UNIWERSYTET IM. ADAMA MICKIEWICZA
WYDZIAŁ MATEMATYKI I INFORMATYKI

Marta Wieczorek
nr albumu: 316906

Lingwistyczne aspekty syntezy mowy z tekstu

Linguistic aspects of text-to-speech synthesis

Praca magisterska na kierunku:
INFORMATYKA
Promotor:
prof. UAM dr hab. Krzysztof Jassem

Poznań 2011

Spis treści

Oświadczenie

Poznań, dnia . . . . . .

Ja, niżej podpisana **Marta Wieczorek** studentka Wydziału Matematyki i Informatyki Uniwersytetu im. Adama Mickiewicza w Poznaniu oświadczam, że przedkładaną pracę dyplomową pt:

Lingwistyczne aspekty syntezy mowy z tekstu

napisałam samodzielnie. Oznacza to, że przy pisaniu pracy, poza niezbędnymi konsultacjami, nie korzystałam z pomocy innych osób, a w szczególności nie zlecałam opracowania rozprawy lub jej części innym osobom, ani nie odpisywałam tej rozprawy lub jej części od innych osób.

Oświadczam również, że egzemplarz pracy dyplomowej w formie wydruku komputerowego jest zgodny z egzemplarzem pracy dyplomowej w formie elektronicznej.

Jednocześnie przyjmuję do wiadomości, że gdyby powyższe oświadczenie okazało się nieprawdziwe, decyzja o wydaniu mi dyplomu zostanie cofnięta.

.....

Wprowadzenie

Synteza mowy to proces polegający na automatycznym przetwarzaniu tekstu z postaci pisanej do mówionej. Składa się na nią wytwarzanie fonetycznego zapisu czytanego tekstu, ustalenie intonacji, rytmu i innych elementów prozodii oraz generowanie mowy na podstawie danych zebranych we wcześniejszych krokach przetwarzania.

Niniejsza praca magisterska dotyczy lingwistycznej części procesu syntezy mowy, czyli normalizacji. Normalizacja polega na przetworzeniu dowolnego tektu do postaci z rozwiniętymi wyrażeniami niestandardowymi. Mianem niestandardowych opisywać będziemy wszystkie wyrażenia, które zapisujemy inaczej niż czytamy, np. skróty wymagające rozwinięcia, skrótowce i ciągi nieleksykalne wymagające przeliterowania, liczby arabskie i liczby rzymskie wymagające zastąpienia ich postacią słowną, itd.

W pracy przedstawione jest zagadnienie normalizacji oraz omówione są występujące w różnych normalizatorach błędy normalizacji tekstów polskich. Przetestowanie pod względem występowania tych błędów normalizatora używanego w systemie Polfest pozwoliło na określenie kierunku prowadzenia prac implementacyjnych będących częścią projektową pracy. Celem części projektowej jest podniesienie jakości działania normalizatora systemu Polfest poprzez eliminację wykrytych błędów.

W Rozdziale 1. omówione są podstawowe zagadnienia związane z syntezą mowy i normalizacją tekstu. W Rozdziale 2. przedstawione są wyniki przeprowadzonego eksperymentu polegającego na zebraniu i porównaniu jakości normalizowanych próbek dla trzech najbardziej popularnych w Polsce systemów syntezy mowy. Rozdział 3. zawiera opis błędów rozpoznanych podczas testowania normalizatora używanego w systemie Polfest oraz sposób ich rozwiązania.

Dodatek A do pracy przedstawia, w jakim stopniu poprawiła się jakość normalizacji po wprowadzeniu nowych i poprawieniu istniejących reguł.

ROZDZIAŁ 1

Podstawy teoretyczne

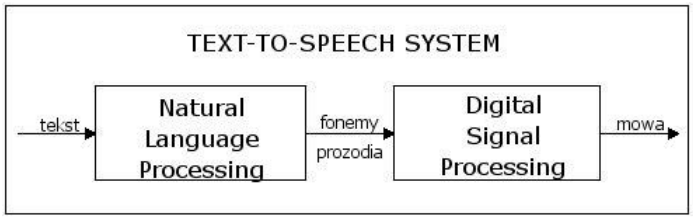
1.1. Synteza mowy

Definicja 1.1 ***Synteza mowy****, sztuczne wytwarzanie, pierwotnie (XIX w.) za pomocą przyrządów mechanicznych, oddzielnych głosek, wyrazów itp.; na przełomie XX i XXI w. mowa wytwarzana za pomocą komputera i zbliżona brzmieniowo do naturalnej; synteza mowy jest szczególnie użyteczna w czytaniu tekstów drukowanych (ang. text-to-speech); stosuje się ją (również w Polsce) w komputerach przystosowanych do obsługi przez niewidomych.*[1]

1.1.1. Działanie

Proces czytania odbywa się przez analizę i obróbkę otrzymanego na wejściu tekstu, a nie przez odtwarzanie wcześniej przygotowanych nagrań. Wymaganiem stawianym przed tekstem wejściowym jest zapis w postaci elektronicznej, w języku naturalnym obsługiwanym przez syntezator.

Spodziewamy się, że syntezator mowy podczas przetwarzania tekstu z postaci pisanej do mówionej będzie wykonywał zabiegi podobne do tych, które podczas czytania wykonuje człowiek. Tak też się dzieje, jednak w systemie TTS (Text-To-Speech) rozgraniczone są poszczególne etapy przetwarzania czytanego teksu. Pierwszy etap to Natural Language Processing (NLP), w którym wytwarzany jest fonetyczny opis czytanego tekstu, a ponadto ustalana jest intonacja i rytm (elementy prozodii). Kolejny etap to Digital Signal Processing (DSP) – na podstawie danych wyjściowych z modułu NLP generowana jest mowa. Proces ten zobrazowany jest na Rysunku 1.1.



Rysunek 1.1. TTS system

[1]Definicja pochodzi z [1]

Pierwsza faza Natural Language Processing to analiza tekstu. Rozpoczyna się od preprocesingu, w trakcie którego tworzona jest lista słów z wyrażenia wejściowego. Rozpoznawane są liczby, skróty, akronimy i tworzone ich tekstowe odpowiedniki. Przykładem jest zdanie: „Wykład prof. Jana Miodka rozpocznie się za 10 min.", które po rozwinięciu skrótów i liczb przybierze postać: „Wykład [profesor] Jana Miodka rozpocznie się za [dziesięć] [minuta/min].". Na tym poziomie nie posiadamy jeszcze informacji o fleksji, a wstawiane w miejsce skrótów odpowiedniki mają formę podstawową (mianownik, liczba pojedyncza). Kolejną fazą jest analiza morfologiczna – określenie zbioru możliwych części mowy dla każdego wyrazu. Dla powyższego przykładu otrzymujemy: wykład – rzeczownik, profesor – rzeczownik, Jan – rzeczownik, Miodek – rzeczownik, rozpocznie się – czasownik, za – przyimek, dziesięć – liczebnik, minuta/min – rzeczownik. W przypadku, gdy nie można jednoznacznie określić do jakiej części mowy należy rozpatrywany wyraz/wyrażenie, wówczas rozpoznawana jest ona na podstawie kontekstu, w którym wyraz wystąpił (analiza semantyczna). Analiza semantyczna umożliwia również wybór bardziej odpowiedniego rozwinięcia skrótu w przypadku niejednoznaczności. Na przykład w powyższym zdaniu pozwoliłaby na przyjęcie założenia, że „min" jest skrótem wyrazu minuta, w przeciwieństwie do „min" będącego jedną z form wyrazu mina (mowa o dopełniaczu liczby mnogiej). Następnie zdanie jest poddane analizie syntaktycznej– rozpoznaniu struktury zdania. Dzięki niej możliwe jest ustalenie poprawnych form gramatycznych skrótów i liczebników (tutaj: wygenerowanie form: „profesora" (dopełniacz rzeczownika „profesor"), dziesięć (biernik liczebnika „dziesięć") i „minut" (dopełniacz l. mnogiej od wyrazu „minuta").

Po analizie tekstu następuje proces automatycznej fonetyzacji. Podstawą tego etapu jest odszukanie w słowniku fonetycznym morfemu dla każdego wyrazu z tekstu, np. dla hasła „trzcina" zostanie znaleziony odpowiednik „czcina". Mogłoby się wydawać, że nie jest to szczególnie złożony etap. Nic bardziej mylnego. Przy dokładniejszej analizie napotykamy następujące trudności:

- w słowniku znajdują się jedynie podstawowe formy haseł,
- tworzenie wymowy dla odmienionego hasła ze słownika nierzadko polega na zmianie wymowy jego podstawy słowotwórczej, np. hasło „chleb" w mianowniku liczby pojedynczej czytamy „chlep", podczas gdy ten sam wyraz w mianowniku l. mn. „chleby" czytamy „chleby",
- większość słów posiada więcej niż jeden odpowiednik fonetyczny, o czym słownik nie informuje, np. przyimek „w" - we frazie „lew w klatce" czytamy „lef f klatce", natomiast „lew w zoo" – „lef w zoo". Należy zwrócić uwagę, że na fleksję wpływ ma kontekst, w którym wystąpił wyraz.
- nie wszystkie wyrazy znajdują się w słowniku.

W związku z napotykanymi trudnościami słownik uzupełniony jest o zbiór reguł, które utworzą odpowiednią wymowę dla nieznalezionych haseł.

Ostatnią częścią przetwarzania teksu jest generowanie prozodii. Na podstawie wyników analiz (syntaktycznej i semantycznej) ustalana jest wartość takich parametrów jak: wysokość tonu, ekspresja, głośność, umiejscowienie akcentu w słowie, akcentu w zdaniu oraz grupowanie wyrazów celem ustalenia odpowiedniego rytmu wymawianego zdania.

Jest to niezwykle trudne, gdyż tekst nie wyraża emocji, których wymagamy przy czytaniu. Nie ma zaznaczonych miejsc, gdzie należy podnieść głos, a gdzie obniżyć. Jednak bez tego zabiegu otrzymalibyśmy wynik porównywalny do przeczytania tekstu przez ucznia drugiej klasy podstawowej, tzn. wszystkie wyrazy przeczytane poprawnie, ale w sposób monotonny, bez zaznaczenia akcentów i emocji, które towarzyszą wyrażaniu myśli.

Do opisu prozodii możemy posłużyć się językiem Speech Synthesis Markup Language (SSML)[7]. Dzięki niemu w łatwy sposób zapisywane są wartości wszystkich parametrów prozodycznych. Przykładowy plik napisany w języku SSML znajduje się na Listingu 1.

```
<?xml version="1.0"?>
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
 http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
 xml:lang="en-US">
Lekcje tańca są~<emphasis> bardzo </emphasis> drogie!
<break time="3s"/>
Jednak są dla Mateusza <emphasis level="strong"> niezwykle </emphasis> ważne!
<break strength="weak"/>
Jedna kosztuje <prosody rate="-10\%"> 100~</prosody> zł.
</speak>
```

Listing 1 Opis prozodii w języku SSML

W pliku przedstawionym na Listingu 1 zawarte są informacje o prozodii dla następujących zdań:

– Lekcje tańca są bardzo drogie!

– Jednak są dla Mateusza niezwykle ważne!

– Jedna kosztuje 100 zł.

Aby wzbogacić tekst o emocje towarzyszące wypowiadaniu powyższych zdań zastosowano następujące znaczniki:

– `emphasis` – otagowany tekst będzie wymawiany z pewnym naciskiem (podkreśleniem). W zależności od oczekiwanego efektu intonacji, atrybutowi `level` należy nadać jedną z następujących wartości: `strong`, `moderate`, `none`, `reduced`.

  Użyta w przykładzie wartość `strong` powoduje, że podkreślenie zostanie wytworzone poprzez zmianę wysokości tonu (podniesienie), zmianę czasu trwania wypowiedzi (spowolnienie), zmianę głośności (podgłoszenie), a także inne zmiany akustyczne.

– `break` – znacznik ustawiany w miejscu, w którym należy wstawić pauzę krótszą/dłuższą niż pauza domyślnie wstawiana między dwoma wyrazami czy też wyrazem a znakiem interpunkcyjnym. Opcjonalnymi atrybutami znacznika `break` są `strength` i `time`.

  Zapisany w przykładzie `break` opisany jest atrybutem `time` o wartości 3s, co oznacza, że domyślna pauza zostanie zastąpiona pauzą o długości trwania 3 sekund.

– `prosody` – znacznik opisuje wysokość tonu, tempo i głośność czytania. Opcjonalnymi atrybutami są: `pitch`, `contour`, `range`, `rate`, `duration`, `volume`.

  W podanym przykładzie atrybutowi `rate` ustawiona została wartość -10%, co oznacza, że tempo otagowanego tekstu zmaleje o 10% w stosunku do tempa czytania otaczającego, ów znacznik, tekstu.

Kolejnym etapem syntezy mowy jest Digital Signal Processing – proces generowania mowy w oparciu o dane zebrane podczas przetwarzania tekstu. Istnieją dwa różne podejścia do rozwiązania tego problemu. Pierwsze podejście oparte jest o zbiór reguł – fale akustyczne odpowiadające wymowie liter wytwarzane są na bieżąco, w trakcie DSP – podejście takie nazywa się syntezą opartą o reguły. Inne podejście mówi o wcześniejszym nagraniu fonemów i łączeniu podczas generowania mowy – nazywamy syntezą konkatenacyjną.

1.1.2. Użytkownicy

Użytkowników systemów syntezy mowy możemy podzielić według różnych kryteriów. Pierwszym jest poziom wiedzy użytkownika o stopniu, w jakim program czy urządzenie korzysta z syntezy mowy. Możemy wyróżnić dwie grupy użytkowników: bezpośredni i pośredni. Użytkownicy bezpośredni nastawieni są na przetwarzanie słowa pisanego, tekstu w formie elektronicznej, na słowo mówione. Przykładem są uczniowie, którzy systemów syntezy mowy używają do czytania lektur szkolnych. Natomiast użytkownikami pośrednimi są osoby często nie wiedzące, że częścią używanego przez nie sprzętu czy oprogramowania jest syntezator mowy. Dobrym przykładem są użytkownicy nawigacji samochodowej, w której głos męski/damski czyta komunikaty wyświetlane na wyświetlaczu urządzenia GPS.

Innym, bardziej naturalnym kryterium podziału użytkowników jest ich charakterystyka według potrzeb i celów, jakim służy używanie syntezy. Możemy wyróżnić trzy grupy użytkowników: klienci sektora usług, osoby niepełnosprawne i inne systemy informatyczne. Klienci sektora usług to największa, najbardziej zróżnicowana grupa użytkowników. Synteza mowy służy im do podwyższenia standardu życia. Moduł syntezy nie jest dla nich niezbędny, jednak znacznie podwyższa jakość zakupionych usług, a także zwiększa zadowolenie z używanych urządzeń. Przykładem są klienci komunikacji miejskiej, którzy otrzymują podczas podróży komunikaty informujące o nazwie następnego przystanku tramwajowego czy autobusowego. Kolejną grupą użytkowników są osoby niepełnosprawne, z niepełnosprawnością wzroku i mowy. Dla nich syntezatory mowy są narzędziem niezbędnym do życia w społeczeństwie. Dzięki „mówiącemu komputerowi" (a dokładniej czytającemu) mogą zdobywać wiedzę czytając treść stron internetowych, książek elektronicznych, czy też komunikować się z innymi ludźmi bez konieczności posługiwania się przez obydwie rozmawiające osoby językiem migowym. Syntezator wyraźnie przeczyta przekazaną mu treść. Ostatnia z wymienionych grup, to inne systemy informatyczne. Bardzo dobrym przykładem są chatboty (chatterboty, linguaboty) - programy komputerowe, których zadaniem jest prowadzenie na wybrany temat rozmowy z człowiekiem. Rozmowa ma być naturalna, ma również symulować dialog między ludźmi, dlatego poza opracowaniem dobrej jakości przekazywanej treści, podłącza się syntezator mowy, który tą treść czyta.

1.1.3. Zastosowanie

Synteza mowy ludzkiej, wraz z rozwojem technologii, znajduje coraz więcej zastosowań. Systemy i dziedziny, w których jest rozpowszechniona to:

- **Serwis informacyjny, serwer foniczny** – informowanie klientów sklepów o promocjach, podróżnych na stacjach kolejowych o zmianach w rozkładzie jazdy pociągów, podróżnych w tramwajach o nazwie następnego przystanku, użytkowników wind o numerze piętra, klientów kin o najbliższym repertuarze.

- **Nauka, rozrywka** – czytanie książek (szczególnie lektur, koniecznie zapisanych w formie elektronicznej) dla uczniów szkół podstawowych, gimnazjów i szkół średnich, a także odsłuchiwanie treści książek w trudnych warunkach – przy ograniczonym dostępie do światła, na urządzeniach nie przystosowanych do wyświetlania dużej ilości tekstu.

- **Komunikacja człowiek-komputer** – rozmowa z linguabotem, programem dostępnym na stronie internetowej przedsiębiorstwa, w celu przekazania informacji o firmie, np. Wojtekbot na stronie firmy Poleng[2] czy też roBOT Janek na stronie Wirtualnej

[2]http://poleng.pl/awatar/

Polski[3], promujący korzystanie z usługi poczty elektronicznej dostępnej na tym portalu.

Komunikacja człowiek-komputer to także proponowany przez banki bankofon, automatyczny system umożliwiający dostęp do informacji o placówkach i aktualnej ofercie banku, stanie konta, oprocentowaniu kredytów, umożliwiający realizację przelewów.

Ponadto do dziedziny tej zaliczymy przekazywanie komunikatów użytkownikowi o stanie systemu, np. komunikaty oprogramowania antywirusowego o dokonaniu aktualizacji bazy danych.

– **Help Desk, Call Centar, serwery IRV** – zastąpienie pracowników przez elektronicznych konsultantów. Taki konsultant to połączenie modułu TTS (Text-To-Speech) i ASR (Automatic Speech Recorgnition) z aplikacjami VoiceXml. W ten sposób wsparcie techniczne może być dostępne w wielu językach o każdej porze dnia i nocy, bez konieczności zatrudnienia pracowników ze znajomością wielu języków (co znacznie zmniejsza koszty obsługi klienta).

  Z drugiej strony, w przypadku przechowywania gotowych nagrań w bazie danych, rozwiązanie to pozwala na uniknięcie zbędnego zajmowania pamięci i oczekiwania na odszukanie, gdy wiadomość należy odtworzyć. W zamian za to wiadomości są czytane na bieżąco, a w przypadku zmiany treści komunikatu, nie trzeba aktualizować nagrań we wszystkich językach.

– **Telekomunikacja** – odsłuchanie wiadomości elektronicznej, np. wiadomości e-mail.

– **Słownik** – możliwość odsłuchania prawidłowej wymowy hasła dostępnego w słowniku elektronicznym.

– **Nauka, rozrywka** – odsłuchanie treści e-booków, zawartości stron internetowych, poczty elektronicznej oraz innych dokumentów elektronicznych przez osoby niewidzące i niedowidzące.

– **Komunikacja z innymi ludźmi** – wyrażanie własnego zdania przez osoby nie mówiące i z poważnymi wadami wymowy. Rozwiązaniem jest zaawansowany pakiet sprzętowo – programowy. Pakiet taki powinien zawierać bogaty w hasła słownik oraz system uczący się najczęściej używanych zwrotów (na podstawie wcześniejszych rozmów użytkownika), podłączony do syntezatora mowy. Z opisanego rozwiązania korzysta naukowiec Stephen Hawking.[12]

Lista dziedzin życia, w których synteza mowy znajduje zastosowanie jest długa, a wraz z rozwojem technologii staje się coraz obszerniejsza. Wiele z zastosowań czeka na dalszy roz-

[3]http://poczta.wp.pl/bot2.html

wój syntezy mowy (sukces mogą osiągnąć tylko przy wysokiej jakości dźwięku generowanego przez syntezatory mowy, które najlepsze lata mają dopiero przed sobą). Oto przykłady:

- **Nauka języka obcego** – powiązanie wysokiej jakości systemu TTS z narzędziem do nauki języka obcego to idealne rozwiązanie do nauki zarazem języka pisanego, jak i „mówionego".
- **Telekomunikacja** – usługa „kto dzwoni". W chwili obecnej dostępna w formie pisanej – na wyświetlaczu telefonu można zobaczyć numer telefonu, bądź nazwę osoby dzwoniącej, gdy zapisana jest w książce telefonicznej. Opisane rozwiązanie można rozwinąć przez przeczytanie wyświetlanej treści.
- **Komunikator internetowy** – czytanie wiadomości przesyłanych przez komunikatory internetowe, gdy głos rozmówcy jest nieprzyjemny lub nie posiadamy mikrofonu.
- **Komunikacja człowiek-komputer** – wydawanie poleceń komputerowi, czytanie przez komputer komunikatów do użytkownika.

