zakończeniu jego działania (rys. 5.3).



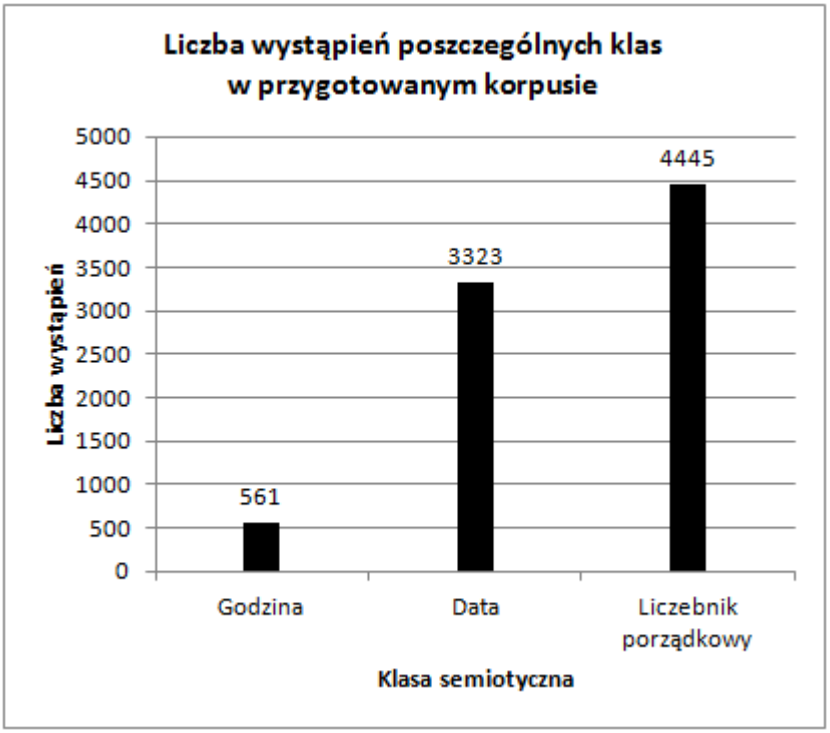
Rys. 5.3 Okno programu po zakończeniu działania skryptu getDateSentences.py

Wynikowe zdania składające się na korpus zapisywane są w plikach, o nazwie rozpoczynającej się tytułem podkatalogu NKJP, z którego zostały wydobyte. Przykładem nazwy takiego pliku może być: *Rzeczpospolita_timeSentences.txt*. Oprócz zdań program zapisuje w podobnej postaci pełny opis formy gramatycznej tagów (*..._form.txt*) oraz w przypadku godzin i dat - uniwersalny zapis czasu (*..._when.txt*). Skrypt generuje też plik zbiorczy wszystkich otrzymanych zdań. W każdym z wymienionych plików tekstowych nowe zdania oddzielane są nową linią.

6. Analiza przygotowanego korpusu

6.1 Analiza ogólna

W wyniku działania skryptu uzyskano 561 zdań z godzinami, 3323 zdania z datami oraz 4445 zdań z szeroko pojętymi liczbami porządkowymi. Razem daje to 8329 zdań treningowych w plikach tekstowych, które mają rozmiar 1,5 MB. Udział poszczególnych klas w przygotowanym korpusie językowym przedstawiono na wykresie (rys. 6.1).



Rys. 6.1 Wykres przedstawiający udział poszczególnych klas w korpusie

Korpus można umownie podzielić na trzy podkorpusy, tak jak zrobiono to w poprzednim rozdziale. Do podkorpusu z godzinami i datami dodatkowo zostały załączone pliki wynikowe skryptów pomocniczych, które stanowią zestawienie wszystkich form ortograficznych danej klasy w NKJP oraz ich czasów zapisanych w formacie standardowym. Noszą one nazwę ‘*class’_orths.txt* oraz ‘*class’_whens.txt* i są umieszczone w katalogu *preview* danej klasy.

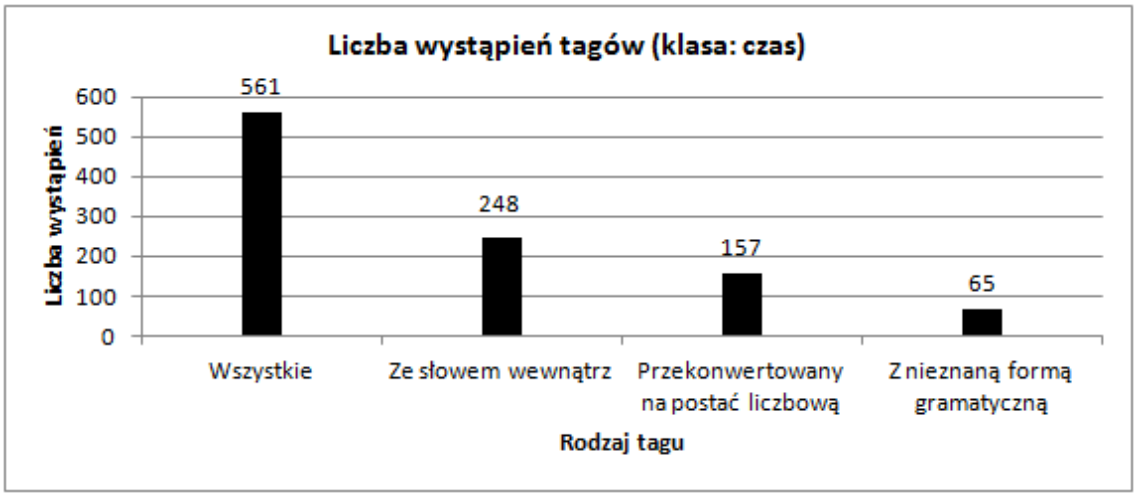
6.2 Analiza szczegółowa

6.2.1 Podkorpus z oznaczeniami czasu (część analityczna)

Automatyczne utworzenie podkorpusu z oznaczeniami czasu zajęło około 42 minut. Poniżej przedstawiono przykładowe zdania, które zostały wygenerowane:

„Najlepiej dziś o <time case="loc">18.30</time>.”
„Około <time case="gen">8.00</time> pojawili się pierwsi wyborcy.”
„Przyszedłem na świat 21 lutego o <time case="loc">godz. 14.20</time>.”

Jak widać na podstawie powyższych przykładów w korpusie występują tagi, w których oprócz godziny w postaci liczbowej znajdują się charakterystyczne słowa bądź skróty. Po analizie wyniku skryptu pomocniczego *time_orths.txt*, do najczęściej spotykanych można zaliczyć: {*‘wieczór’, ‘godzinie’, ‘godz.', ‘godziny’, ‘godzina’, ‘wpół do’, ‘rano’, ‘w nocy’*}. Mogą to być kluczowe słowa na które będzie zwracał uwagę wytrenowany tagger. Wiele z nich wskazuje także na prawidłową formę gramatyczną tagu (*‘godzinie’* - ósmej; *‘godzina’* - ósma), co jest kolejnym atutem takiej formy tagowania. W skrypcie umieszczono licznik form ortograficznych, w których występuje połączenie słowa z liczbą (*alnum symbols count*). Odnotowano 248 takich wystąpień. Zliczono również ilość konwersji wyłącznie słownych form ortograficznych na ich zapis liczbowy (*verbal to numeral transform count*). Wyniki zestawiono na poniższym wykresie (rys. 6.2).



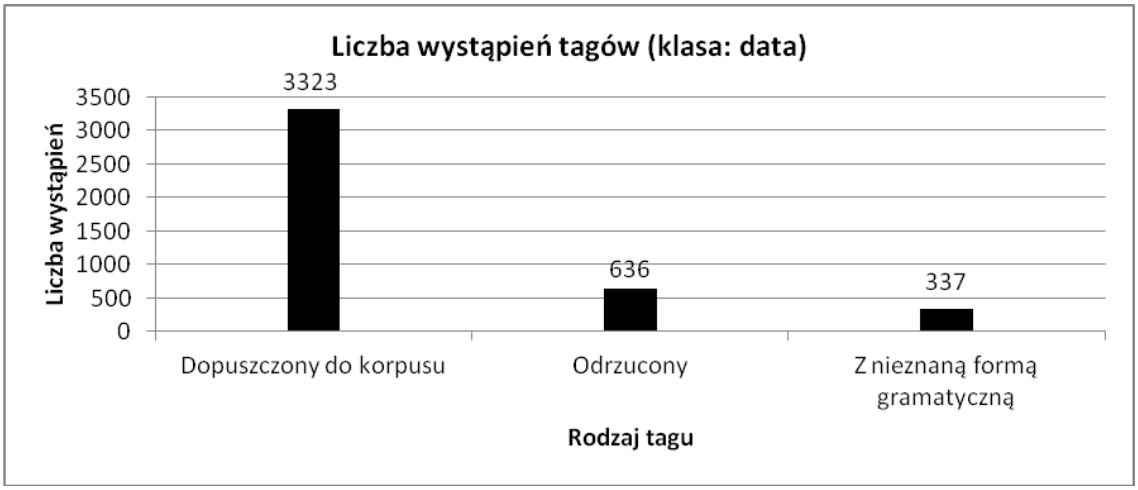
Rys. 6.2 Wykres przedstawiający ilość tagów w podkorpusie z oznaczeniami czasu