1.2. Normalizacja tekstu

Definicja 1.2 ***Normalizacja*** *[łac. normalis 'uregulowany'], standaryzacja, opracowywanie i wprowadzanie w życie norm; ma na celu m.in. zapewnienie funkcjonalności wyrobom i usługom, usuwanie barier w handlu, ułatwianie współpracy nauki i techniki; w Polsce normalizacją zajmuje się Polski Komitet Normalizacyjny.*[4]

1.2.1. Metody normalizacji tekstu

Normalizacja tekstu to automatyczne odwrócenie tekstu z postaci ortograficznej na mówioną. Jest to złożone zadanie, gdyż wymaga wykonania wieloetapowej analizy tekstu pod względem:

- **morfologicznym** (fleksji, słowotwórstwa)
- **syntaktycznym** (składni, funkcji wyrazu w zdaniu)
- **semantycznym** (znaczenia wyrazów)

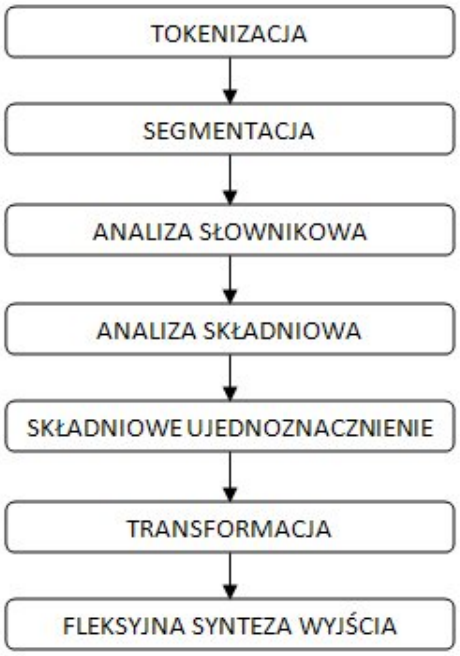
przez komputer – urządzenie, które nie posiada zdolności rozumienia treści, a tym bardziej zapamiętywania i uczenia się struktur występujących w zdaniach określonego języka. W zamian za to komputer można wyposażyć w zestaw reguł opisujących język oraz bogaty słownik używany do tłumaczenia skrótów, akronimów i innych wyrażeń niestandardowych.

[4] Definicja pochodzi z [2]

Istnieje kilka podejść do rozwiązania problemu normalizacji tekstu. Pierwsze traktuje ją jako specyficzną formę tłumaczenia automatycznego, w którym język źródłowy jest jednocześnie docelowym – rozwiązanie wykorzystywane w polskim syntezatorze Polfest. Kolejne używa modułu opartego o WFST (Weighted Finite State Transducer) [15], mapującego różne poziomy opisu wyrazów (ortograficzne wejście, analizę leksykalną, opis fonetyczny) na jeden zapis – mowa o rozwiązaniu użytym w systemach firmy Bell Labs.

Przyjrzyjmy się rozwiązaniu podchodzącemu do normalizacji jak do tłumaczenia automatycznego.

Przed przystąpieniem do pracy należy przygotować bogaty w hasła słownik, w którym język źródłowy jest zarazem językiem docelowym, np. słownik polsko – polski dla normalizatora polskiego. Poza ortograficznym zapisem hasła powinny się w nim znaleźć informacje o jego funkcji w zdaniu, kontekście w jakim może wystąpić (jakiej części zdania można się spodziewać przed, a jakiej po haśle) oraz „tłumaczenie" hasła, np. dla „płk" tłumaczeniem jest „pułkownik". Będąc zaopatrzonym w tak przygotowany słownik można przystąpić do normalizacji, której kolejne etapy zobrazowane są na Rysunku 1.2



Rysunek 1.2. Architektura normalizatora

Przedstawiony na schemacie proces normalizacji rozpoczyna się od tokenizacji. Jak sama nazwa wskazuje, tekst zostaje podzielony na tokeny. Należy rozpoznać gdzie jest początek, a gdzie koniec każdego wyrażenia. Ustalane jest znaczenie białych znaków w napisach numerycznych, a także znaków interpunkcyjnych, szczególnie kropek, w napisach alfanumerycznych. Trzeba m.in. zwrócić uwagę na fakt, że spacje w liczbie służą zwykle zwiększeniu czytelności, np. liczba „1 300" powinna być traktowana jako jeden token, a nie jako dwa – „1" i „300". Kropka natomiast może być użyta jako:

- zakończenie zdania, np. „Nic nie powiedzieć, to prawie jak kłamać.",
- część skrótu nie występującego na końcu zdania, np. „Książka pt. 'Jeśli zimową nocą podróżny'.",
- część skrótu występującego w środku zdania lub na końcu, np. „Adam Małysz zakończy karierę sportową w marcu 2011 r.", „W marcu 2011 r. karierę sportową zakończy Adam Małysz."
- część skrótu zwykle występującego na końcu zdania, np. „O czarownicach, czarnoksiężnikach, magach, itp.",
- część napisu nie będącego ani skrótem, ani końcem zdania, np. www.wp.pl.

W procesie tokenizacji ustalamy, jak ma być traktowany znak kropki – jako oddzielny token czy część napisu.

Poczynione ustalenia są zapisywane i przekazywane do kolejnego etapu, którym jest segmentacja. Tutaj obróbka tekstu odbywa się na poziomie tokenów. Szukane są wyrażenia, które nie są tylko numeryczne lub tylko alfabetyczne. Dla odnalezionych w ten sposób napisów system podejmuje decyzję: dzielić dalej czy pozostawić bez zmian. Zobaczmy przykłady:

- „25-lecie" – decyzja: niepodzielny token, pozostawić bez zmian,
- „3-4" – decyzja: podzielić na tokeny: „3", „-", „4",
- „1:0" – decyzja: podzielić na tokeny: „1", „:", „0",
- „88-300" – decyzja: niepodzielny token (kod pocztowy), pozostawić bez zmian,
- „7.10.1989" – decyzja: niepodzielny token (data), pozostawić bez zmian.

Trzecim etapem jest analiza słownikowa. Pełni ona dwie funkcje. Po pierwsze, dla każdego tokenu próbuje odnaleźć hasło w słowniku i pobrać zapisane o nim informacje, przede wszystkim opis semantyczny i syntaktyczny. W przypadku, gdy poszukiwane wyrażenie nie zostanie

znalezione – wywołuje odpowiednie procedury, których zadaniem jest ustalenie brakującego opisu. Po drugie, znajduje tłumaczenie hasła będącego skrótem.

Po analizie słownikowej następuje analiza syntaktyczna, w której wyrazy łączone są kolejno we frazy i drzewa składniowe. Zabieg ten ma umożliwić nadanie wszystkim niestandardowym wyrażeniom z normalizowanego tekstu odmiany wynikającej bezpośrednio z kontekstu, w którym wystąpiły. Dla otrzymanych hipotez składniowych nadane zostają oceny, po czym ta najwyżej punktowana zostaje wybrana jako poprawne drzewo składniowe tłumaczonego zdania.

Podsumowując dotychczasowe wyniki normalizacji: podzielony na tokeny tekst jest przeanalizowany pod względem składni i zaprezentowany w postaci fraz. Skróty są przetłumaczone na ich rozwinięcia (co prawda w formie podstawowej, ale niosą ze sobą informacje o odmianie). Pozostaje przetłumaczyć tokeny numeryczne i alfanumeryczne, co traktujemy jako kolejny etap normalizacji - transformację.

W etapie tym dzielimy wspomniane wyrażenia na kilka grup:

- liczebniki główne, np. „weź 3 szklanki wody",
- liczebniki porządkowe, np. „zapoznaj się z punktem 2. podpisywanej umowy",
- numery czytane cyfra po cyfrze,
- numery, których cyfry czytane są grupami o tej samej wielkości (parami, trójkami), np. nr dowodu osobistego,
- numery, których cyfry czytane są grupami o różnej wielkości, np. PESEL,

po czym odbywa się transfer do postaci słów.

Ostatnią fazą całego procesu jest generowanie poprawnej odmiany dla wszystkich niestandardowych wyrażeń. Informacje o fleksji pochodzą z wcześniejszych etapów, teraz należy przetworzyć je nadając wyrazom odpowiednią formę.

1.2.2. Zastosowanie

Normalizacja tekstu znajduje zastosowanie w dwóch różnych dziedzinach życia. Jako pierwsze należy wymienić ujednoznacznianie tekstu, będące etapem przejściowym w procesie przetwarzania tekstu (przystosowanie do dalszej obróbki). Normalizacja polega wówczas na rozwinięciu wszystkich niestandardowych słów – skrótów, akronimów i symboli – by tekst nie zawierał nieścisłości odnośnie znaczenia użytych słów i wyrażeń. Na przykład rozwinięcie zdania:

> „1) Art. 10 § 2 zmieniony przez art. 1 pkt 1 ustawy z dnia 5 listopada 2009 r. (Dz.U.09.206.1589) zmieniającej nin. ustawę z dniem 8 czerwca 2010 r."[5]

[5]Przykład pochodzi z [6]

może przyjąć postać:

> „1) Artykuł 10 paragraf 2 zmieniony przez artykuł 1 punkt 1 ustawy z dnia 5 listopada 2009 roku (Dziennik Ustaw 09.206.1589) zmieniającej niniejszą ustawę z dniem 8 czerwca 2010 roku."

Zabieg normalizacji w wyżej podanej postaci jest wykonywany przez zleceniodawców tłumaczeń, np. sąd zlecający tłumaczenie ustawy czy wyroku, od którego tłumacz żąda znormalizowanego tekstu. Tłumacz, nie będąc specjalistą w dziedzinie, z której zobowiązany jest przetłumaczyć tekst, nie obciąża tłumaczenia dodatkowymi błędami związanymi z niewłaściwym doborem rozwinięć skrótów. Dodatkowym atutem jest przyspieszenie procesu tłumaczenia. Przykładem może być chociażby użyty wcześniej w pracy skrót DSP. W różnych źródłach można odnaleźć następujące rozwinięcia i znaczenia:

- Digital Signal Processing – cyfrowe przetwarzanie sygnałów – dziedzina nauki i techniki,
- Digital Signal Processor – procesor sygnałowy – rodzaj mikroprocesora o architekturze przystosowanej do przetwarzania cyfrowych sygnałów
- Dom Słowa Polskiego – niegdyś największa drukarnia w Polsce.[6]

Drugie rozwiązanie jest bardziej skrupulatne i polega na przepisaniu treści do postaci bez niestandardowych wyrażeń – skrótów, symboli, liczb i innych wyrażeń, których przeczytanie wymagałoby szerszej wiedzy niż znajomość liter i umiejętność przetworzenia ich na dźwięk. To zastosowanie jest wykorzystywane w syntezatorach mowy. Programy czytające tekst nie przetwarzają go w żaden sposób – czytają dokładnie to co dostaną na wejściu, a to czego nie „umieją" przeczytać (wszystkie znaki spoza alfabetu i zbioru znaków przestankowych) – pomijają.

1.3. Named Entity Recorgnition and Translation (NERT)

Definicja 1.3 ***Named entity recognition*** *(NER) (identyfikacja jednostek nazwanych, ekstrakcja jednostek nazwanych) to podzadanie procesu pozyskiwania informacji, którego celem jest zlokalizowanie i sklasyfikowanie atomowych elementów tekstu do wcześniej zdefiniowanych kategorii, takich jak nazwy osób, organizacji, miejsc, wyrażenia opisujące czas, ilość, wartości monetarne, procenty, itp.*[7]

NERT jest środowiskiem do rozpoznawania i tłumaczenia jednostek nazwanych opartym o zestaw reguł, opracowanym przez naukowców z Uniwersytetu im. A. Mickiewicza w Poznaniu. Jednostkami nazwanymi są m.in. nazwy spółek, daty i wyrażenia dotyczące czasu,

[6] hasła i definicja pochodzą z [3]

[7] Definicja pochodzi z [4]

numery ustaw, numery telefonów, adresy, wyrażenia walutowe, imiona i nazwiska ludzi (również w połączeniu ze zwrotami grzecznościowymi), np. p. dyr. Roman Żółkiewicz.

Potrzeba utworzenia narzędzia do rozpoznawania bytów nazwanych wyniknęła podczas prac nad tłumaczeniem automatycznym. W artykule (Graliński, Jassem, Marcińczuk, 2009)[10] autorzy opisują, że błędy spowodowane złym tłumaczeniem jednostek nazwanych stanowią ponad 10% wszystkich błędów. Zatem rozwiązanie tego problemu znacznie poprawiłoby jakość tłumaczenia, a także normalizacji, o ile traktowana jest jako szczególny przypadek tłumaczenia automatycznego.

Do rozwoju polskiego NER-a znacznie przyczyniła się praca A. Przepiórkowskiego nad systemem SPEJD. Właśnie jako rozszerzenie formalizmu SPEJD, powstało narzędzie NERT, które poza rozpoznawaniem bytów nazwanych odpowiedzialne jest za ich tłumaczenie. System opiera się o zestaw reguł, o ustalonej, złożonej budowie. Listing 2 przedstawia przykładową regułę dopasowującą zwrot grzecznościowy „pani", imię i nazwisko:

```
Rule (person_rule)
Match: <base~[pP]ani> <{TITLEPL}> <{UL}\.> <{PersonInfix}>* <{ProperPL}>
Action: prepend(\1 \2:t \3 \4 \5; sem=female)
```

Listing 2 Reguła systemu NERT

Natomiast zdanie, które zostanie dopasowane do reguły z Listingu 2 może mieć postać: „Zapoznałem panią abp M. Niezwykłą.". Wynikiem zastosowania reguły dla powyższego zdania jest „Zapoznałem panią arcybiskup M. Niezwykłą.". W regule z Listingu 2 zastosowano sekcję `Match` o następującym znaczeniu: Poszukiwane wyrażenie powinno zawierać: zwrot którego formą podstawową jest „Pani" lub „pani", po którym podany jest tytuł, inicjał (imienia) i nazwisko. W części odpowiedzialnej za ustalenie wartości wyjścia (sekcja `Action`) użyto: kopiowanie grup: pierwszej, trzeciej, czwartej i piątej, tłumaczenia drugiej dopasowanej grupy i ustalono grupę semantyczną całego dopasowanego wyrażenia na „female".

Reguły składają się z szeregu sekcji, które należą do dwóch grup. Pierwsza odpowiedzialna jest za dopasowanie tłumaczonego (normalizowanego) tekstu do wzorca. Do sekcji dopasowujących należą:

- `Match` – warunek na poszukiwaną sekwencję tokenów,
- `Left` – warunek na sekwencję tokenów występujących bezpośrednio przed sekcją `Match`,
- `Right` – warunek na sekwencję tokenów występujących bezpośrednio po sekcji `Match`,
- `Before` – warunek na sekwencję tokenów występujących przed sekcją `Match`,
- `After` – warunek na sekwencję tokenów występujących po sekcji `Match`,

– `Exist` – warunek na sekwencję tokenów występujących w dowolnym miejscu zdania.

Druga grupa składa się z jednej sekcji `Action`, w której ustala się odpowiedniki do dopasowanych wcześniej grup tokenów. Zamiana tekstu wejściowego na jego tłumaczenie możliwa jest przez zastosowanie `prepend`. Dodaje ono tłumaczenie na początku listy tłumaczeń.

Następny krok to ustawienie modyfikatorów dla dopasowanych grup (grupy są numerowane od 1, zgodnie z kolejnością ich wystąpienia w użytych sekcjach: `Left`, `Right`, `Before`, `After`, `Exist` i `Match`). Mamy do dyspozycji:

– `n` – zmianę przypadku tokenu na mianownik,

– `s` – wycięcie fragmentu tokenu,

– `t` – użycie tłumaczenia słownikowego dla tokenu,

– `u` – zmianę pierwszej litery tokenu na wielką.

Natomiast już dla tekstu wyjściowego można ustawić wartość atrybutów:

– `gen` – osoba; wartości: rodzaj męski osobowy, męski zwierzęcy, męski rzeczowy, żeński, nijaki,

– `trginf` – opis sposobu odmiany wszystkich dopasowanych grup.

Przygotowane zgodnie z powyższymi wytycznymi reguły mają za zadanie rozpoznać w tekście jednostki nazwane i w bardziej odpowiedni sposób, niż tłumacząc słowa jedno po drugim, wygenerować tłumaczenie całości. Opisane rozwiązanie w praktyce sprawdza się bardzo dobrze, a tłumaczenie i normalizacja są tym lepsze, im więcej przemyślanych reguł zostaje dodanych.

1.4. Tłumaczenie automatyczne

Definicja 1.4 ***Tłumaczenie automatyczne****, tłumaczenie maszynowe (ang. Machine Translation) jest dziedziną językoznawstwa komputerowego, które zajmuje się stosowaniem algorytmów tłumaczenia tekstu z jednego języka (naturalnego) na drugi.*[8]

Tłumaczenie automatyczne zajmuje się tłumaczeniem tekstów z pominięciem czynnika ludzkiego. Rolę człowieka – tłumacza odgrywa program komputerowy, który przetwarza tekst w języku źródłowym, podając na wyjściu odpowiadający mu tekst, w języku docelowym – oczywiście w obrębie par języków, do których został stworzony. Istnieje kilka różnych podejść do rozwiązania problemu tłumaczenia maszynowego. Oto niektóre z nich:

[8]Definicja pochodzi z [5]

– **tłumaczenie oparte o parsing powierzchniowy** (parsing płytki, parsing częściowy) – podstawą jest analizator składniowy, rozpoznający elementy składowe zdania: grupy rzeczownikowe, grupy czasownikowe, bez ich wnikliwej analizy składniowej – nie gromadzi informacji o strukturze rozpoznanych elementów, ani ich roli w zdaniu. Dla tak rozpoznanych struktur tworzone jest drzewo parsingu, po czym generowane jest tłumaczenie.

– **tłumaczenie z użyciem parsingu pełnego** (parsingu głębokiego) – podstawą jest analizator składniowy, tworzący dla każdego zdania drzewo rozkładu. W powstałym grafie, poza wyrazami z rozkładanego zdania, znajdują się informacje o jego odmianie oraz pełnionej w zdaniu funkcji. W oparciu o zestaw reguł, brakująca o węzłach wiedza zostaje uzupełniona, po czym na podstawie zebranych danych generowane jest tłumaczenie końcowe.

– **tłumaczenie statystyczne** (SMT – Statistical Machine Translation) – podstawą jest bogaty korpus dwujęzyczny. Na jego podstawie generowane są możliwe tłumaczenia zdania. Dla każdego z nich wyznaczane jest prawdopodobieństwo *p(p—a)*, że ciąg znaków p w języku źródłowym jest tłumaczony przez ciąg a w języku docelowym. Znalezienie najlepszego tłumaczenia polega na wybraniu tego o najwyższym prawdopodobieństwie. Podczas tworzenia możliwych tłumaczeń zdania, a później w czasie wyznaczania prawdopodobieństwa należy uwzględnić ograniczenia czasu oraz ograniczenia wynikające z dostępności zasobów. Przykładem oprogramowania opartego o statystyki jest polsko-hiszpański tłumacz firmy Poleng, w którym korpusy pozyskane zostały z publicznie dostępnych dokumentów wielojęzycznych czy też Google Translate[9], dla którego korpusy pochodzą ze stron internetowych.

– **tłumaczenia oparte o przykłady** (EMTB – Example-Based Machine Translation) – podstawą jest korpus dwujęzyczny. W porównaniu do podejścia SMT oczekuje się, że korpus może być mniejszej wielkości, często ukierunkowany na konkretną dziedzinę, ale za to wyższej jakości. W tekstach języka źródłowego poszukiwane jest zdanie możliwie najbardziej podobne do tego, które należy przetłumaczyć. Dla tak znalezionego zdania pobierane jest jego tłumaczenie, usuwane są wyrażenia nie występujące w źródle, a w ich miejsce dodawane tłumaczenia tych, które wystąpiły w tekście wejściowym. Zdanie jest łączone celem uzyskania ostatecznego tłumaczenia.