6.2.2 Podkorpus z oznaczeniami dat (część analityczna)

Automatyczne utworzenie podkorpusu z oznaczeniami dat zajęło około 3 godzin. Poniżej przedstawiono przykładowe zdania, które zostały wygenerowane:

„<date case="gen">15 lipca 2000 r.</date> kończy się sezon.”
„W <date case="loc">1938</date> Polacy stanowili już 70 % mieszkańców.”
„Zdarzenie miało miejsce <date case="gen">20 kwietnia</date> tego roku.”
„Obowiązuje od <date case="gen">1.01.1993 r.</date>”

Podobnie jak w przypadku podkorpusu z godzinami, przeanalizowano jakie charakterystyczne słowa mogą znaleźć się wewnątrz tagu. Można do nich zaliczyć: {*' roku' , ' r.', ' rok', 'latach ', ' r', 'dn. ', 'dniem ', 'dnia ', 'dniu'* , nazwy miesięcy}. Niektóre z odnalezionych symboli oznaczonych jako data w grupie jednostek nazewniczych (*ann_named.xml*) zostały odrzucone (głównie wieki oraz izolowane miesiące), było ich 636. Z kolei dla 337 zdań nie odnaleziono formy gramatycznej. Powyższe wyniki zestawiono na wykresie (rys. 6.3).



Rys. 6.3 Wykres przedstawiający ilość tagów w podkorpusie z datami

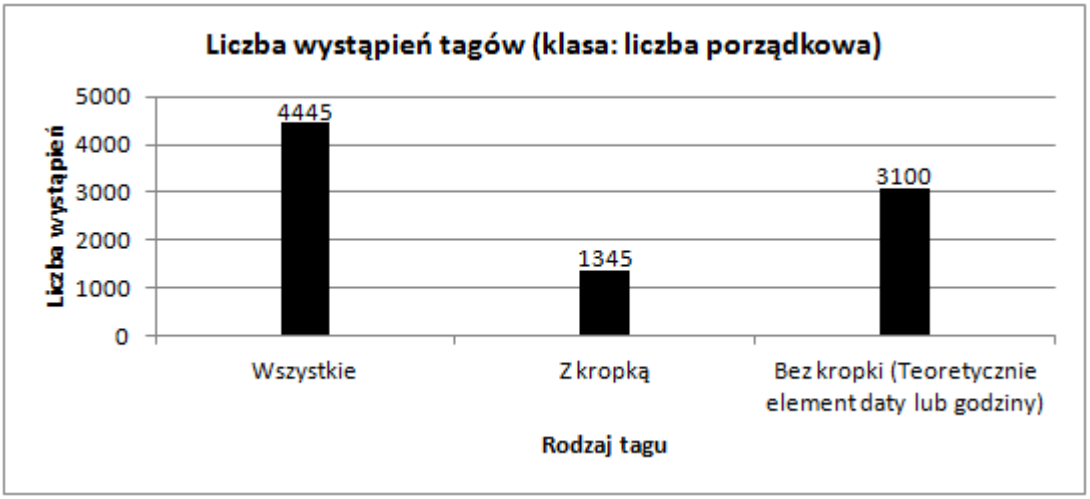
6.2.3 Podkorpus z oznaczeniami liczb porządkowych (część analityczna)

Automatyczne utworzenie podkorpusu z oznaczeniami liczb porządkowych zajęło około 1 godziny, co wydaje się być sprzeczne z jego rozmiarem, gdyż utworzenie mniej licznego podkorpusu dat trwało około 3 razy dłużej. Może to być uwarunkowane korzystaniem z mniejszej liczby plików podczas działania skryptu (pominięcie pliku

ann_named.xml). Po wstępnej analizie tekstowego pliku wyjściowego, zauważono że znalazło się w nim wiele klas o różnej złożoności, w tym godziny, dni, lata, czyli tagi zawarte również w innych przygotowanych podkorpusach. Poniżej przedstawiono przykładowe zdania, które zostały wygenerowane:

„Wróciło do łask w <ordinal case="loc" gen="m3">1960</ordinal> r.”
„<ordinal case="nom" gen="n">1.</ordinal> Perfekcjonizm to twoja dewiza.”
„mowa w art. <ordinal case="loc" gen="m3">4</ordinal> ust. 1 pkt 6”