[9]http://translate.google.com

1.5. System Polfest

System Polfest to syntezator mowy polskiej, używany w systemie Translatica[10]. Prace nad systemem rozpoczęły się w roku 2008 i trwają po dziś dzień. Kolejne wersje systemu prezentują wyższą jakość generowanego automatycznie dźwięku – głos jest coraz bardziej naturalny, ton wypowiedzi jest dopasowywany do czytanej treści (przestaje być monotonny), uwzględniane są akcenty w wyrazach i akcenty w zdaniu. Polepszeniu ulega również jakość normalizacji. Obsługiwany jest coraz większy zbiór skrótów, ulepszane czytanie znaków niealfabetycznych. Wszystko po to, by obdarzyć komputer głosem możliwie najbardziej czystym, płynnym i przyjaznym dla ucha.

1.6. Narzędzia normalizatora używanego w Polfest

1.6.1. Słownik normalizatora

W słowniku normalizatora zapisywane są dwa rodzaje haseł: leksemy, czyli pojedyncze wyrazy oraz idiomy. Idiomami są jednostki leksykalne złożone z przynajmniej dwóch tokenów (wyrazów lub znaków interpunkcyjnych). Opis leksemów i idiomów obejmuje szereg pól: `semantics`, `complementation`, `context`, `ssyntax`, `tsyntax`, `tinflection` i `sinflection`[11]. Z punktu widzenia prowadzonych prac interesować nas będzie sposób tworzenia i zawartość opisu struktury idiomu – `sinflection`. Pole zawiera następujące informacje:

- kod części mowy leksemu z opisującymi go flagami,
- opis struktury idiomu – oddzielone spacjami kody odmiany tokenów lub „0”, w przypadku gdy token nie podlega odmianie.

Do opisu pola dostępne są następujące części mowy (wraz z kodami):

- **C** – czasownik,
- **LG** – liczebnik główny,
- **LP** – liczebnik porządkowy,
- **P** – przymiotnik,
- **PR** – przyimek,
- **PS** – (portykuło)przysłówek,

[10] http://www.translatica.pl/

[11] Opis ten odziedziczony jest ze słownika systemu Translatica, przeznaczonego do tłumaczenia automatycznego.

– **R** – rzeczownik,

– **S** – spójnik,

– **So** – spójnik odmienny,

– **WY** – wykrzyknienie,

– **Z** – zdanie,

– **ZD** – zaimek dzierżawczy,

– **ZP** – zaimek przymiotny,

– **ZR** – zaimek rzeczowny,

– **ZS** – zaimek przysłowny.

Flagi niosą dodatkowe informacje gramatyczne:

– **1** – rodzaj męskoosobowy,

– **2** – rodzaj męskożywotny,

– **3** – rodzaj męskonieżywotny,

– **4** – rodzaj żeński,

– **5** – rodzaj nijaki,

Dodatkowo, wykrzyknikiem (!) oznaczony jest główny element.

```
Kazimierz Wielki R:1[R! R]
```

Listing 3 Hasło ze słownika polsko-polskiego

Idiom „Kazimierz Wielki" z Listingu 3, opisany jest w polu `sinflection` kodem R:1[R! R]. R:1 oznacza, że idiom jest rzeczownikiem/frazą rzeczownikową rodzaju męskoosobowego. [R! R] oznacza, że obydwa elementy frazy są rzeczownikami odmiennymi, a pierwszy z nich jest elementem głównym idiomu.

1.6.2. Parser składniowy

Zadaniem parsera składniowego jest opisanie struktury zdania za pomocą pewnej reprezentacji – najczęściej jest to struktura hierarchiczna reprezentowana przez drzewo składniowe. Wyrazy i jednostki wielowyrazowe zdania włączane są do drzewa w oparciu o pewien zestaw reguł. Reguły są zależne od języka i obejmują wszystkie zasady gramatyki dotyczące budowy zdania w zadanym języku.

Reguły parsera używanego w normalizatorze systemu Polfest mają postać przedstawioną na Listingu 4 (reguła rozpoznająca przykładowe wyrażenie „z pomysłem").

```
fprzyim = przyimek frz* %FPR[przyimek, frz]%
        P :== przyimek.P
```

Listing 4 Schemat budowy reguł parsera

Opis wprowadzonych oznaczeń:

- `fprzyim, przyimek, frz` – symbole nieterminalne gramatyki oznaczające kolejno frazę przyimkową, przyimek, frazę rzeczownikową,
- `*` - oznaczenie elementu głównego frazy (tzw. głowy),
- `%FPR[przyimek, frz]%` – sposób budowy gałęzi drzewa składniowego.
- `P :== przyimek.P` – uzupełnienie wartości atrybutu P (P - przypadek) powstającego węzła.

Każda reguła parsera składa się z trzech części:

- dopasowującej – lista symboli nieterminalnych gramatyki,
- przetwarzającej – sposób przetwarzania dopasowanych symboli nieterminalnych gramatyki,

– uzupełniającej – uzupełnienie wartości atrybutów powstającego węzła.

Fragment reguły parsera, który nas najbardziej interesuje, to sposób dopasowania analizowanego tekstu, w którym dla wszystkich wyrazów i wyrażeń zostały określone odpowiadające im symbole nieterminalne, a także sposób tworzenia struktury drzewa składniowego.

Część dopasowująca, to lista symboli nieterminalnych, które odpowiadają wyrazom i wyrażeniom z parsowanego tekstu. Dopasowanie do tej części umożliwia zastosowanie całej reguły do budowy drzewa składniowego. W przypadku reguły z Listingu 4 do drzewa zostanie dodany podwęzeł z potomkami dopasowanymi przez symbole: przyimek i frz.

Jako przykład rozważmy wyrażenie „czterdzieści cztery", dla którego zostanie zastosowana reguła z Listingu 5:

```
flicz = flicz liczebnik_główny* %FLG[flicz, liczebnik_główny]%
        P == flicz.P
```

Listing 5 Reguła parsera

Na wskutek zastosowania tej reguły wygenerowane zostanie drzewo składniowe w postaci podanej na Listingu 6:

```
() FLG: {(R,mnż), (P,mian)}
+---(0) LG: { (R,mnż), (P,mian)}
|   `---(0) '$czterdzieści':# { (R,mnż), (P,mian)}
|       `---(0) 'czterdzieści':# { }
`---(1) LG:# { (R,mnż), (P,mian)}
    `---(0) '$cztery':# { (R,mnż), (P,mian)}
        `---(0) 'cztery':# { }
```

Listing 6 Reguła parsera

W powyższym drzewie znakiem dolara oznaczone są formy podstawowe wyrazów wchodzących w skład analizowanego wyrażenia.

1.6.3. System transferu reguł

Transfer reguł polega na przetwarzaniu drzew składniowych, a dokładnie węzłów drzew składniowych (otrzymywanych podczas parsowania tekstu) z postaci źródłowej do drzew postaci docelowej.

Węzłem będziemy reprezentować jeden wyraz, hasło ze słownika lub jednostkę nazwaną. Do węzła przypisane są następujące atrybuty:

– kategoria – nazwa części mowy (np. rzeczownik), frazy (np. fraza rzeczownikowa) , forma podstawowa leksem (np.'$książka') lub wyraz (np. 'książkami'),

– etykieta – funkcja składniowa frazy reprezentowanej przez węzeł, np. dopełnienie, podmiot, element główny,

– atrybuty i ich wartości – liczba, przypadek, rodzaj, klasa semantyczna, itd.,

– węzły podrzędne (dzieci),

– węzeł nadrzędny (rodzic),

– odpowiednik węzła w drzewie docelowym (drzewie przetwarzanym podczas transferu reguł).

Drzewem źródłowym zbudowanym z węzłów jest drzewo otrzymane z tekstu wejściowego. Węzły oraz zapisane w nich dane pochodzą tylko i wyłącznie z etapu analizy tekstu. Zobaczmy przykładowe drzewo składniowe w postaci źródłowej wygenerowane dla wyrażenia „książka nt. wyprawy Kolumba" przedstawione na Listingu 7:

```
() FR: { (R,ż), (L,1), (P,mian)}
+--(0) R:# { (L,1), (R,ż), (P,mian)}
|   `--(0) '$książka':# { (L,1), (R,ż), (P,mian)}
|       `--(0) 'książka':# { }
`--(1) FPR: { (R,ż), (O,3), (L,1), (P,dop)}
    +--(0) PR: { (P,dop)}
    |   `--(0) '$nt.':# { }
    |       `--(0) 'nt.':# { }
    `--(1) FR:# { (R,ż), (L,1), (P,dop)}
        +--(0) R:# { (L,1), (R,ż), (P,dop)}
        |   `--(0) '$wyprawa':# { (L,1), (R,ż), (P,dop)}
        |     ~ `--(0) 'wyprawy':# { }
        `--(1) FR:dopełniacz { (R,mż), (L,1), (P,dop)}
            `--(0) R:# { (R,mż), (L,1), (P,dop)}
                `--(0) '$Kolumb':# { (L,1), (R,mż), (P,dop)}
                    `--(0) 'Kolumba':# { }
```

Listing 7 Drzewo składniowe postaci źródłowej

Reprezentacja podana na Listingu 7 oznacza, że wyrażenie „książka nt. wyprawy Kolumba" jest frazą rzeczownikową, na którą składa się rzeczownik „książka" oraz fraza rzeczownikowa z przysłówkiem „nt. wyprawy Kolumba". Węzeł opisujący rzeczownik „książka"

zbudowany jest z podwęzła '$książka', będącego leksemem. Jego podwęzeł to wyraz 'książka', odmieniony zgodnie z wartościami atrybutów opisujących ów leksem (liczba pojedyncza, rodzaj żeński, mianownik). Opis pozostałych podwęzłów jest analogoczny do budowy węzła opisującego rzeczownik „książka".

Drzewo docelowe każdego normalizowanego wyrażenia budowane jest w oparciu o strukturę drzewa źródłowego. Na rzecz każdego węzła drzewa źródłowego wywoływane są reguły transferu poprawiające i uzupełniające dane o węźle, na którym była uruchomiona reguła, jego rodzicu (węźle nadrzędnym) i dzieciach (węzłach podrzędnych).

Transfer drzew źródłowych przebiega w trzech następujących po sobie etapach:

– transfer właściwy – utworzenie pierwszej wersji docelowego drzewa składniowego z ustawionymi (na podstawie drzewa źródłowego i reguł transferu) kategoriami, etykietkami i atrybutami każdego węzła.

– modyfikacja – w przypadku dwujęzycznych tłumaczeń automatycznych – dostosowanie drzewa składniowego do gramatyki języka docelowego. Dla normalizacji etap modyfikacji nie ma większego znaczenia.

– generowanie postaci powierzchniowej – przejście po wszystkich węzłach drzewa postaci docelowej w celu wygenerowania końcowej postaci normalizowanego tekstu.

Przykładowe drzewo docelowe dla wyrażenia „książka nt. wyprawy Kolumba" wygenerowane w wyniku przejścia przez wszystkie etapy transferu ma postać podaną na Listingu 8.

```
() FR: { (P,N)}
+--(0) R:# { (Equiv,książka), (R,4), (L,1), (P,N)}
|   `--(0) '$książka':# { }
|       `--(0) 'książka':# { }
`--(1) : { }
    +--(0) : { (Equiv,na temat)}
    |   `--(0) '$nt.':# { }
    |       `--(0) 'nt.':# { }
    `--(1) FR:# { (P,G)}
        +--(0) R:# { (Equiv,wyprawy), (R,4), (L,1), (P,G)}
        |   `--(0) '$wyprawa':# { }
        |       `--(0) 'wyprawy':# { }
        `--(1) FR:dopełniacz { (P,G)}
            `--(0) R:# { (Equiv,Kolumba), (R,2), (P,G), (L,1)}
                `--(0) '$Kolumb':# { }
                    `--(0) 'Kolumba':# { }
```

Listing 8 Drzewo docelowe wyrażenia „książka nt. wyprawy Kolumba"

Jak widzimy na Listingu 8 – struktura drzewa w porównaniu z drzewem źródłowym nie uległa zmianie. Jest to zgodne z oczekiwaniami ze względu na fakt, że język źródłowy jest w normalizacji tożsamy z językiem docelowym. Obowiązują te same zasady gramatyczne, nie ma mowy o zmianie kolejności wyrazów we frazie. Zmieniły się za to wartości opisujące węzły. Między innymi dodany został atrybut `Equiv` z wartością będącą tłumaczeniem etykiety węzła 'nt.'. Przeprowadzony transfer drzew składniowych, w którego skład wchodzi uzupełnianie informacji o węzłach znacznie wpływa na poprawienie jakości normalizacji.

ROZDZIAŁ 2

Problemy w normalizacji tekstu

Odkrycie i zrozumienie przyczyn występowania błędów w normalizacji tekstów stanowi punkt wyjścia do rozpoczęcia prac programistycznych, mających na celu zmniejszenie lub całkowite ich usunięcie. Rozpoznanie należy przeprowadzić wykonując serię eksperymentów – testów regresywnych – różnych normalizatorów. Otrzymane w ten sposób wyniki umożliwią ocenę, jak rozwiązane są problemy normalizacji, podjęcie decyzji czy należy kontynuować prace nad polepszeniem jakości zwracanych wyników, a także ustalenie czy otrzymany wynik jest zadawalający.

Ze względu na fakt, że normalizacja na ogół nie występuje jako odrębne narzędzie, eksperymenty zostały przeprowadzone na systemach syntezy mowy, które używają normalizatorów. W ramach prac powstały testy syntezatorów następujących producentów: Nuance, Acapella group i IVONA Software.

Produktem udostępnionym przez firmę Nuance jest system RealSpeak[1]. Do celów testowania wybraliśmy język polski, do którego dostępny był głos kobiecy o imieniu Agata. Syntezator obsługuje łącznie 33 języki, w tym około 50 głosów męskich i żeńskich.

Kolejny syntezator Acapella Text-To-Speech[2] udostępniony przez firmę Acapella group z polskim głosem nazwanym Ania. System udostępnia syntezę mowy w 23 językach – 43 różnych głosów męskich i żeńskich.

Trzecim poddanym eksperymentom syntezatorem był system Ivona[3], firmy IVONA Software z ustawionym polskim głosem Ewy. Dla języka polskiego dostępne są cztery głosy: dwa męskie i dwa żeńskie. System obsługuje łącznie 7 języków, w tym 20 różnych głosów męskich i żeńskich.

Wyniki z przeprowadzonych eksperymentów zostały zaprezentowane w tabelach z rozróżnieniem na rodzaj błędu. Łatwo zauważyć jak działa każdy z syntezatorów, a także porównać ze sobą podobne, dostępne są na rynku narzędzia.

[1] http://212.8.184.250/tts
[2] http://www.acapela-group.com/text-to-speech-interactive-demo.html
[3] http://www.ivona.com/

2.1. Rozwijanie skrótów

2.1.1. Format

Występujące w tekstach skróty zapisywane są:

- bez kropki:
  - pierwsza i ostatnia litera skrótu jest pierwszą i ostatnią literą skracanego wyrazu, np. wg, mgr, dr,
  - symbol matematyczny, np. sin, ln, tg,
  - jednostka miary, np. kg, ha, m,
- z kropką na końcu:
  - ostatnia litera skrótu nie jest ostatnią literą skracanego wyrazu, np. dyr., im., dn.,
  - skrót utworzony jest z pierwszych liter skracanych wyrazów, z których wszystkie rozpoczynają się spółgłoską, np. ww., jw., np.,
- ze znakami interpunkcyjnymi wewnątrz skrótu:
  - skrót utworzony z pierwszych liter skracanych wyrazów, z których przynajmniej jeden, poza pierwszym, rozpoczyna się samogłoską, np. p.n.e., m.in.

Ponadto istnieją wyrażenia o różnym znaczeniu, skracane do takich samych ciągów liter. W celu rozróżnienia ich między sobą stosujemy zabieg:

- zmiany wielkości liter, np. dla skrótu „pt." [pod tytułem] i „P.T." [pleno titulo, łac. z zachowaniem należnych tytułów][4].
- zmiany interpunkcji, np. br. [bieżący rok] i b.r. [brak roku]

Inną przyczyną jest tradycja czy też przyzwyczajenie, czego przykładem jest skrót 'n.p.m.'– przyjęło się, że zapisuje się z kropkami po każdej pierwszej literze wyrazu składającego się na skrót.

2.1.2. Wyniki

Tabela 2.1 przedstawia znormalizowane teksty otrzymane podczas syntezy mowy w polskich syntezatorach.

[4] http://www.slownik-online.pl

Normalizowane wyrażenie	Agata	Ania	Ewa
przyszedł dr Jerzy Nowak	przyszedł doktor Jerzy Nowak	przyszedł doktor Jerzy Nowak	przyszedł de er Jerzy Nowak
ustaw wg wagi	ustaw według wagi	ustaw według wagi	ustaw według wagi
funkcje trygonometryczne: sin, cos, tg, ctg	funkcje trygonometryczne sin cos te gje ce te gje	funkcje trygonometryczne sin cos te gje ce te gje	funkcje trygonometryczne sinus kosinus te gje ce te gje
kupił 2 m wstążki	kupił dwa metry wstążki	kupił dwa metry wstążki	kupił dwa em wstążki
spotkasz prof. Jana Miodka	spotkasz profesor Jana Miodka	spotkasz profesor Jana Miodka	spotkasz profesor Jana Miodka
Uniwersytet im. Adama Mickiewicza	Uniwersytet imienia Adama Mickiewicza	Uniwersytet imienia Adama Mickiewicza	Uniwersytet imienia Adama Mickiewicza
książka pt. 'Żmija'	książka pod tytułem Żmija	książka pe te Żmija	książka pod tytułem Żmija
w 56 r. p.n.e.	w pięćdziesiątego szóstego roku przed naszą erą	w pięćdziesiątego szóstego roku przed naszą erą	w pięćdziesiąt sześć er przed naszą erą
nie widziałam dra Kowalika	nie widziałam dra Kowalika	nie widziałam dra Kowalika	nie widziałam dra Kowalika
przyszła p. Wieczorek	przyszła pana Wieczorek	przyszła pe Wieczorek	przyszła pe Wieczorek
praca p. Wieczorka	praca pana Wieczorka	praca pe Wieczorka	praca pe Wieczorka

Tabela 2.1. Porównanie normalizacji tekstu ze skrótami

2.1.3. Podsumowanie

Wyniki normalizacji zostały ocenione – oznaczone dwoma kolorami (- dobrze, - źle). Ocenie podlegało jedynie rozwijanie skrótów, dlatego pomimo niepoprawnej odmiany liczebnika w wyrażeniu 'w 56 r. p.n.e.', wynik normalizacji uznajemy jako prawidłowy.

W wierszach, które w całości zostały uznane za nieprawidłowe spodziewamy się następujących wyników:

- funkcje trygonometryczne: sin, cos, tg, ctg – funkcje trygonometryczne: sinus, cosinus, tangens, cotangens,
- spotkasz prof. Jana Miodka – spotkasz profesora Jana Miodka,
- nie widziałam dra Kowalika – nie widziałam doktora Kowalika,
- przyszła p. Wieczorek – przyszła pani Wieczorek.

Testy przeprowadzone na normalizatorach używanych w powyższych systemach syntezy wykazują szereg trudności związanych z normalizacją skrótów. Próba klasyfikacji błędów prowadzi do wyznaczenia trzech, niekoniecznie rozłącznych, grup.

Do pierwszej zaliczymy zbiór skrótów, których nie ma w słowniku normalizatora. W takim przypadku, rozwijany skrót nie jest normalizowany, a odczytywany jakby stanowił cały wyraz, np. „cos", bądź podejmowana jest próba literowania, co ma miejsce w przypadku skrótu „pt.".

Kolejna grupa to skróty, które są źle rozpoznane. Ma to miejsce w przypadku skrótów mających więcej niż jedno rozwinięcie, np. „p." – pan, pani, patrz, piętro. Rozpoznanie właściwego odpowiednika powinno być dokonywane na podstawie kontekstu, w którym wystąpił skrót. W ten sposób „p." w wyrażeniu „przyszła p. Wieczorek" w syntezatorze Agata powinno być znormalizowane do postaci „pani" zamiast „pan".

Ostatnią grupę stanowią skróty, których rozwinięcia mają niepoprawną odmianę. Z zapisu niektórych skrótów od razu wiemy jak powinny być one odmieniane, np. skrót „dra" – powinien zostać odczytany jako „doktora". Jednak dla zdecydowanej większości skrótów odmianę należy wywnioskować z treści zdania, a dokładniej z formy wyrazów, które skróty określają lub którymi są określane, np. „1 m" – jeden metr, ale „2 m" – dwa metry.

2.2. Rozwijanie skrótowców

2.2.1. Format

Wyróżniamy następujące typy skrótowców:

- **głoskowe** – pisane wielkimi literami, złożone z pierwszych liter skracanych wyrazów, które powinny być czytane łącznie, np. KUL, NATO, MON,
- **grupowe** – pisane małymi literami poza pierwszą, złożone z pierwszych sylab skracanych wyrazów, które powinny być czytane łącznie, np. Pafawag, Polfa, Wifama,
- **literowe** – pisane wielkimi literami, złożone z pierwszych liter skracanych wyrazów, które powinny być czytane osobno, np. AGD, PKO, UAM,

– **mieszane** – różna wielkość liter, budowa jest kombinacją skrótowców głoskowych, grupowych i literowych; takie skróty powinny być w części czytane osobno, a w części – łącznie, np. CPAN, PZMot, SPATiF,

– **złożeniowe** – złożone z wyrazu określanego i części wyrazu określającego, czytane łącznie, np. Investbank, Amerbank.

2.2.2. Wyniki

Tabela 2.2 przedstawia znormalizowane teksty otrzymane podczas syntezy mowy w polskich syntezatorach.

Normalizowane wyrażenie	**Agata**	**Ania**	**Ewa**
KUL, NATO, MON	kul, nato, mon	kul, nato, mon	kul, nato, mon
AGD	agd	a gje de	a gje de
PKO	pe ka o	pe ka o	pe ka o
UTP	utp	utp	utp
CPAN	ce pe an	cpan	cpan
PZMot	pe zet mot	pe zet mot	pzmot
SPATiF	spatif	spatif	spatif

Tabela 2.2. Porównanie normalizacji tekstu ze skrótowcami

2.2.3. Podsumowanie

Wyniki normalizacji zostały ocenione – oznaczone dwoma kolorami (- dobrze, - źle). W wierszach, które w całości zostały uznane za nieprawidłowe, spodziewamy się następujących wyników:

– UTP – u te pe,

– CPAN – ce pan.

W zależności od oczekiwań użytkownika, wyniki normalizacji możemy interpretować na dwa sposoby. Jeśli będziemy wymagać, żeby normalizator – podobnie jak to się dzieje podczas normalizacji skrótów – zwrócił na wyjściu tekst, który jest rozwinięciem skrótowca,

wówczas żaden wynik z Tabeli 2.2 nie uzyska pozytywnej oceny. Testowane normalizatory nie wskazują na próbę odszyfrowywania treści skrótowca.

Drugie podejście do rozwiązania problemu ze skrótowcami prezentowane jest w Tabeli 2.2 i polega na literowaniu, bądź czytaniu w grupach elementów składowych skrótowca zależnie od jego typu. Wymagania co do sposobu czytania nie zawsze znajdują pokrycie w systemach normalizacji. Otrzymujemy dwie grupy błędów. Do pierwszej zaliczymy wszystkie skrótowce, które czytane są jak jeden wyraz, podczas gdy powinny być literowane, np. AGD, UTP. Do kolejnej grupy należą skrótowce, które powinny być przeczytane w grupach, a są w całości literowane, np. CPAN, PZMot.

Próba pogrupowania błędów ze względu na przyczynę wystąpienia nie przyniosła spodziewanych rezultatów. Trudno jest jednoznacznie określić, dlaczego jedne skrótowce są poprawnie czytane, podczas gdy czytanie innych (tego samego typu) wiąże się z wystąpieniem błędów. Prawdopodobnie związane jest to z przechowywaniem w bazie danych skrótowców oraz opisu ich normalizacji: bardziej znane są przechowywane w bazie systemu i dlatego są prawidłowo czytane, natomiast mniej znane nie znajdują się w bazie i to powoduje ich błędny odczyt.

2.3. Przetwarzanie liczb arabskich

2.3.1. Format

W tekście mogą wystąpić liczby, które powinny być znormalizowane do postaci liczebników głównych lub liczebników porządkowych. Występują one w postaci:

- bez separatorów, np. 1000, 223, 987920,
- z sepaatorami, np. 1.000.000, 2 230, 987 920,
- ułamków dziesiętnych, np. 1,31, 22.3, 9879,20,
- ułamków zwykłych, np. 1/6, 1 1/2.

2.3.2. Wyniki

Tabela 2.3 przedstawia znormalizowane teksty otrzymane podczas syntezy mowy w polskich syntezatorach.

Normalizowane wyrażenie	Agata	Ania	Ewa
987020	dziewięć osiem siedem zero dwa zero	dziewięćset osiemdziesiąt siedem tysięcy dwadzieścia	dziewięćset osiemdziesiąt siedem tysięcy dwadzieścia
987 020	dziewięć osiem siedem zero dwa zero	dziewięćset osiemdziesiąt siedem tysięcy dwadzieścia	dziewięćset osiemdziesiąt siedem zero dwa zero
w 223 odcinku	w dwieście dwadzieścia trzy odcinku	w dwieście dwadzieścia trzy odcinku	w dwieście dwadzieścia trzy odcinku
kup 3,5 kg arbuza	kup trzy i pół kilogramów arbuza	kup trzy przecinek pięć kilograma arbuza	kup trzy i pół kilograma arbuza
błąd pomiaru wynosi 0,0002 mm	błąd pomiaru wynosi zero przecinek zero zero zero dwa minimetry	błąd pomiaru wynosi zero przecinek zero zero zero dwa minimetra	błąd pomiaru wynosi zero i dwie dziesięciotysięczne minimetra
dodaj 2/3 l mleka	dodaj dwie trzecie litry mleka	dodaj dwie trzecie el mleka	dodaj dwa slesz trzy el mleka
art. 242 pkt 1 kodeksu karnego	artykuł dwieście czterdzieści dwa punkty kodeksu karnego	art dwieście czterdzieści dwa pkt jeden kodeksu karnego	artykuł dwieście czterdziesty drugi punkt pierwszy kodeksu karnego

Tabela 2.3. Porównanie normalizacji tekstu z liczbami arabskimi

2.3.3. Podsumowanie

Wyniki normalizacji zostały ocenione – oznaczone dwoma kolorami (- dobrze, - źle). Ocenie podlegała jedynie normalizacja liczb, dlatego pomimo niepoprawnego rozwinięcia skrótów w wyrażeniu „dodaj 2/3 l mleka”, wynik normalizacji uznajemy jako prawidłowy. W wierszach, które w całości zostały uznane za nieprawidłowe spodziewamy się następujących wyników:

– w 223 odcinku – w dwieście dwudziestym trzecim odcinku.

Przeprowadzone testy pokazują, że normalizacja liczb wiąże się z wystąpieniem kilku rodzajów błędów. Pierwszy jest wynikiem używania separatorów w liczbach. Separatorem może być spacja, znak kropki lub przecinek oznaczający, że liczba jest ułamkiem dziesiętnym. Liczba zostaje wówczas podzielona w miejscach, w których postawiony jest separator i traktowana jak kilka liczb, a każda z nich normalizowana jest oddzielnie. W ten sposób dla „2 000" dostajemy „dwa zero zero zero" zamiast „dwa tysiące", a dla ułamka „0,0002" – „zero przecinek zero zero zero dwa" w zamian za oczekiwane „dwie dziesięciotysięczne".

Kolejny błąd wywołany jest nieprawidłową interpretacją liczby jako liczebnika głównego, podczas gdy powinna być traktowana jak liczebnik porządkowy. Trudność polega na tym, że zapis liczebników porządkowych niczym nie różni się od zapisu liczebników głównych. Dobrze pokazuje to następujący przykład: „w 223 odcinku" normalizowane do „w dwieście dwadzieścia trzy odcinku" zamiast „w dwieście dwudziestym trzecim odcinku". Sporadycznie zdarza się, że po liczebniku porządkowym występuje znak kropki, ale i to nie rozwiązuje problemu, na przykład w sytuacji, gdy kropka dodatkowo kończy zdanie. Pozostaje próba identyfikacji liczebnika na podstawie treści zdania.

W niektórych systemach syntezy, również ułamki zwykłe powodują błędy w normalizacji. W zapisie „2/3" ukośnik nie jest traktowany jak kreska ułamkowa, lecz jak ukośnik prawy. Wówczas dla ułamka „2/3" otrzymujemy odpowiednik w postaci „dwa slesz trzy", chociaż oczekiwaliśmy „dwie trzecie".

Zaskakującym jest fakt, że prawidłowo zapisana liczba (zapisana na sześciu pozycjach) jest źle znormalizowana przez system *Agata*. Jest to bardzo poważny błąd systemu, którego nie sposób pozostawić bez komentarza.

2.4. Przetwarzanie liczb rzymskich

2.4.1. Format

Liczby rzymskie zapisane są jako kombinacja znaków:

- I – 1,
- V – 5,
- X – 10,
- L – 50,
- C – 100,
- D – 500,
- M – 1000.

Sposób formułowania liczb rzymskich jest następujący:

Aby liczbę rzymską powiększyć należy z jej prawej strony dopisać liczbę o tym samym lub niższym rzędzie, np. chcąc zapisać 20, z prawej strony „X" dopisujemy „X", otrzymując „XX". Dla liczby „13" – do „X" dodajemy „III", co zapisuje się jako „XIII".

Aby liczbę rzymską pomniejszyć należy z lewej strony ostatniego znaku liczby dopisać liczbę o niższym rzędzie, np. 19 zapisujemy dopisując z lewej strony drugiego „X" (w liczbie „XX") znak „I", otrzymując „XIX".

2.4.2. Wyniki

Tabela 2.4 przedstawia znormalizowane teksty otrzymane podczas syntezy mowy w polskich syntezatorach.

Normalizowane wyrażenie	**Agata**	**Ania**	**Ewa**
Liceum Ogólnokształcące nr XIX	Liceum Ogólnokształcące nr ksiks	Liceum Ogólnokształcące nr dziewiętnasty	Liceum Ogólnokształcące nr dziewiętnaście
klasa VII	klasa siódma	klasa siódmy	klasa siedem
w XII wieku	w dwunastego wieku	w dwunastym wieku	w dwunastym wieku
tom II rozdział V	tom drugi rozdział piąty	tom drugi rozdział fał	tom drugi rozdział piąty
Benedykt XVI	Benedykt szesnaście	Benedykt szesnasty	Benedykt szesnasty
II wojna światowa	druga wojna światowa	druga wojna światowa	druga wojna światowa

Tabela 2.4. Porównanie normalizacji tekstu z liczbami rzymskimi

2.4.3. Podsumowanie

Wyniki normalizacji zapisane w Tabeli 2.4 zostały ocenione – oznaczone dwoma kolorami (- dobrze, - źle). Liczby rzymskie podobnie jak liczby arabskie podlegają odmianie, dlatego tekst z tego rodzaju błędem oceniany był negatywnie. Błędy nie związane z normalizacją liczb rzymskich nie mają wpływu na wystawioną ocenę.

Przeprowadzone testy normalizacji liczb rzymskich wykazały istnienie dwóch rodzajów błędów. Pierwszy polega na nie rozpoznawaniu wyrażenia jako liczby rzymskiej, a traktowaniu go jako napisu złożonego z liter. Efekt można zobaczyć normalizując wyrażenie „Liceum

Ogólnokształcące nr XIX". Dla jednego z normalizatorów otrzymujemy „Liceum Ogólnokształcące nr ksiks".

Drugi błąd jest wynikiem nadania złej formy (liczebnik główny, liczebnik porządkowy) oraz niepoprawnej odmiany, np. przypadku, rodzaju. W przeciwieństwie do liczb arabskich, liczby rzymskie na ogół normalizuje się do postaci liczebników porządkowych.

2.5. Czytanie dat

2.5.1. Format

Istnieje wiele formatów zapisu dat. Najczęściej używanymi są:

- dd.mm.rrrr, dd-mm-rrrr, dd/mm/rrrr – mm zapisany liczbą arabską lub rzymską,
- dd.mm.rr, dd-mm-rr, dd/mm/rr– mm zapisany liczbą arabską lub rzymską,
- dd miesiąc słownie rrrr,
- dd miesiąc słownie rr.

2.5.2. Wyniki

Tabela 2.5 przedstawia znormalizowane teksty otrzymane podczas syntezy mowy w polskich syntezatorach.

Normalizowane wyrażenie	Agata	Ania	Ewa
dnia 12.04.2011r.	dnia dwunastego kwietnia dwa tysiące jedenastego roku	dnia dwunastego kwietnia dwa tysiące jedenastego roku	dnia dwunasta zero cztery dwa tysiące jedenaście er
dnia 5.06.2000 r.	dnia pięć kropka zero sześć kropka dwa tysiące er	dnia piątego czerwca dwa tysiące roku	dnia piątego czerwca dwutysięcznego roku
dnia 4-07-1957r.	dnia cztery zero siedem tysiąc dziewięćset pięćdziesiąt siedem er	dnia cztery łącznik zero siedem łącznik tysiąc dziewięćset pięćdziesiątego siódmego roku	dnia cztery kreska zero siedem kreska jeden dziewięć pięć siedem er
dnia 8/12/2001 roku	dnia osiem slesz dwanaście slesz dwa tysiące jeden roku	dnia ósmego grudnia dwa tysiące pierwszego roku	dnia ósmego grudnia dwa tysiące pierwszego roku
dnia 20.12.98r.	dnia dwadzieścia kropka dwanaście kropka dziewięćdziesiąt osiem er	dnia dwudziestego grudnia tysiąc dziewięćset dziewięćdziesiątego ósmego roku	dnia dwudziesta dwanaście dziewięćdziesiąt osiem er
od 16 X 1987 roku	od szesnastego października tysiąc dziewięćset osiemdziesiątego siódmego roku	od szesnasty października tysiąc dziewięćset osiemdziesiątego siódmego roku	od szesnastego października tysiąc dziewięćset osiemdziesiątego siódmego roku
jest 23 maj 1900r	jest dwudziesty trzeci maj tysiąc dziewięćset er	jest dwudziesty trzeci maj tysiąc dziewięćset roku	jest dwadzieścia trzy maj tysiąc dziewięćset er

Tabela 2.5. Porównanie normalizacji tekstu z datami

2.5.3. Podsumowanie

Wyniki normalizacji zapisane w Tabeli 2.5 zostały ocenione – oznaczone dwoma kolorami (- dobrze, - źle). Ocenie podlegała całość normalizowanego wyrażenia. W wierszach, które dla wszystkich normalizatorów zostały uznane za nieprawidłowe spodziewamy się następujących wyników:

– dnia 4-07-1957r. – dnia czwartego lipca tysiąc dziewięćset pięćdziesiątego siódmego roku,

– jest 23 maj 1900r – jest dwudziesty trzeci maj tysiąc dziewięćsetnego roku.

Przedstawione w tabeli wyniki testów wskazują na dwa rodzaje błędów normalizacji dat. Pierwszym jest nierozpoznawanie wyrażenia opisującego datę, drugim – nieprawidłowa odmiana dnia i roku. Problem nierozpoznawania szczególnie często występuje w datach, w których separatorem jest znak inny niż kropka – w powyższych przykładach były to „-” oraz „/”. Jak się okazuje, duże znaczenie odgrywa również sposób zapisu opisu roku – czy między datą a „r.”, „r”, „rok” postawimy spację, czy nie. Zauważmy, że zapis „dnia 5.06.2000 r.” w dwóch na trzy przypadkach nie został prawidłowo znormalizowany, podobnie jak w przykładzie, w którym „r.” zapisane było bezpośrednio po roku – bez spacji.

Przyczyną nierozpoznania daty jest również zapis numeru roku na dwóch pozycjach (dwie ostatnie cyfry). Przykładem jest „dnia 20.12.98” rozwijany na przykład do „dnia dwadzieścia kropka dwanaście kropka dziewięćdziesiąt osiem”.

Zapisując wyrażenie opisujące datę, podając miesiąc słownie, otrzymujemy niepopraną odmianę dnia i roku.

2.6. Czytanie godzin

2.6.1. Format

Godziny zapisuje się według następujących formatów:

– gg:mm.ss

– gg:mm

2.6.2. Wyniki

Tabela 2.6 przedstawia znormalizowane teksty otrzymane podczas syntezy mowy w polskich syntezatorach.

Normalizowane wyrażenie	Agata	Ania	Ewa
godz. 21:20	godzina dwudziesta pierwsza dwadzieścia	godzina dwudziesta pierwsza dwadzieścia	godzina dwudziesta pierwsza dwadzieścia
4:05	zero pięć	godzina czwarta zero pięć	czwarta pięć
12:23.47	dwunasta dwadzieścia trzy czterdzieści siedem	godzina dwunasta dwadzieścia trzy i czterdzieści siedem sekund	dwanaście dwukropek dwadzieścia trzy kropka czterdzieści siedem
o godz. 17:00	o godzina siedemnasta	o godzinie siedemnastej zero zero	o godzinie siedemnastej

Tabela 2.6. Porównanie normalizacji tekstu z godzinami

2.6.3. Podsumowanie

Wyniki normalizacji zapisane w Tabeli 2.6 zostały ocenione – oznaczone dwoma kolorami (- dobrze, - źle).

Otrzymane wyniki wskazują, że w większości przypadków normalizatory prawidłowo rozwijają wyrażenia opisujące czas. Przypadki błędne pojawiają się, przy zapisie godziny w wersji z sekundami, rzadziej – gdy liczba oznaczająca godzinę jest jednocyfrowa.

Sporadycznym błędem jest również niepoprawna odmiana godziny, o ile podana jest w zdaniu. Przykładem jest wyrażenie „o godz. 17:00”, w którym źle rozpoznany przypadek doprowadził do otrzymania na wyjściu „o godzina siedemnasta”.

2.7. Czytanie charakterystycznych numerów

2.7.1. Numer PESEL

Format

Numer PESEL jest zapisywany jako liczba jedenastocyfrowa bez separatorów.

Wyniki

Tabela 2.7 przedstawia znormalizowane teksty otrzymane podczas syntezy mowy w polskich syntezatorach.

Normalizowane wyrażenie	Agata	Ania	Ewa
87101615141	osiemdziesiąt siedem miliardów sto jeden milionów sześćset piętnaście tysięcy sto czterdzieści jeden	osiemdziesiąt siedem miliardów sto jeden milion sześćset piętnaście tysięcy sto czterdzieści jeden	osiemdziesiąt siedem miliardów sto jeden milionów sześćset piętnaście tysięcy sto czterdzieści jeden
PESEL 87101615001	pesel osiemdziesiąt siedem miliardów sto jeden milionów sześćset piętnaście tysięcy jeden	pesel osiemdziesiąt siedem miliardów sto jeden milion sześćset piętnaście tysięcy jeden	pesel osiemdziesiąt siedem miliardów sto jeden milionów sześćset piętnaście tysięcy jeden

Tabela 2.7. Porównanie normalizacji tekstu z numerem PESEL

Podsumowanie

Wyniki normalizacji zapisane w Tabeli 2.7 zostały ocenione – oznaczone dwoma kolorami (- dobrze, - źle). Wszystkie wiersze w tabeli zostały uznane za nieprawidłowe. Spodziewamy się następujących wyników:

– 87101615141 – osiemdziesiąt siedem, dziesięć, szesnaście, piętnaście, sto czterdzieści jeden,

– PESEL 87101615001 – pesel osiemdziesiąt siedem, dziesięć, szesnaście, piętnaście, zero zero jeden.