Na podstawie przykładowych zdań można stwierdzić, że klasa liczb porządkowych jest klasą bardzo ogólną, w skład której wchodzą również numery artykułów, punktów. Jednak z obserwacji zdań wyjściowych, największą grupę stanowią zdania z otagowanymi dniami oraz latami. Nie świadczy to, że podkorpus z liczbami porządkowymi jest bezużyteczny. Może on posłużyć do eksperymentu sprawdzającego na jakich danych efektywniej można trenować automatyczny tagger – na zdaniach z otagowanymi klasami złożonymi (pełne godziny lub daty), czy podstawowymi (liczby porządkowe). Dodatkowo obliczono ilość tagów z liczbami porządkowymi, w których wystąpiła kropka, czyli teoretycznie ich poprawną pisownią. Zakładając, że w datach i godzinach nie jest ona wymagana, można oszacować liczbę zdań z tymi klasami, odejmując od liczby wszystkich zdań, liczbę tagów z kropką. Obliczono, że może być około 3100 takich zdań, co w pewnym stopniu odpowiada ich liczbie w innych podkorpusach. Wyniki przedstawiono na poniższym wykresie (rys. 6.4).



Rys. 6.4 Wykres przedstawiający ilość tagów w podkorpusie z liczbami porządkowymi

7. Podsumowanie

Podsumowując wszystkie działania przy realizacji niniejszej pracy, można stwierdzić, że osiągnięto założone cele. Przeprowadzono analizę potrzeb normalizacji tekstu i wyciągnięto z niej wnioski. Z analizy wynika dla których klas użycie metod uczenia maszynowego może być efektywne przy ich klasyfikacji, a dla których podejście to nie przyniesie pożądanych efektów. Zwrócono również uwagę, że ogromny wpływ na skuteczność klasyfikacji ma odpowiednie zaprojektowanie systemu pod określony cel. To on determinuje ostateczną postać algorytmów w module normalizacji. Szczególnym przypadkiem są systemy IVR, w których dobrze zaprojektowany przebieg specyficznych dialogów jest kluczowy i może w zupełności usunąć potrzebę użycia złożonych metod normalizacji tekstu.

Udało się również przygotować korpus językowy, czyli tytułowy cel projektu inżynierskiego. Wiąże się z nim również sporządzenie narzędzi programowych niezbędnych do jego realizacji. Przygotowane skrypty stanowią bazę do efektywnego poruszania się po strukturze Narodowego Korpusu Języka Polskiego i mogą być wykorzystywane do innych celów związanych z pozyskiwaniem informacji lingwistycznych z tego korpusu.

Przy analizie zdań będących wynikiem skryptów, zauważono liczne zależności między klasami, a słowami w ich otoczeniu, co może być wartościowe przy projektowaniu automatycznych klasyfikatorów. Jeśli chodzi o główny produkt, czyli przygotowany korpus językowy, jest on podstawą do dalszych badań w zakresie uczenia maszynowego na potrzeby normalizacji tekstu. Projekt ten będzie rozwijany w przyszłości na potrzeby syntezatora mowy firmy Techmo [11].

Bibliografia

[1] Black A., Lenzo K.: *Buliding Synthetic Voices*, 1999-2014. Dostępny: http://festvox.org/bsv/bsv.pdf.

[2] Taylor P.: *Text-to-Speech Synthesis*, Cambridge, UK, New York, Cambridge University Press, 2009.

[3] Wikipedia: *Interactive Voice Response*. Dostępny: https://pl.wikipedia.org/wiki/Interactive_Voice_Response

[4] Słownik Języka Polskiego: *homograf*. Dostępny: http://sjp.pl/homograf

[5] Woliński M.: *System znaczników morfosyntaktycznych w korpusie IPI PAN.* „Polonica", 2003. Dostępny: http://nlp.ipipan.waw.pl/CORPUS/znakowanie.pdf

[6] Woliński M., Miłkowski M., Ogrodniczuk M., Przepiórkowski A.: *PoliMorf: a (not so) new open morphological dictionary for Polish*, Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12), European Language Resources Association (ELRA), 2012. Dostępny: http://www.lrec-conf.org/proceedings/lrec2012/pdf/263_Paper.pdf

[7] Przepiórkowski A., Bańko M, Górski R., Lewandowska-Tomaszczyk B.: *Narodowy Korpus Języka Polskiego.* Wydawnictwo Naukowe PWN, Warsaw, 2012. Dostępny: http://nkjp.pl/settings/papers/NKJP_ksiazka.pdf

[8] Bird S., Klein E., Loper E.: *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, 2009

[9] Richardson L.: *Beautiful Soup Documentation*, 2015. Dostępny: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[10] Savary A., Waszczuk J., Przepiórkowski A.: *Towards the Annotation of Named Entities in the National Corpus of Polish*. ,LREC 2010 proceedings Dostępny: http://www.lrec-conf.org/proceedings/lrec2010/pdf/879_Paper.pdf

[11] Demo syntezatora mowy Techmo: http://techmo.pl/index.php/synteza/demo