Pomimo, że sposób czytania numerów PESEL nie jest sformalizowany, to spodziewamy się podzielenia liczby dziewięciocyfrowej na mniejsze części, których czytanie będzie bardziej zrozumiałe do słuchacza. Najczęstszym sposobem czytania jest czytanie po dwie cyfry. Liczba zapisana jest na nieparzystej ilości pozycji, dlatego proponujemy przeczytanie końcówki jako liczby trzycyfrowej. Właśnie takiej formy spodziewamy się jako wyniku normalizacji, podczas gdy otrzymujemy liczbę przeczytaną w całości, np. osiemdziesiąt siedem miliardów sto jeden milionów sześćset piętnaście tysięcy sto czterdzieści jeden.

2.7.2. Numer NIP

Format

Numer NIP jest liczbą dziesięciocyfrową zapisywaną według trzech schematów:

- nnnnnnnnnn – bez separatorów,
- nnn-nnn-nn-nn – z separatorem „-” po trzeciej, szóstej i ósmej cyfrze,
- nnn-nn-nn-nnn – z separatorem „-” po trzeciej, piątej i siódmej cyfrze.

Wyniki

Tabela 2.8 przedstawia znormalizowane teksty otrzymane podczas syntezy mowy w polskich syntezatorach.

Normalizowane wyrażenie	**Agata**	**Ania**	**Ewa**
NIP 100-123-02-10	nip sto sto dwadzieścia trzy zero dwa dziesięć	nip sto sto dwadzieścia trzy zero dwa dziesięć	nip sto sto dwadzieścia trzy zero dwa dziesięć
NIP 100-12-30-210	nip sto dwanaście trzydzieści dwieście dziesięć	nip sto łącznik dwanaście łącznik trzydzieści łącznik dwieście dziesięć	nip sto dwanaście trzydzieści dwieście dziesięć
NIP 1001230210	nip miliard milion dwieście trzydzieści tysięcy dwieście dziesięć	nip jeden miliard jeden milion dwieście trzydzieści tysięcy dwieście dziesięć	nip miliard jeden milionów dwieście trzydzieści tysięcy dwieście dziesięć

Tabela 2.8. Porównanie normalizacji tekstu z numerem NIP

Podsumowanie

Wyniki normalizacji zapisane w Tabeli 2.8 zostały ocenione – oznaczone dwoma kolorami (- dobrze, - źle). W wierszach, które dla wszystkich normalizatorów zostały uznane za nieprawidłowe spodziewamy się następujących wyników:

- NIP 1001230210 – nip sto dwanaście trzydzieści dwieście dziesięć.

Wyniki normalizacji pokazują, że normalizatory są bardzo dobrze przygotowane do czytania liczb z separatorami. W każdym teście poradziły sobie bezbłędnie. Jednak podobnie jak w przypadku numeru PESEL, pisanego bez separatorów, spodziewamy się, że pisząc łącznie numer NIP, zostanie on przeczytany w bardziej czytelnej formie. Dokładniej – w jednym z dwóch pozostałych formatów pisanych z separatorami.

2.7.3. Numer REGON

Format

Numer REGON jest zapisywany w dwóch formatach:

– liczba dziewięciocyfrowa bez separatorów,

– liczba czternastocyfrowa bez separatorów.

Wyniki

Tabela 2.9 przedstawia znormalizowane teksty otrzymane podczas syntezy mowy w polskich syntezatorach.

Normalizowane wyrażenie	**Agata**	**Ania**	**Ewa**
REGON 101478194	regon sto jeden milionów czterysta siedemdziesiąt osiem tysięcy sto dziewięćdziesiąt cztery	regon sto jeden milion czterysta siedemdziesiąt osiem tysięcy sto dziewięćdziesiąt cztery	regon sto jeden milionów czterysta siedemdziesiąt osiem tysięcy sto dziewięćdziesiąt cztery
REGON 19376498137645	regon dziewiętnaście bilionów trzysta siedemdziesiąt sześć miliardów czterysta dziewięćdziesiąt osiem milionów sto trzydzieści siedem tysięcy sześćset czterdzieści pięć	regon jeden dziewięć trzy siedem sześć cztery dziewięć osiem jeden trzy siedem sześć cztery pięć	regon jeden dziewięć trzy siedem sześć cztery dziewięć osiem jeden trzy siedem sześć cztery pięć

Tabela 2.9. Porównanie normalizacji tekstu z numerem REGON

Podsumowanie

Wyniki normalizacji zapisane w Tabeli 2.9 zostały ocenione – oznaczone dwoma kolorami (- dobrze, - źle). W wierszach, które dla wszystkich normalizatorów zostały uznane za nieprawidłowe spodziewamy się następujących wyników:

– REGON 101478194 – regon dziesięć czternaście siedemdziesiąt osiem sto dziewięćdziesiąt cztery,

– REGON 19376498137645 – regon dziewiętnaście trzydzieści siedem sześćdziesiąt cztery dziewięćdziesiąt osiem trzynaście siedemdziesiąt sześć czterdzieści pięć.

Sposób czytania numerów REGON nie jest ściśle określony, jednak spodziewamy się podzielenia liczby ma mniejsze segmenty i czytania według nowego podziału. Porównując wyniki testów z wynikami, których oczekujemy jako słuchacze, zauważamy, że zastosowanie podziału znacznie zwiększa zrozumiałość słuchanego tekstu.

2.7.4. Numer konta bankowego

Format

Numer konta bankowego to liczba dwudziestosześciocyfrowa z separatorami (spacjami) po drugiej, szóstej, dziesiątej, czternastej, osiemnastej i dwudziestej drugiej cyfrze.

Wyniki

Tabela 2.10 przedstawia znormalizowane teksty otrzymane podczas syntezy mowy w polskich syntezatorach.

Normalizowane wyrażenie	Agata	Ania	Ewa
numer konta 10 0000 0006 2376 1321 0764 7020	numer konta dziesięć zero zero zero zero zero zero zero sześć dwadzieścia trzy siedemdziesiąt sześć trzynaście dwadzieścia jeden zero siedem sześćdziesiąt cztery siedemdziesiąt dwadzieścia	numer konta dziesięć zero zero zero zero zero zero zero sześć dwadzieścia trzy siedemdziesiąt sześć tysiąc trzysta dwadzieścia jeden zero siedemset sześćdziesiąt cztery siedem tysięcy dwadzieścia	numer konta dziesięć zero zero zero zero zero zero zero sześć dwa tysiące trzysta siedemdziesiąt sześć tysiąc trzysta dwadzieścia jeden zero siedem sześć cztery siedem tysięcy dwadzieścia

Tabela 2.10. Porównanie normalizacji tekstu z numerem konta bankowego

Podsumowanie

Wyniki normalizacji zapisane w Tabeli 2.10 zostały ocenione – oznaczone dwoma kolorami (- dobrze, - źle).

Liczba opisująca numer konta jest podzielona separatorami spacji. W jednoznaczny sposób pokazują one granice liczb składających się na cały numer. Mimo wszystko zauważamy, że część normalizatorów nie odczytuje liczb zgodnie z separatorami i dzieli liczby na jeszcze mniejsze części. Ponadto podział nie jest wykonywany konsekwentnie dla każdej składowej, np. 2376 podlega segmentacji i wynikiem jest „dwadzieścia trzy siedemdziesiąt sześć", podczas gdy 1321 jej nie podlega i wyjściem jest „tysiąc trzysta dwadzieścia jeden".

2.7.5. Numer dowodu osobistego

Format

Numer dowodu osobistego zapisujemy zgodnie ze schematem:

– trzy pozycje (seria dowodu osobistego) – znaki alfabetyczne,

– sześć pozycji (numer dowodu osobistego) – cyfry.

Wyniki

Tabela 2.11 przedstawia znormalizowane teksty otrzymane podczas syntezy mowy w polskich syntezatorach.

Normalizowane wyrażenie	Agata	Ania	Ewa
dowód osobisty o numerze AHR745021	dowód osobisty o numerze a ha er siedem cztery pięć zero dwa jeden	dowód osobisty o numerze ahr siedemset czterdzieści pięć tysięcy dwadzieścia jeden	dowód osobisty o numerze ahr siedem cztery pięć zero dwa jeden

Tabela 2.11. Porównanie normalizacji tekstu z numerem dowodu osobistego

Podsumowanie

Wyniki normalizacji zapisane w Tabeli 2.11 zostały ocenione – oznaczone dwoma kolorami (- dobrze, - źle).

Jedynym błędem jaki zauważamy podczas normalizacji numeru dowodu osobistego jest brak literowania znaków alfabetycznych z części opisującej serię dowodu. W większości przypadków zostają odczytane tak jakby stanowiły wyraz.

2.7.6. Numer rejestracyjny pojazdu

Format

Polski system oznaczania pojazdów dopuszcza następujące numery rejestracji:

- I sposób – 2 litery + 5 cyfr, np. XY 12345,
- II sposób – 2 litery + 4 cyfry + 1 litera, np. XY 1234A,
- III sposób – 2 litery + 3 cyfry + 2 litery, np. XY 123AC,
- IV sposób – 2 litery + 1 cyfra + 1 litera + 3 cyfry, np. XY 1A234,
- V sposób – 2 litery + 1 cyfra + 2 litery + 2 cyfry, np. XY 1AC23,
- VI sposób – 3 litery + 1 litera + 3 cyfry, np. XYZ A123,
- VII sposób – 3 litery + 2 cyfry + 2 litery, np. XYZ 12AC,

– VIII sposób – 3 litery + 1 cyfra + 1 litera + 2 cyfry, np. XYZ 1A23,

– IX sposób – 3 litery + 2 cyfry + 1 litera + 1 cyfra, np. XYZ 12A3,

– X sposób – 3 litery + 1 cyfra + 2 litery + 1 cyfra, np. XYZ 1AC2,

– XI sposób – 3 litery + 2 litery + 2 cyfry, np. XYZ AC12,

– XII sposób – 3 litery + 5 cyfr, np. XYZ 12345,

– XIII sposób – 3 litery + 4 cyfry + 1 litera, np. XYZ 1234A,

– XIV sposób – 3 litery + 3 cyfry + 2 litery, np. XYZ 123AC,

– XV sposób - 3 litery + 1 litera + 2 cyfry + 1 litera, np. XYZ A12C,

– XVI sposób - 3 litery + 1 litera + 1 cyfra + 2 litery, np. XYZ A1CE,

– XVII sposób – tablice indywidualne,

– XVIII sposób – tablice samochodów zabytkowych,

– XIX sposób – tablice samochodów instytucji strzegących bezpieczeństwa, porządku publicznego, itp.[5]

Wyniki

Tabela 2.12 przedstawia znormalizowane teksty otrzymane podczas syntezy mowy w polskich syntezatorach.

Normalizowane wyrażenie	**Agata**	**Ania**	**Ewa**
CMGG957	ce em gje gje dziewięć pięć siedem	ce em gje gje dziewięćset pięćdziesiąt siedem	ce em gje gje dziewięćset pięćdziesiąt siedem
PO8S424	pe o osiem es czterysta dwadzieścia cztery	pe o osiem es czterysta dwadzieścia cztery	pe o osiem es czterysta dwadzieścia cztery

Tabela 2.12. Porównanie normalizacji tekstu z numerem rejestracyjnym pojazdu

[5]Klasyfikacja pochodzi z polskiej Wikipedii

Podsumowanie

Wyniki normalizacji zapisane w Tabeli 2.12 zostały ocenione – oznaczone dwoma kolorami (- dobrze, - źle).

Wyniki normalizacji są zadowalające.

2.7.7. Numer telefonu

Format

W użyciu jest wiele sposobów zapisu numerów telefonów. Najczęściej używane to:

- dla telefonów stacjonarnych:
  - +nn nn nnn nn nn – z numerem kierunkowym kraju,
  - nn nnn nn nn – bez numeru kierunkowego kraju,
- dla telefonów komórkowych:
  - +nn nnn nnn nnn – z numerem kierunkowym kraju,
  - nnn nnn nnn – bez numeru kierunkowego kraju.

Wyniki

Tabela 2.13 przedstawia znormalizowane teksty otrzymane podczas syntezy mowy w polskich syntezatorach.

Normalizowane wyrażenie	Agata	Ania	Ewa
tel. kom. 607 300 112	telefon kom. sześćset siedem milionów trzysta tysięcy sto dwanaście	telefon komórkowy sześćset siedem trzysta sto dwanaście	telefon kom. sześćset siedem trzysta sto dwanaście
tel. +48 315 17 01	telefon plus cztery osiem trzy jeden pięć siedemnaście zero jeden	telefon czterdzieści osiem trzysta piętnaście siedemnaście zero jeden	telefon plus czterdzieści osiem trzysta piętnaście siedemnaście zero jeden
tel. +48 52 315 17 01	telefon czterdzieści osiem pięćdziesiąt dwa trzysta piętnaście siedemnaście zero jeden	tel czterdzieści osiem pięćdziesiąt dwa trzysta piętnaście siedemnaście zero jeden	telefon plus czterdzieści osiem pięćdziesiąt dwa trzysta piętnaście siedemnaście zero jeden

Tabela 2.13. Porównanie normalizacji tekstu z numerem telefonu

Podsumowanie

Wyniki normalizacji zapisane w Tabeli 2.13 zostały ocenione – oznaczone dwoma kolorami (- dobrze, - źle).

W zależności od formatu zapisanego numeru telefonu otrzymujemy różne wyniki normalizacji. Najczęstsze błędy to: niezwracanie uwagi na użyte w liczbie separatory i czytanie liczby jak zwykłej liczby, a nie numeru telefonu. Kolejny błąd to pomijanie znaku „+” w numerach telefonów zapisywanych z numerem kierunkowym kraju.

2.7.8. Przetwarzanie znaków specjalnych

Format

Znaki ze zbioru: ‘ ! @ # $ % ^ & * ( ) - _ = + [{] } — ; : ’ ” , ¡ . ¿ / ?

Wyniki

Tabela 2.14 przedstawia znormalizowane teksty otrzymane podczas syntezy mowy w polskich syntezatorach.

Normalizowane wyrażenie	Agata	Ania	Ewa
# $ % ^ * { { : & *:	dolar procent i	hasz dolar procent daszek gwiazdka	dolar procent ampersand
# $ % ^ * { { : & *	dolar procent i	hasz dolar procent daszek gwiazdka	dolar procent daszek ampersand

Tabela 2.14. Porównanie normalizacji tekstu ze znakami specjalnymi

Podsumowanie

Wyniki normalizacji zapisane w Tabeli 2.14 zostały ocenione – oznaczone dwoma kolorami (- dobrze, - źle).

Znaki dodatkowe (specjalne) należy rozwijać do ich nazw. Natomiast jak pokazują testy – nie wszystkie znaki są przeczytane, a niektórym z nich nadane są błędne nazwy, np. „&” to „ampersand”. „I” jest angielskim odpowiednikiem znaku, a nie polskim.

2.7.9. Czytanie adresów elektronicznych

Format

Wyróżniamy następujące typy adresów elektronicznych:

- adresy internetowe, zapisane w postaci:
  - nazwa protokołu (opcjonalnie) – https://, http://, ftp://,
  - nazwa strony – znaki alfabetyczne, numeryczne, dodatkowe: # $ % &, itd.,
- adresy mailowe, zapisane w postaci:
  - nazwa konta pocztowego - znaki alfabetyczne, numeryczne, dodatkowe: _.,
  - znak @,
  - nazwa domeny – znali alfabetyczne, numeryczne, kropki.

Wyniki

Tabela 2.15 przedstawia znormalizowane teksty otrzymane podczas syntezy mowy w polskich syntezatorach.

Normalizowane wyrażenie	Agata	Ania	Ewa
www.wp.pl	wu wu wu wu pe pe el	wu wu wu kropka wu pe kropka pe el	wu wu wu kropka wu pe kropka pe el
http://poradnia.pwn.pl/lista.php?szukaj=zapiszemy&kat=1&od=10	ha te te pe dwukropek slash slash poradnia kropka pe wu en kropka pe el slash lista kropka pe ha pe znak zapytania szukaj znak równość zapiszemy i kat znak równość jeden i od znak równość jeden zero	ha te te pe dwukropek ukośnik ukośnik poradnia kropka pe wu en kropka pe el ukośnik lista kropka pe ha pe pytajnik szukaj równa się zapiszemy	ha te te pe dwukropek slash slash poradnia kropka pe wu en kropka pe el slash lista kropka pe ha pe szukaj równe zapiszemy i kat równe jeden ampersand od równe dziesięć
mw362@st.amu.edu.pl	em wu trzy sześć dwa małpa es te kropka amu kropka edu kropka pl	me wu trzy sześć dwa małpa es te kropka a em u kropka e de u kropka pe el	m wu trzysta sześćdziesiąt dwa małpa es te kropka amu kropka edu kropka pe el
m.wieczorek87@gmail.com	m kropka wieczorek osiem siedem małpa gje mail kropka kom	em kropka wieczorek osiem siedem małpa gmail kropka kom	em kropka wieczorek osiemdziesiąt siedem małpa gmail kropka com

Tabela 2.15. Porównanie normalizacji tekstu z adresem elektronicznym

Podsumowanie

Wyniki normalizacji zapisane w Tabeli 2.15 zostały ocenione – oznaczone dwoma kolorami (- dobrze, - źle). W wierszu, który dla wszystkich normalizatorów został uznany za nieprawidłowy spodziewamy się następującego wyniku:

– http://poradnia.pwn.pl/lista.php?szukaj=zapiszemy&kat=1&od=10 – ha te te pe dwu-

kropek slash slash poradnia kropka pe wu en kropka pe el slash lista kropka pe ha pe znak zapytania szukaj równe zapiszemy ampersand kat równe jeden ampersand od równe dziesięć.

Podając w treści testu adres elektroniczny oczekujemy, że każdy jego znak zostanie znormalizowany. Pojęcie normalizacji przyjmuje tutaj różne formy. Istnieją w adresach grupy liter, które należy przeliterować, np. www.wp.pl, podczas gdy inne powinny zostać przeczytane łącznie, jak wyraz „wieczorek" w adresie m.wieczorek87@gmail.com. Ustalenie prawidłowej wymowy nie jest zadaniem trywialnym. Przykładem błędu jest wspomniane wcześniej wyrażenie „m.wieczorek87@gmail.com", gdzie pierwsza litera nie została przeliterowana. Drobnym błędem jest także niewłaściwe interpretowanie niektórych znaków. W powyższych przykładach można zauważyć, że „&" zostaje rozwinięty do „i", pomimo że nazwa znaku to „ampersand". Ponadto adres internetowy
„http://poradnia.pwn.pl/lista.php?szukaj=zapiszemy&kat=1&od=10" został obcięty w miejscu wystąpienia znaku „&", co nie powinno mieć miejsca.

2.7.10. Prawidłowa fleksja

Języki fleksyjne, w tym język polski, wymagają od użytkownika stosowania się do zasad fleksji. W zagadnieniu normalizacji problem fleksji pojawia się w przypadku odczytu liczb lub skrótów, które powinny zostać odmienione. Zasady odmiany są ściśle określone składnią języka, a pomimo to moduły normalizacji stosowane w testowanych syntezatorach wykazują sporą liczbę błędów.

Przyczyną takiego stanu rzeczy jest brak wysokiej jakości, dostępnych narzędzi do analizy składniowej. Istniejące rozwiązania są dalekie od ideału, co ostatecznie powoduje błędy w normalizacji – etapie następującym po fazie analizy składniowej.

2.7.11. Szybkość działania

Każdy z testowanych normalizatorów, w postaci w jakiej jest udostępniony użytkownikom, połączony jest z syntezatorem dźwięku. Nie jest możliwe oddzielenie tych dwóch etapów syntezy mowy, dlatego też trudno określić ile czasu pochłania sama normalizacja. Można pokusić się o porównanie całkowitych czasów syntezy tych samych zdań, jednak wynik takiego badania trudno byłoby uznać za wiarygodny dla oceny czasu normalizacji.

ROZDZIAŁ 3

Normalizacja w systemie Polfest

Prace nad polepszeniem jakości normalizacji, przedstawione w niniejszej pracy, testowane były na systemie Polfest. Zadanie polegało na przeanalizowaniu sposobu działania, uzupełnieniu i modyfikacji istniejących oraz dopisaniu nowych reguł. Ostatecznym celem było polepszenie jakości normalizacji. W niniejszym rozdziale zostaną przedstawione rozpoznane błędy lingwistyczne, a także sposób ich rozwiązania.

3.1. Testy normalizatora

Podobnie jak syntezatory pochodzące z Nuance, Acapella i Ivona, również system Polfest wymagał przeprowadzenia szeregu testów rozpoznających, które obszary normalizacji są problematyczne. W tym przypadku nie ograniczyliśmy się do testów 'tematycznych', tzn. sprawdzających jakość tłumaczenia wyrażeń podlegających pewnym standardom czy konwencjom zapisu, np. numer dowodu osobistego czy numer PESEL. Testy zostały rozszerzone o korpus wygenerowany ze słownika frekwencyjnego Instytutu Podstaw Informatyki Polskiej Akademii Nauk (IPIPAN)[1]. Korpus zapisany był w postaci plików .xml i liczył 40748 zdań. Wybrane zdania miały tą własność, że nie wymagały normalizacji.

Rozpoczynając testy spodziewaliśmy się, że tekst wejściowy do procesu normalizacji będzie miał taką samą postać jak wyjściowy. Jednak porównanie wyników oczekiwanych z otrzymanymi wykazało istnienie szeregu błędów, które zostały przydzielone do dwóch głównych kategorii. Pierwsza charakteryzowała się usuwaniem wyrazów, np. wyrażenie „wszystkim tym, którzy przyczynili się" normalizowane było do postaci „wszystkim, którzy przyczynili się", a „Czym zasłużył sobie na wyróżnienie?" - „Czym zasłużył na wyróżnienie?". Druga natomiast wykazywała zmianę kolejności wyrazów w wyrażeniu wynikowym, np. „Nie było dotąd takich koncertów u Iwaszkiewicza." normalizowane do „Nie było takich koncertów dotąd u Iwaszkiewicza." czy też „które nie przekraczały mil ośmiu" - „które nie przekraczały ośmiu mil". Stwierdzono, że miejscami, w którym należy szukać rozwiązań wymienionych problemów są reguły parsera (Podrozdział 3.2) oraz reguły transferu (Podrozdział 3.4).

Drugim zbiorem testów były zdania, w których byliśmy nastawieni na normalizację konkretnych wyrażeń, np. skrótów, skrótowców, liczb arabskich, liczb rzymskich, itd. Wiedzieliśmy, że w podobnych systemach (Nuance, Ivona, Acapella) czytanie tego rodzaju treści powodowało błędy, co wskazywało, że i Polfest może mieć z nimi trudności. Stwierdzono,

[1] http://korpus.pl/

że rozwiązania tego rodzaju błędów można uzyskać przez poprawienie reguł NERT (Podrozdział 3.3) oraz słownika polsko-polskiego (Podrozdział 3.5).

Zebrane w fazie testów informacje były niezbędne do rozpoczęcia poprawiania systemu. Natomiast same testy były nierozłączną częścią prac prowadzonych na każdym poziomie normalizacji (poziom parsera, transferu reguł, NERT, słownik polsko-polski).

3.2. Dostosowanie reguł parsera składniowego

Parser używany przy normalizacji przygotowany był i przeznaczony początkowo do tłumaczeń dwujęzycznych (polsko-angielskich, angielsko-polskich, polsko-niemieckich, itd.). Dla wszystkich tych kierunków otrzymywane drzewa składniowe były poprawne, nie zauważono błędów tłumaczenia wynikających z niewłaściwego sparsowania tekstu. Jednak analiza wyników otrzymanych podczas normalizacji pokazała, że parser pomijał pewne ważne dla normalizacji informacje i nie zapisywał ich w drzewach lub zapisywał w niepoprawnej formie. Aby przywrócić prawidłową strukturę drzew, bez pomijania wyrazów i zmiany ich kolejności, należało poprawić reguły parsera.

3.2.1. Usuwanie wyrazów, zmiana kolejności wyrazów

Przeprowadzone na dużym korpusie testy wykazały, że w regułach parsera istnieją błędy istotne z punktu widzenia normalizacji. Ich efektem było nieuzasadnione usuwanie wyrazów oraz zmiana kolejności wyrazów podczas normalizacji. Po dokładnej analizie wyników testów dokonaliśmy szczegółowego podziału błędów. Do tej samej grupy zaliczaliśmy wyrażenia o takiej samej strukturze składniowej. W ten sposób zostało utworzonych 8 grup błędów. Należało przystąpić do odszukania i poprawienia reguł parsera odpowiedzialnych za występowanie znalezionych błędów. Ścieżkę procesu poprawiania możemy zobaczyć na przykładzie grupy zdań z Listingu 9.

WEJŚCIE: Cóż ona może z tego pojąć, co ją to obchodzi?
WYJŚCIE: Cóż ona może z tego pojąć, co to obchodzi?

WEJŚCIE: nie wiedziałby co z nią zrobić
WYJŚCIE: nie wiedziałby z co zrobić

WEJŚCIE: A co u niego?
WYJŚCIE: A u co?

WEJŚCIE: tą co go w wierszyku wstęgą opasała
WYJŚCIE: tą co w wierszyku wstęgą opasała

WEJŚCIE: Co mu odpowiesz, królowo?
WYJŚCIE: Co odpowiesz, królowo?

Listing 9 Przykłady zdań z usuwanymi przez parser zaimkami.

Reguły odpowiedzialne za wystąpienie błędów z Listingu 9 mają postać przedstawioną na Listingu 10.

```
(1) uzaimekktóry = 'co' ('$on'*|'$ona'*|'$ono'*) %FR['co']%
(2) uzaimekktóry = 'co' przyimek ('$on'*|'$ona'*|'$ono'*)
                          %FPR[przyimek,FR['co']]%
```

Listing 10 Reguły parsera

Reguły zapisane na Listingu 10 interpretowane są w następujący sposób:

– (1) uzaimekktóry: jeśli wyrażenie wejściowe (lub jego fragment) pasuje do wzorca: wyraz + 'co' + jeden z zaimków osobowych w formie podstawowej 'on', 'ona', 'ono', wówczas do drzewa składniowego wstaw podgałąź, w której zapisany jest węzeł odpowiadający wyrazowi 'co',

– (2) uzaimekktóry: jeśli wyrażenie wejściowe (lub jego fragment) pasuje do wzorca: wyraz + 'co' + dowolny przyimek + jeden z zaimków osobowych w formie podstawowej 'on', 'ona', 'ono', wówczas do drzewa składniowego wstaw podgałąź o dwóch węzłach: zawierającym dopasowany przyimek i wyraz 'co'.

W obydwu regułach z Listingu 10 zostało pominięte wstawienie do drzewa węzła odpowiadającego dopasowanemu zaimkowi. Poprawione reguły powinny mieć postać zapisaną na Listingu 11.

```
(1) uzaimekktóry = 'co' ('$on'*|'$ona'*|'$ono'*)   %FR['co','$on']%
(2) uzaimekktóry = 'co' przyimek ('$on'*|'$ona'*|'$ono'*)
                          %FPR[przyimek,FR['co'], '$on']%
```

Listing 11 Poprawione reguły parsera

3.3. Zastosowanie reguł NERT

Narzędzie NERT służy do rozpoznawania i tłumaczenia jednostek nazwanych (dokładny opis znajduje się w Podrozdziale 1.3) i właśnie z ich normalizacją było najwięcej trudności. Większość wyrażeń, o których mowa, zawierała cyfry i różnego rodzaju separatory, a każdy typ jednostki charakteryzował się innym sposobem odczytywania (grupowania cyfr). Tym samym każdy typ jednostek wymagał oddzielnego potraktowania.

3.3.1. Dostosowanie reguł NERT do normalizacji

Używany w środowisku NERT zestaw reguł przeznaczony był do tłumaczeń dwujęzycznych, a dopasowywane jednostki nazwane tłumaczone były zgodnie z zasadami obowiązującymi w języku angielskim i niemieckim. W normalizacji powodowało to zmianę kolejności, usuwanie i wstawianie tłumaczeń dopasowanych w sekcji `Match` grup, a także ustawianie niepoprawnej odmiany. W związku z tym w celu zastosowania reguł NERT w normalizacji należało je odpowiednio dostosować.

Tłumacząc przykładowe wyrażenie „w I kwartale 2011 r" za pomocą reguły polsko-angielskiej otrzymywaliśmy wynik: „w 1st quarter of 2010". Regułę z Listingu 12:

```
Rule( date; 1st~quarter II )
Match: <1|I> <base~kwartał> <[0-9]{4}> <r\.>?
Action: prepend(1st quarter of \3; sem=time_moment)
```

Listing 12 Reguła NERT

należało zastąpić regułą z Listingu 13:

```
Rule( date; 1st~quarter II )
Match: <1|I> <base~kwartał> <[0-9]{4}> <r\.>?
Action: prepend(\1 \2~\3:nn_y \4:t;
                          sem=time_moment, trginf=\1:0 \2:R \3:LP-P \4:R)
```

Listing 13 Poprawiona reguła NERT

Reguła działa w następujący sposób:

– przepisuje grupę pierwszą i drugą,

– tłumaczy grupę czwartą – zgodnie ze słownikiem polsko–polskim, w którym odpowiednikiem wyrażeń tekstowych są ich fonetyczne reprezentacje,

– używając flagi nn_y (opis flagi znajduje się w Podrozdziale 3.3.2) normalizuje rok,

– dla tłumaczeń dopasowanych grup ustala sposób odmiany:

  – grupa 1, 3 – według odmiany dla liczebnika porządkowego

  – grupa 2, 4 – według odmiany rzeczownikowej.

W efekcie wynikiem tłumaczenia ww. wyrażenia jest: „I kwartał dwa tysiące dziesiątego roku".

Tłumaczenie na język angielski wystąpiło również w przypadku normalizacji wyrażeń zawierających nazwiska, np. fraza „Spotkałam pana Nowaka." normalizowana była do postaci „Spotkałam Mr Nowak.", za co odpowiadała reguła z Listingu 14:

```
Rule(Person pan~name 5)
Match: <base~[pP]an> <{PersonInfix}>* <{ProperPL}; sem~surname>
Action: prepend(Mr \2 \3:nomu; sem=person)
```

Listing 14 Reguła NERT

Problem został rozwiązany przez usunięcie reguły dla normalizatora, ponieważ wyrażenia dopasowywane do niej nie wymagały żadnych zmian.

Podane wyrażenia są przykłami spośród zbioru wszystkich reguł NERT. Czynności podobne do przedstawionych należało wykonać dla pozostałych 99 reguł.

3.3.2. Normalizacja dat

Normalizacja dat wymagała napisania zestawu reguł obsługujących różne formaty ich zapisu. Wyniki normalizacji przy użyciu dotychczasowego zbioru reguł przedstawione są na Listingu 15.

WEJŚCIE: dnia 12.02.2000r.
WYJŚCIE: 12.02.2000

WEJŚCIE: z dnia 20 maja 1946 r
WYJŚCIE: z 25 # 1946

Listing 15 Wynik normalizacji dat bez użycia reguł NERT

Wyniki ujawniają dwa rodzaje błędów – pierwszy związany z samą normalizacją daty, drugi to usuwanie słów z wyrażenia, czego powodem jest opisane w Podrozdziale 3.2.1 używanie reguł do tłumaczenia polsko-angielskiego.

Z licznego zbioru różnych sposobów zapisu dat wybraliśmy te, które najczęściej występowały w tekstach polskich, czyli:

– dd.mm.rrrr,

– dd.mm.rr,

– dd-mm-rrrr,

– dd-mm-rr,

– dd/mm/rrrr,

– dd/mm/rr, dd

– {miesiąc słownie} rrrr,

– {miesiąc słownie} rrrr

i dla nich dopisane zostały reguły NERT.

Listing 16 zawiera reprezentację dodanych reguł (pozostałe są wariacjami odnośnie zapisu wyrazu „rok” ze spacją lub bezpośrednio po ostatniej cyfrze roku, w postaci „r”, „r.” i „rok” z wszystkimi odmianami, a także zapisem numeru roku na dwóch lub czterech pozycjach):

```
Rule( date 1 )
Left: {NOT_NUM}
Match: <base~{MonthNomPL}> <[0-9]{4}r\.?>
Action: prepend(\2 \3:nn_y roku; sem=time_moment)

Rule( date 2 )
Match: <base~dzień|dn\.>? <[0-9]{1,2}> <base~{MonthNomPL}>
                          <[0-9]{4}> <roku|rok|r\.?>?
Action: prepend(\1:t \2:nn_d \3 \4:nn_y roku;
                          sem=day, trginf=\1:R \2:LP-P \3:0 \4:0)

Rule( date 3 )
Match: <base~dzień|dn\.>?
               <[0123]?[0-9][\.\-][01][0-9][\.\-]([12][0-9])?[0-9][0-9]>
               <rok|roku|r\.?>?
Action: prepend(\1:t \2:nn_D roku; sem=day, trginf=\1:R \2:LP-P \3:0)



Rule( date 4 )
Match: <base~dzień|dn\.>? <[0123]?[0-9]> <[\/\.\-]> <[01][0-9]> <[\/\.\-]>
                          <([12][0-9])[0-9][0-9](r\.?)?> <rok|roku|r\.?>?
```

```
Action: prepend(\1:t \2:nn_d \4:nn_m \6:nn_y roku;
                            sem=day, trginf=\1:R \2:LP-P \3:0)

Rule( date 5 )
Match: <base~dzień|dn\.>? <[0123]?[0-9]>
                <I|II|III|IV|V|VI|VII|VIII|IX|X|XI|XII>
                <([12][0-9])?[0-9][0-9]> <rok|roku|r\.?>?
Action: prepend(\1:t \2:nn_d \3:nn_m \4:nn_y roku;
                            sem=day, trginf=\1:R \2:LP-P \3:0 \4:0)
```

Listing 16 Reguły NERT dla dat

W każdej z podanych reguł, kolejne dopasowywane w sekcjach `Match` grupy są obsługiwane w sekcjach `Action`. Schemat działania jest następujący:

- dla grup zapisanych bez flag ich wartość jest przepisywana w postaci, w której wystąpiła w tekście,
- dla grup zapisywanych z flagami (nn_d, nn_m, nn_y, nn_D) ich wartość jest zastępowana odpowiednikiem obliczonym przez użyte flagi,
- ustawiana jest wartość pola `trginf` (ustalany sposób odmiany wszystkich dopasowanych grup).

Dopisanie reguł NERT obsługujących wyrażenia opisujące daty wymagało dodania nowych flag generujących tłumaczenie: numeru dnia, numeru miesiąca, numeru roku oraz całej daty. W celu obsługi dat w formalizmie NERT wprowadzono nowe flagi:

- nn_d – flaga zwracająca numer dnia w postaci alfabetycznej, używana dla grup pasujących do wzorca [0123]?[0-9] (jedna lub dwie cyfry z przedziału 0 do 39),
- nn_m – flaga zwracająca nazwę miesiąca odpowiadającego podanej liczbie arabskiej lub rzymskiej, używana dla grup pasujących do wzorca [01][0-9] (dwucyfrowe liczby z przedziału 00 do 19), I, II, III, IV, V, VI, VII, VIII, IX, X, XI lub XII,
- nn_y – flaga zwracająca numer roku w postaci alfabetycznej, używana dla grup pasujących do wzorca ([12][0-9])?[0-9][0-9] (czterocyfrowe liczby z zakresu 1000 – 2999 lub dwucyfrowe liczby z zakresu 00 do 99)
- nn_D – flaga zwracająca postać alfabetyczną daty, używana dla grup pasujących do wzorca dd.mm.rrrr, dd.mm.rr, dd-mm-rrrr, dd-mm-rr, dd/mm/rrrr, dd/mm/rr.

Algorytm czytania daty podany został na Listingu 17.

```
function czytaj\_datę(data)
    dzień = pobierz\_dzień(data)
    miesiąc = obierz\_miesiąc(miesiąc)
    rok = pobierz\_rok(rok)
    if (rok jest dwuznakowy)
    then
        rok = ujednolicaj\_rok(rok)
    wynik = czytaj\_dzień(dzień)
    wynik = wynik + czytaj\_miesiąc(miesiąc)
    wynik = wynik + czytaj\_rok(rok)

    return wynik

function ujednolicaj\_rok(rok)
    tysiące\_setki = dwie pierwsze cyfry bieżącego roku
    dziesiątki\_jedności = dwie ostatnie cyfry bieżącego roku
    if (dziesiątki\_jedności < rok)
        uzupełnienie = tysiące\_setki - 1
    else
        uzupełnienie = tysiące\_setki

    return uzupełnienie + rok
```

Listing 17 Algorytm przetwarzania dat

Daną wejściową przedstawionego na Listingu 17 algorytmu jest data zapisana w jednym z formatów: dd.mm.rrrr, dd.mm.rr, dd-mm-rrrr, dd-mm-rr, dd/mm/rrrr, dd/mm/rr. W pierwszym kroku zostaje ona podzielona na części składowe, czyli numer dnia, miesiąca i roku, po czym ustalane jest w jakim formacie podano rok. O ile jest on liczbą dwucyfrową, to wyznaczane są cyfry z pozycji tysięcy i setek. Algorytm ich ustalania (ujednolicaj_rok) oparty jest o aktualną datę. Jeśli dwie ostatnie cyfry bieżącego roku są mniejsze od roku zapisanego na dwóch pozycjach, wówczas rozpatrywany rok dotyczy poprzedniego stulecia.

Funkcja czytająca dzień zwraca alfabetyczną reprezentację numeru dnia – liczebnik porządkowy, liczba pojedyncza, mianownik.

Funkcja czytająca miesiąc zwraca nazwę miesiąca odpowiadającego podanej liczbie arabskiej lub rzymskiej – rzeczownik, liczba pojedyncza, mianownik.

Funkcja czytająca rok zwraca alfabetyczną reprezentację numeru roku – liczebnik porządkowy, liczba pojedyncza, mianownik.

Dodane reguły i obsługujące je flagi dają wyniki normalizacji dat zapisane na Litingu 18.

WEJŚCIE: dnia 12.02.2000r.
WYJŚCIE: dnia dwunastego lutego dwutysięcznego roku

WEJŚCIE: wygasa z dniem 8.10.87r.
WYJŚCIE: wygasa z dniem ósmym października tysiąc dziewięćset osiemdziesiątego siódmego roku

Listing 18 Wyniki normalizacji dat po wprowadzeniu reguł

3.3.3. Normalizacja wyrażeń opisujących godzinę

Normalizacja godzin wymagała napisania reguł obsługujących następujące formaty jej zapisu: gg:mm, gg.mm, gg:mm.ss. Wyniki normalizacji przy użyciu dotychczasowego zbioru reguł przedstawione na Listingu 19 były dalekie od oczekiwanych.

WEJŚCIE: godz. 20:03
WYJŚCIE: godz. 20:03

WEJŚCIE: o godz. 12:20:03
WYJŚCIE: o godz. 12:20:03

Listing 19 Wyniki normalizacji godzin po wprowadzeniu reguł

W celu usprawnienia normalizacji godzin powstały reguły NERT zapisane na Listingu 20.

```
Rule( hour 1 )
Match: <base~godzina|godz.>? {HOUR}
Action: prepend(\1:t \2:nn_h; sem=time_moment, trginf=\1:R \2:LP-P 0, gnd=f)

Rule( hour 2 )
Match: <base~godzina|godz.>? <[0-2]?[0-9][\.:][0-6][0-9][\.:][0-6][0-9]>
Action: prepend(\1:t \2:nn_h;
                          sem=time_moment, trginf=\1:R \2:LP-P \3:LP-P, gnd=f)
```

Listing 20 Reguły NERT dla godzin

W każdej z podanych reguł, kolejne dopasowywane w sekcjach `Match` grupy są obsługiwane w sekcjach `Action` według następującego schematu:

- dla grup zapisanych z flagą `t` ich treść jest tłumaczona (rozwijanie skrótów),
- dla grup zapisywanych z flagą `nn_h` ich wartość jest zastępowana odpowiednikiem obliczonym przez użytą flagę,
- ustawiana jest wartość pola `trginf`, ustalany jest sposób odmiany wszystkich dopasowanych grup,
- ustawiana jest wartość pola `gnd` na rodzaj żeński.

Nowe reguły NERT-a wymagały dodania flagi generującej fonetyczną postać godziny. Powstała flaga nn_h, używana dla grup pasujących do wzorca [0-2]?[0-9][:][0-6][0-9]([:][0-6][0-9])? (jedna lub dwie cyfry opisujące godzinę, kropka lub dwukropek, dwie cyfry opisujące minuty, po czym opcjonalnie: kropka lub dwukropek, dwie cyfry opisujące sekundy).

Algorytm implementujący generowanie postaci alfabetycznej godzin zapisany jest na Listingu 21.

```
function czytaj_godzinę(godzina)
    godz = pierwszy + drugi znak z ,,godzina''
    minuty = czwarty + piąty znak z ,,godzina''
    wynik = czytaj_godz(godz)
    wynik = wynik + czytaj_minuty(minuty)
    if (godzina jest ośmioznakowa)
    then
        sekundy = siódmy + ósmy znak z ,,godzina''
        wynik = wynik + czytaj_sekundy(sekundy)

    return wynik
```

Listing 21 Algorytm przetwarzania godzin

W pierwszym kroku z podanej (jako argument wywołania) godziny wydzielane są liczby oznaczające ilość: godzin, minut i sekund (o ile w podanej na wejściu godzinie zapisane są sekundy). Następnie dla każdej z liczb wywoływane są funkcje generujące postać alfabetyczną, a otrzymane wyniki są łączone w jeden napis.

Należy zaznaczyć, że:

– funkcja czytająca godzinę zwraca alfabetyczną reprezentację liczby opisującej ilość godzin – liczebnik porządkowy, liczba pojedyncza, mianownik,

– funkcja czytająca minuty zwraca alfabetyczną reprezentację liczby opisującej ilość minut – liczebnik główny, liczba pojedyncza, mianownik,

– funkcja czytająca sekundy zwraca alfabetyczną reprezentację liczby opisującej ilość sekund – liczebnik główny, liczba pojedyncza, mianownik.

Dodane reguły i obsługujące je flagi dają wyniki normalizacji godzin zapisane na Listingu 22.

WEJŚCIE: godzina 20:03
WYJŚCIE: godzina dwudziesta zero trzy

WEJŚCIE: godz. 12:20:03
WYJŚCIE: godzina dwunasta dwadzieścia i trzy sekundy

Listing 22 Wyniki normalizacji godzin po wprowadzeniu reguł

3.3.4. Normalizacja numeru PESEL

Normalizacja wymagała dopisania reguł rozpoznających i we właściwy sposób przetwarzających wyrażenia zawierające numery PESEL. Decyzja o podjęciu prac nad czytaniem PESELów została podjęta po zapoznaniu się z wynikami dotychczasowej normalizacji. Przykładowe wyrażenia przedstawione są na Listingu 23.

WEJŚCIE: Mój numer PESEL to 88101287565.
WYJŚCIE: Mój numer pesel to osiemdziesiąt osiem miliardów sto jeden milionów dwieście osiemdziesiąt siedem tysięcy pięćset sześćdziesiąt pięć.

WEJŚCIE: Pacjent o numerze PESEL 87101604005.
WYJŚCIE: Pacjent o numerze pesel osiemdziesiąt siedem miliardów sto jeden milionów sześćset cztery tysiące pięć.

Listing 23 Wyniki normalizacji numeru PESEL bez reguł NERT

Każdy numer PESEL czytany był jak jedenastocyfrowa liczba. Jednak standardowo oczekuje się, że zostanie on podzielony na mniejsze części i znormalizowany sekwencjami. Reguły normalizacji PESELU uwzględniające takie oczekiwania zapisano na Listingu 24:

```
Rule( PESEL 1 )
Before: <PESEL|pesel>
Match: <\d{11}>
Action: prepend(\2:nn_p)

Rule( PESEL 2 )
Match: <PESEL|pesel> <\d{11}>
Action: prepend(\1 \2:nn_p)
```

Listing 24 Reguły NERT dla numerów PESEL

Reguły z Listingu 24 dopasowują i przetwarzają wyrażenia zawierające numery PESEL według następującego schematu:

- sprawdzane jest, czy użyto wyrazu „PESEL” lub „pesel” przed numerem
- sprawdzane jest, czy numer jest jedenastocyfrowy,
- dla grup spełniających powyższe warunki ustawia się flagę nn_p i stosuje się dla nich funkcję czytaj_pesel,

– nie jest konieczne ustawienie wartości pola trginf, gdyż znormalizowany PESEL jest wyrażeniem nieodmieniającym się.

Dopasowaną liczbę, dla której użyto flagi nn_p należało przetworzyć – wygenerować zapis fonetyczny liczby. Flaga wywołuje funkcję, której algorytm przedstawiony jest na Listingu 25

```
function czytaj_pesel(pesel)
    wynik = ,,''
    while (pesel jest więcej niż trzyznakowy)
        dwuznak = pierwszy + drugi znak ,,peselu''
        wynik = wynik + czytaj_liczbę(dwuznak)
        pesel = ,,pesel'' od trzeciego znaku włącznie
    wynik = wynik + czytaj_liczbę(pesel)

    return wynik
```

Listing 25 Algorytm przetwarzania numeru PESEL

Jedenastocyfrowa liczba zostaje podzielona na fragmenty dwucyfrowe i jeden fragment trzycyfrowy i czytana tak powstałymi segmentami. Wszystkie cząstkowe wyniki są liczebnikami głównymi zapisanymi w mianowniku liczby pojedynczej.

Wyniki normalizacji numerów PESEL po wprowadzeniu reguł i dodaniu flagi nn_p przedstawia Listingu 26.

WEJŚCIE: Mój numer PESEL to 88101287565.
WYJŚCIE: Mój numer pesel to osiemdziesiąt osiem, dziesięć, dwanaście, osiemdziesiąt siedem, pięćset sześćdziesiąt pięć.

WEJŚCIE: Pacjent o numerze PESEL 87101604005.
WYJŚCIE: Pacjent o numerze PESEL osiemdziesiąt siedem, dziesięć, szesnaście, zero cztery, zero zero pięć.

Listing 26 Wyniki normalizacji numerów PESEL po wprowadzeniu reguł NERT

3.3.5. Normalizacja numeru NIP

Normalizacja numerów NIP bez reguł rozpoznających i przetwarzających ten rodzaj bytu nazwanego przynosiła wyniki podane na Listing 27.

WEJŚCIE: Numer NIP Juliana Kowalskiego to 980-87-87-569.
WYJŚCIE: Numer nip Juliana Kowalskiego to 980-87-87-569.

WEJŚCIE: Klient o numerze NIP 198-09-12-000.
WYJŚCIE: Klient o numerze nip 198-09-12-000.

Listing 27 Wyniki normalizacji numerów NIP bez reguł NERT

Wprowadzenie reguł w środowisku NERT zapisanych na Listing 28 oraz obsługa specyficznego sposobu czytania numeru NIP poprawiło wyniki normalizacji.

```
Rule( NIP 1 )
Before: <NIP|nip>
Match: <\d{3}-\d{3}-\d{2}-\d{2}|\d{3}-\d{2}-\d{2}-\d{3}|\d{10}>
Action: prepend(\2:nn_n)

Rule( NIP 2 )
Match: <NIP|nip> <\d{3}-\d{3}-\d{2}-\d{2}|\d{3}-\d{2}-\d{2}-\d{3}|\d{10}>
Action: prepend(\1 \2:nn_n)
```

Listing 28 Reguły normalizacji numerów NIP

Zapisane na Listingu 28 reguły dopasowują liczby dziewięciocyfrowe z separatorami postawionymi zgodnie z zasadami zapisu numerów NIP (opis w Podrozdziale 2.7.2). Wyraz „NIP” lub „nip” zostaje w obydwu przypadkach przepisany bez zmian, natomiast sam numer NIP przetwarzany jest przez flagę nn_n. Flaga ta wywołuje funkcję dzielącą numer w miejscach wystąpienia separatorów, normalizuje (rozwija) liczby otrzymane po podziale i zastępuje nimi napis, dla którego użyta była omawiana flaga. Normalizując przykładowe zdania otrzymujemy lepsze wyniki (Listing 29).

WEJŚCIE: Numer NIP Juliana Kowalskiego to 980-87-87-569.
WYJŚCIE: Numer nip Juliana Kowalskiego to dziewięćset osiemdziesiąt, osiemdziesiąt siedem, osiemdziesiąt siedem, pięćset sześćdziesiąt dziewięć.

WEJŚCIE: Klient o numerze NIP 198-09-12-000.
WYJŚCIE: Klient o numerze nip sto dziewięćdziesiąt osiem, zero dziewięć, dwanaście, zero zero zero.

Listing 29 Wyniki normalizacji numerów NIP po wprowadzeniu reguł NERT

3.3.6. Normalizacja numeru REGON

Normalizacja numeru REGON przynosiła wyniki zapisane na Listingu 30.

WEJŚCIE: Numer REGON Juliana Kowalskiego to 98081210191334.
WYJŚCIE: Numer REGON Juliana Kowalskiego to dziewięćdziesiąt osiem bilionów osiemdziesiąt jeden miliardów dwieście dziesięć milionów sto dziewięćdziesiąt jeden tysięcy trzysta trzydzieści cztery.

WEJŚCIE: Klient o nr REGON 297601872.
WYJŚCIE: Klient o numerze REGON dwieście dziewięćdziesiąt siedem milionów sześćset jeden tysięcy osiemset siedemdziesiąt dwa.

Listing 30 Wyniki normalizacji numerów REGON bez reguł NERT

Numery REGON czytane były jak zwykłe liczby, co utrudniało ich zrozumienie. Oczekuje się, że numer zostanie podzielony i czytany mniejszymi segmentami, podobnie jak to miało miejsce w przypadku numeru PESEL. Reguły NERT dopasowujące wyrażenie zawierające numer REGON, a przede wszystkim umożliwiające jego dalsze przetwarzanie zapisane są na Listingu 31.

```
Rule( REGON 1 )
Before: <REGON|regon>
Match: <\d{9}|\d{14}>
Action: prepend(\2:nn_r)

Rule( REGON2 )
Match: <REGON|regon> <\d{9}|\d{14}>
Action: prepend(\1 \2:nn_r)
```

Listing 31 Reguły NERT dla numerów REGON

Użyta flaga nn_r działa analogicznie do flagi nn_p dla normalizacji numerów PESEL, przechodząc od początku do końca dopasowanego numeru.

Wyniki normalizacji po uzupełnieniu zbioru reguł prezentowane są na Listingu 32.

WEJŚCIE: Numer REGON Juliana Kowalskiego to 98081210191334.
WYJŚCIE: Numer REGON Juliana Kowalskiego to dziewięćdziesiąt osiem, zero osiem, dwanaście, dziesięć, dziewiętnaście, trzynaście, trzydzieści cztery.

WEJŚCIE: Klient o nr REGON 297601872.
WYJŚCIE: Klient o numerze REGON dwadzieścia dziewięć, siedemdziesiąt sześć, zero jeden, osiemset siedemdziesiąt dwa.

Listing 32 Wyniki normalizacji numerów REGON po wprowadzeniu reguł NERT

3.3.7. Normalizacja numeru konta bankowego

Normalizacja numeru konta bankowego przynosiła rezultaty, których przykład zapisany jest na Listingu 33.

WEJŚCIE: Wpłać na numer konta 87 0000 7680 4392 0926 8230 1020
WYJŚCIE: Wpłać na numer konta osiemdziesiąt siedem 0000 7680 4392 0926 8230 1020

Listing 33 Wyniki normalizacji numeru konta bankowego

Błąd został naprawiony przez dodanie reguły NERT zapisanej na Listingu 34.

```
Rule( bank account )
Before: <base~konto>
Match: <(\d{2} ?\d{4} \d{4} \d{4} \d{4} \d{4} \d{4})|(\d{26})>
Action: prepend(\2:nn_b)
```

Listing 34 Reguły NERT normalizacji numerów kont bankowych

Reguła dopasowuje i przetwarza wyrażenia zawierające numery kont bankowych według schematu:

– sprawdzane jest, czy w wyrażeniu wystąpił wyraz „konto”,

– sprawdzane jest, czy numer jest dwudziestosześciocyfrowy,

– dla grupy spełniającej powyższe warunki zastosowana jest flaga nn_b.

Flaga nn_b pobiera zawartość trzeciej grupy i zwraca napis przeczytany dwójkami – działa na tej samej zasadzie co flaga nn_p dla normalizacji numerów PESEL.

Wyniki normalizacji po uzupełnieniu zbioru reguł prezentowane są na Listingu 35.

WEJŚCIE: Wpłać na numer konta 87 0000 7680 4392 0926 8230 1020
WYJŚCIE: Wpłać na numer konta osiemdziesiąt siedem, zero zero, zero zero, siedemdziesiąt sześć, osiemdziesiąt, czterdzieści trzy, dziewięćdziesiąt dwa, zero dziewięć, dwadzieścia sześć, osiemdziesiąt dwa, trzydzieści, dziesięć, dwadzieścia

Listing 35 Wyniki normalizacji numerów kont bankowych po wprowadzeniu reguł NERT

3.3.8. Normalizacja numeru dowodu osobistego

Normalizacja numeru dowodu osobistego dotyczy przykładów z Listingu 36.

WEJŚCIE: Legitymuje się dowodem osobistym o numerze AHE981201.
WYJŚCIE: Legitymuje się dowodem osobistym o numerze AHE981201.

WEJŚCIE: Numer dowodu osobistego AHE981201.
WYJŚCIE: Numer dowodu osobistego AHE981201.

Listing 36 Wyniki normalizacji numeru dowodu osobistego

Wyniki wskazują na potrzebę dodania obsługi przetwarzania omawianego typu jednostek nazwanych. Dodane zostały reguły środowiska NERT (Listing 37).

```
Rule( identity card 1)
Before: <base~dowód> <base~osobisty>?
Match: <base~numer>? <[A-Z]{3}\d{6}>
Action: prepend(\3 \4:nn_i)

Rule( identity card 2~)
Match: <base~numer> <base~dowód> <base~osobisty>? <\:>? <[A-Z]{3}\d{6}>
Action: prepend(\1 \2~\3 \4~\5:nn_i)
```

Listing 37 Reguły NERT dla normalizacji numeru dowodu osobistego

Dopasowany ciąg nieleksykalny zostaje przetworzony przez użytą na jego rzecz flagę nn_i. Flaga literuje znaki alfabetyczne, natomiast sześć ostatnich cyfr czyta jak liczby dwucyfrowe. Ważnym elementem podczas dopasowania wyrażenia do zapisanych reguł jest odróżnienie numeru dowodu osobistego od dowolnego innego ciągu nieleksykalnego, który czytany jest zgodnie z opisem z Podrozdziału 3.4.3.

Używając nowych reguł normalizacji, dla przykładowych numerów dowodów osobistych otrzymamy wyniki podane na Listingu 38.

WEJŚCIE: Legitymuje się dowodem osobistym o numerze AHE981201.
WYJŚCIE: Legitymuje się dowodem osobistym o numerze a ha e dziewięćdziesiąt osiem, dwanaście, zero jeden.

WEJŚCIE: Numer dowodu osobistego AHE981201.
WYJŚCIE: Legitymuje się dowodem osobistym o numerze a ha e dziewięćdziesiąt osiem, dwanaście, zero jeden.

Listing 38 Wyniki normalizacji numerów dowodów osobistych po wprowadzeniu reguł

3.4. Poprawienie reguł transferu

Reguły transferu operują na drzewach składniowych wygenerowanych przez parser. Dokładny opis procesu przetwarzania drzew składniowych opisany jest w Podrozdziale 1.6.3.

3.4.1. Usuwanie „nie" ze zdania

Podobnie jak w przypadku reguł parsera, również reguły transferu wymagały naniesienia poprawek. Testy i ich analiza wykazały istnienie jednego błędu generowanego na tym etapie przetwarzania tekstu. Przykładami, w których uwidocznił się ten błąd są zdania z Listingu 39.

WEJŚCIE: Takie rzeczy nie mijają bez śladu.
WYJŚCIE: Takie rzeczy mijają bez śladu.

WEJŚCIE: Na innych kierunkach nie wygląda to tak dobrze
WYJŚCIE: Na innych kierunkach wygląda to tak dobrze

WEJŚCIE: Siedemnaście parkingów z pewnością nie zaspokaja potrzeb
WYJŚCIE: Siedemnaście parkingów z pewnością zaspokaja potrzeb

WEJŚCIE: Już wówczas plan państwowy nie miał charakteru dyrektywnego
WYJŚCIE: Już wówczas plan państwowy miał charakteru dyrektywnego

Listing 39 Przykłady zdań z usuniętą negacją czasownika

Należy zaznaczyć, że błąd nie występował dla wszystkich wyrażeń z zanegowanym czasownikiem. Dla zdań „A więc nikt nie będzie jadł obiadu." i „Nie będziemy nigdy stawać do raportu" normalizacja przebiegła poprawnie.

Pomijanie negacji czasownika miało miejsce jedynie w zdaniach, w których wystąpiły jednostki wielowyrazowe (zapisane w słowniku), takie jak „mijać bez śladu", „dobrze wyglądać", „zaspokajać potrzebę" czy „mieć charakter". Przykładowe drzewo dla wyrażenia „nie mija bez śladu" ma postać przedstawioną na Listingu 40:

```
() Z: { }
‘--(0) FC: { (Neg,1), (O,3), (R,nil), (L,1)}
    +--(0) C:# { (O,3), (R,nil), (L,1)}
       ‘--(0) ’$mijać bez śladu’:# { (O,3), (R,nil), (L,1)}
           ‘--(0) ’mijać bez˜śladu’:# { }
```

Listing 40 Przykładowe drzewo składniowe wyrażenia „nie mija bez śladu"

Negacja czasownika pomimo, że zapisana była w węźle drzewa składniowego, nie była przekładana na jej odpowiednik (wyraz „nie") w zdaniu wygenerowanym z postaci docelowej drzewa. Należało uzupełnić regułę przetwarzania węzła kategorii C o dodanie brakującego węzła „nie" w przypadkach, w których był on pomijany. Obecna reguła, której algorytm przedstawiony jest na Listingu 41 pozwoliła na całkowite rozwiązanie omawianego problemu.

```
C=>C {
    if (jeden z podwęzłów jest jednostką wielowyrazową &&
        węzeł ma ustawiony atrybut Neg na wartość 1 &&
        nie ma podwęzła o kategorii ,,nie’’)
    then
        dodaj do drzewa węzeł o kategorii ,,nie’’
}
```

Listing 41 Reguła transferu reguł

Zapisana na Listingu 41 reguła wywoływana jest dla każdego węzła o kategorii C (czasownik) w drzewie źródłowym i przetwarza go na węzeł o kategorii C w drzewie docelowym. Dodatkowo sprawdza, jaka jest struktura zbudowanego w ten sposób węzła. O ile jednym z podwęzłów węzła C jest jednostka wielowyrazowa i atrybut Neg równy jest 1, jednak nie ma podwęzła o kategorii „nie", wówczas należy dodać węzeł o kategorii „nie".

Przykładowe drzewo dla wyrażenia „nie mija bez śladu", po dodaniu reguły z Listingu 41 ma postać podaną na Listingu 42:

```
() : { }
‘--(0) FC: { }
    +--(0) ’nie’:# { }
    +--(1) C: { (Inflection,), (Equiv,mija), (Context,), (Eid,19)}
       ‘--(0) ’$mijać bez śladu’:# { }
           ‘--(0) ’mijać bez śladu’:# { }
```

Listing 42 Przykładowe drzewo składniowe wyrażenia „nie mija bez śladu” po poprawieniu reguły

Dla błędnie normalizowanego dotąd wyrażenia „nie mijać bez śladu” do postaci „mijać bez śladu”, otrzymujemy prawidłową postać „nie mijać bez śladu”. Dodana reguła pozwoliła na przywrócenie pomijanej dotąd negacji dla wszystkich haseł wielowyrazowych.

3.4.2. Normalizacja adresów elektronicznych

Zbiór reguł został uzupełniony o funkcje normalizacji adresów elektronicznych oraz jednostek nieleksykalnych. W obydwu przypadkach zmiany zostały wprowadzone na poziomie generowania postaci powierzchniowej. Listing 43 przedstawia sposób, w jaki otrzymaliśmy zamierzone rezultaty.

```
function tłumacz_adres_elektroniczny(interpretowany, pozostały)
if (pozostały jest pusty)
then
       if (interpretowany nie jest wyrazem)
        then
                wynik = literuj(interpretowany)
        else
                wynik = interpretowany
        return wynik
else
        znak = pierwszy znak z ’pozostały’
        pozostały = ’pozostały’ bez pierwszego znaku
        if (znak nie jest literą)
        then
                wynik = literuj(interpretowany)
                wynik = wynik + literuj(znak)
        else
                interpretowany = interpretowany + znak
```

```
return wynik + tłumacz_adres_elektroniczny(interpretowany, pozostały)
```

Listing 43 Algorytm normalizacji adresów elektronicznych

Przeanalizujmy sposób działania algorytmu z Listingu 43 dla przykładu adresu internetowego: www.onet.pl.

Rozwiązanie proponuje, aby z adresu elektronicznego wydzielane były ciągi alfabetyczne ('www', 'onet', pl). Dla każdego takiego napisu podejmowana jest decyzja czy czytać go jak wyraz, czy literować ('www' – literuj, 'onet' – wyraz, 'pl' - literuj). Napisy będące wynikami podejmowanych decyzji (kolejno: 'wu wu wu', 'onet', 'pe el') są ze sobą łączone przeliterowanymi znakami niealfabetycznymi ('.' - 'kropka', '.' - 'kropka'), które początkowo je rozdzielały. Otrzymujemy wynik: 'wu wu wu kropka onet kropka pe el'.

3.4.3. Normalizacja ciągów nieleksykalnych

Napisy rozpoznane jako nieleksykalne (numery seryjne sprzętu, nazwy obiektów astronomicznych – w szczególności planetoid, np. (35396) 1997 XF11, numery rejestracyjne samochodów, itd.) zostają w całości przeliterowane. Dotychczasowa normalizacja napisu: „Numer seryjny: P0502571." do postaci: „numer seryjny: P0502571" została sprowadzona do oczekiwanej formy „numer seryjny pe zero pięć zero dwa pięć siedem jeden".

3.5. Wykorzystanie słownika polsko-polskiego

Słownik jest bazą wiedzy na temat sposobu normalizacji pojedynczych wyrazów i jednostek wielowyrazowych. Zapisane w nim dane wraz z poprawnym opisem sposobu odmiany znacznie polepszają jakość normalizacji. Budowa słownika została szczegółowo opisana w Podrozdziale 1.6.1.

3.5.1. Normalizacja liczb rzymskich

Znajomość sposobu uzupełniania słownika okazała się być niezbędna podczas usuwania błędów normalizacji liczb rzymskich. Dla większości przypadków przetwarzanie było bezbłędne, o czym świadczą wyniki zapisane na Listingu 44.

WEJŚCIE: w XX w.
WYJŚCIE: w dwudziestym wieku

WEJŚCIE: chodzi do V klasy
WYJŚCIE: chodzi do piątej klasy

WEJŚCIE: czyta IX rozdział
WYJŚCIE: czyta dziewiąty rozdział

Listing 44 Przykłady wyrażeń z poprawnie normalizowanymi liczbami rzymskimi

Istniały jednak wyrażenia, w których liczby rzymskie nie podlegały normalizacji. Listingu 45 przedstawia niektóre z nich.

WEJŚCIE: w XV wieku
WYJŚCIE: w XV wieku

WEJŚCIE: czyta rozdział III
WYJŚCIE: czyta rozdział III

Listing 45 Przykłady wyrażeń z błędnie normalizowanymi liczbami rzymskimi

Przyczyną takiego stanu były błędy w słowniku. Błędy dotyczyły zawartości pola sinflection dla idiomów z liczbami rzymskimi (opis odmiany liczby rzymskiej był ustawiony na 0). Problem ten rozwiązano następująco: wypisano ze słownika wszystkie wyrażenia zawierające liczby rzymskie, wygenerowano dla nich poprawny opis pola sinflection i w ten sposób wygenerowanymi hasłami uzupełniono bazę danych. Całość prac została wykonana półautomatycznie, za pomocą napisanych skryptów Perl-owych.

Słownik został uzupełniony hasłami, których przykłady znajdują się na Listing 46:

```
XV wiek R3:[0 R!] piętnasty wiek R3:[piętnasty_LP-P R!]
rozdział III R3:[R! 0] rozdział trzeci R3:[R! trzeci_LP-P]
II wojna światowa R4:[0 R!~R]
                          druga wojna światowa R4:[drugi_LP-P R!~R]

XIX wiek przed naszą erą R3:[0 R!~0 0~0]
                          dziewiętnasty wiek przed naszą erą
                          R3:[dziewiętnasty_LP-P R!~0 0~0]
```

Listing 46 Przykładowe hasła (z opisem) w słowniku polsko-polskim

Podane na Listingu 46 przykłady należy odczytywać w następujący sposób:

– **XV wiek** (1) – forma podstawowa wyrażenia do znormalizowania,

– **R3:[0 R!]** – opis wyrażenia (1) wyjaśniający, w jakiej formie może ono wystąpić w tekście do znormalizowania; dla podanego przykładu: fraza rzeczownikowa, rodzaj męskonieżywotny, złożona z wyrazu nieodmieniającego się i wyrazu odmieniającego się jak rzeczownik

– **piętnasty wiek** (2)– forma podstawowa wyrażenia, którym zostanie zastąpione normalizowane wyrażenie,

– **R3:[piętnasty_LP-P R!]** – opis wyrażenia wyjaśniający, w jaki sposób należy odmieniać wyrażenie (2); dla podanego przykładu: fraza rzeczownikowa, rodzaj męskonieżywotny, złożona z wyrazu odmieniającego się jak liczebnik porządkowy i wyrazu odmieniającego się jak rzeczownik.

Wyniki zapisane na Listingu 47 potwierdzają słuszność podjętych kroków.

WEJŚCIE: w XV wieku
WYJŚCIE: w piętnastym wieku

WEJŚCIE: czyta rozdział III
WYJŚCIE: czyta rozdział trzeci

Listing 47 Wyniki normalizacji wyrażeń z liczbami rzymskimi po uzupełnieniu słownika

ROZDZIAŁ 4

Podsumowanie

Celem niniejszej pracy była analiza problemów związanych z normalizacją tekstu. Została ona przeprowadzona na trzech najbardziej popularnych syntezatorach mowy (Acapella, Ivona, Nuance), a także na systemie Polfest. Otrzymany zestaw błędów posłużył do określenia zakresu prac przy normalizatorze używanym w syntezatorze Polfest, a prowadzone i omówione w Rozdziale 3 prace znacznie poprawiły jakość otrzymywanych wyników.

Omówione w pracy podejście do procesu normalizacji wzorowane było na wieloetapowym tłumaczeniu automatycznym. Wielokrotne przetwarzanie normalizowanego tekstu daje ogromne ilości informacji o wyrazach i wyrażeniach, co widać w drzewach składniowych generowanych w kolejnych etapach normalizacji. Dzięki zebranym danym wiemy jakie związki zachodzą pomiędzy wyrazami w zdaniu. Jednak wiele z otrzymywanych informacji jest zbędnych, a ich otrzymanie i przechowywanie niepotrzebnie zużywa takie zasoby jak pamięć i czas. Należałoby zastanowić się, czy można przyspieszyć proces normalizacji bez obniżania jakości otrzymanych wyników. Należy rozważyć możliwość zastąpienia głębokiego parsera parserem płytkim, co najprawdopodobniej uprościłoby dalszą obróbkę tekstu i pozwoliło na znaczną oszczędność czasu.

Dodatek A

DODATEK A

Prezentacja wyników implementacji

Rozdział przedstawia wyniki normalizacji tekstu normalizatorem używanym w systemie Polfest. Aby przedstawić zakres wykonanych prac, a przede wszystkim pokazać, w jakim stopniu prace implementacyjne wpłynęły na polepszenie jakości normalizacji, dla każdego przykładu (tekstu wejściowego, linia 1) podany jest wynik normalizacji sprzed rozpoczęcia prac (linia 2), a także po zakończeniu, tj. na dzień 5. maja 2011r. (linia 3).

A.1. Reguły parsera

A.1.1. Usuwanie wyrazów

1. Usuwanie „tym" ze zdania

tym wszystkim, którzy przyczynili się
wszystkim, którzy przyczynili się
tym wszystkim, którzy przyczynili się

wszystkim tym, którzy przyczynili się
wszystkim, którzy przyczynili się
wszystkim tym, którzy przyczynili się

2. Usuwanie „sobie" ze zdania

Czym zasłużył sobie na wyróżnienie?
Czym zasłużył na wyróżnienie?
Czym zasłużył sobie na wyróżnienie?

3. Usuwanie zaimków ze zdania

Cóż ona może z tego pojąć, co ją to obchodzi?
Cóż ona może z tego pojąć, co to obchodzi?
Cóż ona może z tego pojąć, co ją to obchodzi?

nie wiedziałby co z nią zrobić
nie wiedziałby z co zrobić
nie wiedziałby co z nią zrobić

Co mu odpowiesz, królowo?
Co odpowiesz, królowo?
Co mu odpowiesz, królowo?

4. Usuwanie „właśnie" ze zdania

to właśnie on go zabił
to on go zabił
to właśnie on go zabił

to właśnie z nim bawiłem się
z to nim bawiłem się
to właśnie z nim bawiłem się

A.1.2. Zmiana kolejności wyrazów

1. Zmiana kolejności wyrazów w zdaniu z „siebie samym":

bo tacy odkrywcy ośmieszają i siebie samych i całą hecę
bo tacy odkrywcy ośmieszają i samych siebie i całą hecę
bo tacy odkrywcy ośmieszają i siebie samych i całą hecę

2. Zmiana kolejności wyrazów w zdaniu z rzeczownikiem i liczebnikiem

cząsteczki o energii dziesięć do potęgi czternastej
cząsteczki o dziesięć energii do potęgi czternastej
cząsteczki o energii dziesięć do potęgi czternastej

długości siedemnastu i trzech dziesiątych kilometra
siedemnastu długości i trzech dziesiątych kilometra
długości siedemnastu i trzech dziesiątych kilometra

różnicą jednej tylko bramki trzy do czterech
różnicą jednej tylko trzy bramki do czterech
różnicą jednej tylko bramki trzy do czterech

3. Zmiana kolejności wyrazów w zdaniu z „nie” + „być”(odmienione)

Nie było dotąd takich koncertów u Iwaszkiewicza.
Nie było takich koncertów dotąd u Iwaszkiewicza.
Nie było dotąd takich koncertów u Iwaszkiewicza.

Nie było wówczas Polski wśród państw niepodległych.
Nie było Polski wówczas wśród państw niepodległych.
Nie było wówczas Polski wśród państw niepodległych.

4. Zmiana kolejności wyrazów w zdaniu „to” + przyimek

to dzięki jego świetnemu biegowi
dzięki to jego świetnemu biegowi
to dzięki jego świetnemu biegowi

A.2. Reguły NERT

A.2.1. Normalizacja dat

gazeta z dnia 25 maja 1946r
gazeta z 25 # 1946
gazeta z dnia dwudziestego piątego maja tysiąc dziewięćset czterdziestego szóstego roku

wygasa z dniem 13.01.2012r.
wygasa z dniem 13.01.2012 rok
wygasa z dniem trzynastym stycznia dwa tysiące dwunastego roku

od dnia 16.10.87r.
od dnia 16.10.87 roku
od dnia szesnastego października tysiąc dziewięćset osiemdziesiątego siódmego roku

Sprzedaż drukarek w październiku 2006 roku.
Sprzedaż drukarek w # 2006.
Sprzedaż drukarek w październiku dwa tysiące szóstego roku.

A.2.2. Normalizacja wyrażeń opisujących czas

godzina 14:00
godzina 14:00
godzina czternasta zero zero

godzina 13:03
godzina 13:03
godzina trzynasta zero trzy

A.2.3. Normalizacja numerów PESEL

Mój numer PESEL to 88101287565.
Mój numer pesel to osiemdziesiąt osiem miliardów sto jeden milionów dwieście osiemdziesiąt siedem tysięcy pięćset sześćdziesiąt pięć.
Mój numer pesel to osiemdziesiąt osiem, dziesięć, dwanaście, osiemdziesiąt siedem, pięćset sześćdziesiąt pięć.

o numerze PESEL 99021202021
o numerze pesel dziewięćdziesiąt dziewięć miliardów dwadzieścia jeden milionów dwieście dwa tysiące dwadzieścia jeden
o numerze PESEL dziewięćdziesiąt dziewięć, zero dwa, dwanaście, zero dwa, zero dwadzieścia jeden

A.2.4. Normalizacja numerów NIP

Numer NIP Juliana Kowalskiego to 980-87-87-569.
Numer nip Julian Kowalski to 980-87-87-569.
Numer nip Juliana Kowalskiego to dziewięćset osiemdziesiąt, osiemdziesiąt siedem, osiemdziesiąt siedem, pięćset sześćdziesiąt dziewięć.

Klient o numerze NIP 198-091-20-00.
Klient o numerze nip 198-091-20-00.
Klient o numerze NIP sto dziewięćdziesiąt osiem, zero dziewięćdziesiąt jeden, dwadzieścia, zero zero.

A.2.5. Normalizacja numerów REGON

Numer REGON Juliana Kowalskiego to 98081210191334.
Numer REGON Julian Kowalski to dziewięćdziesiąt osiem bilionów osiemdziesiąt

jeden miliardów dwieście dziesięć milionów sto dziewięćdziesiąt jeden tysięcy trzysta trzydzieści cztery.
Numer REGON Juliana Kowalskiego to dziewięćdziesiąt osiem, zero osiem, dwanaście, dziesięć, dziewiętnaście, trzynaście, trzydzieści cztery.

Klient o nr REGON 297601872.
Klient o numerze REGON dwieście dziewięćdziesiąt siedem milionów sześćset jeden tysięcy osiemset siedemdziesiąt dwa.
Klient o numerze REGON dwadzieścia dziewięć, siedemdziesiąt sześć, zero jeden, osiemset siedemdziesiąt dwa.

A.2.6. Normalizacja numerów kont bankowych

Wpłać na numer konta 87 0000 7680 4392 0926 8230 1020
Wpłać na numer konta osiemdziesiąt siedem 0000 7680 4392 0926 8230 1020
Wpłać na numer konta osiemdziesiąt siedem, zero zero, zero zero, siedemdziesiąt sześć, osiemdziesiąt, czterdzieści trzy, dziewięćdziesiąt dwa, zero dziewięć, dwadzieścia sześć, osiemdziesiąt dwa, trzydzieści, dziesięć, dwadzieścia

A.2.7. Normalizacja numerów dowodów osobistych

Legitymuje się dowodem osobistym o numerze AHE981201.
Legitymuje się dowodem osobistym o numerze AHE981201.
Legitymuje się dowodem osobistym o numerze a ha e dziewięćdziesiąt osiem, dwanaście, zero jeden.

dowód o numerze KJA928790
dowód o numerze KJA928790
dowód o numerze ka jot a dziewięćdziesiąt dwa, osiemdziesiąt siedem, dziewięćdziesiąt

numer dowodu to KJG511201
numer dowodu to KJG511201
numer dowodu to ka jot gje pięćdziesiąt jeden, dwanaście, zero jeden

numer dowodu QWN091233
numer dowodu QWN091233
numer dowodu qu wu en zero dziewięć, dwanaście, trzydzieści trzy

A.3. Reguły transferu reguł

A.3.1. Usuwanie „nie” ze zdania

nie traktowali zbyt poważnie
traktowali zbyt poważnie
nie traktowali zbyt poważnie

nie tylko nie uzna
nie tylko uzna
nie tylko nie uzna

Takie rzeczy nie mijają bez śladu.
Takie rzeczy mijają bez śladu.
Takie rzeczy nie mijają bez śladu.

A.3.2. Normalizacja adresów elektronicznych

1. Adresy internetowe

www.wp.pl
www.wp.pl
wu wu wu kropka wu pe kropka pe el

http://poradnia.pwn.pl/lista.php?szukaj=zapiszemy&kat=1&od=10
http://poradnia.pwn.pl/lista.php?szukaj=zapiszemy&kat=1&od=10
ha te te pe dwukropek ukośnik ukośnik poradnia kropka pe wu en kropka pe el ukośnik lista kropka pe ha pe znak zapytania szukaj równa się zapiszemy ampersand ka a te równa się jeden ampersand o de równa się jeden zero

2. Adresy mailowe

m.wieczorek87@gmail.com
m.wieczorek87@gmail.com
em kropka wieczorek osiem siedem małpa gmail kropka kom

A.3.3. Normalizacja ciągów nieleksykalnych

Numery rejestracyjne pojazdów

samochód o rejestracji CMGG957
samochód o rejestracji CMGG957
samochód o rejestracji ce em gje gje dziewięć pięć siedem

PO8S424
PO8S424
PE O OSIEM ES CZTERY DWA CZTERY

Nazwy obiektów astronomicznych

planetoida (35396) 1997 XF11
planetoida (trzydzieści pięć tysięcy trzysta dziewięćdziesiąt sześć) tysiąc dziewięćset dziewięćdziesiąt siedem XF11
planetoida (trzydzieści pięć tysięcy trzysta dziewięćdziesiąt sześć) tysiąc dziewięćset dziewięćdziesiąt siedem iks ef jeden jeden

Nazwy seryjne sprzętu

numer seryjny drukarki PSC-1400
numer seryjny drukarki PSC-1400
numer seryjny drukarki pe es ce myślnik jeden cztery zero zero

A.4. Słownik polsko-polski

A.4.1. Normalizacja liczb rzymskich

czyta rozdział III
czyta rozdział III
czyta rozdział trzeci

tom II, część IV
tom II, część IV
tom drugi, część czwarta

od edycji XX Tańca z Gwiazdami
od edycji XX Tańca z Gwiazdami
od edycji dwudziestej Tańca z Gwiazdami

przepisał się do grupy XII
przepisał się do grupy XII
przepisał się do grupy dwunastej

na II piętrze
na II piętrze
na drugim piętrze

XII wiek p.n.e
XII wiek p.n.e
Dwunasty wiek przed naszą erą

w czasie II wojny światowej
w czasie II wojny światowej
w czasie drugiej wojny światowej

Bibliografia

[1] Encyklopedia pwn. http://encyklopedia.pwn.pl/haslo/3982133/synteza-mowy.html. 6

[2] Encyklopedia pwn. http://encyklopedia.pwn.pl/haslo/3948277/normalizacja.html. 12

[3] Wikipedia. http://pl.wikipedia.org/wiki/DSP. 16

[4] Wikipedia. http://en.wikipedia.org/wiki/Named_entity_recognition. 16

[5] Wikipedia. http://pl.wikipedia.org/wiki/T\OT4\lumaczenie_automatyczne. 18

[6] Ustawa z dnia 5 listopada 2009 r. o zmianie ustawy - kodeks karny, ustawy - kodeks postępowania karnego, ustawy - kodeks karny wykonawczy, ustawy - kodeks karny skarbowy oraz niektórych innych ustaw. http://isap.sejm.gov.pl/DetailsServlet?id=WDU20092061589, 2009. 15

[7] A. Hunt D.C. Burnett, M. R. Walker. Speech synthesis markup language (ssml). http://www.w3.org/TR/speech-synthesis/, 2004. 8

[8] T. Dutoit. A short introduction to text-to-speech synthesis.

[9] A. Wagner F. Graliński, K. Jassem. Text normalization as a special case of machine translation.

[10] M. Marcińczuk F. Graliński, K. Jassem. An environment for named entity recorgnition and translation. *Proceedings of the 13th Annual Conference of the EAMT*, 2009. str. 88 - 95. 17

[11] G. Kouroupertroglou G. Xydas, G.Karberis. Text normalization for the pronunciation of non-standard words in an inflected lnguage. 2004.

[12] M. Hanlon. Stephen hawking chooses a new voice. *Gizmag Emerging Technology Magazine*. 11

[13] W. Monjer L. Dybkjaer, H. Helmsen. *Evaluation of Text and Speech systems*. 2007.

[14] F. Graliński M. Marcińczuk. Specyfikacja modułu do rozpoznawania i tłumaczenia nazw własnych. 2009.

[15] M. Riley M. Mohri, F. Pereira. Speech recognition with weighted finite-state transducers. http://www.cs.nyu.edu/mohri/pub/hbka.pdf. 13

Spis tabel

Spis rysunków