Polsko-Japońska Wyższa Szkoła Technik Komputerowych

Katedra Multimediów

Michał Wójtowski
Nr albumu s1773

Segmentacja akustycznej bazy językowej na potrzeby realizacji korpusowej syntezy mowy w systemie Festival

Praca magisterska
napisana pod kierunkiem
prof. dra hab. Krzysztofa Maraska

Konsultant: mgr inż. Krzysztof Szklanny

Warszawa, grudzień 2007

Spis Treści

WSTĘP ... **3**

1. WPROWADZENIE ... **6**

1.1. JEDNOSTKI AKUSTYCZNE ... 9
1.2. TRANSKRYPCJA FONETYCZNA ... 12
1.3. JĘZYKOWA BAZA DANYCH ... 17
1.4. SEGMENTACJA SYGNAŁU MOWY ... 18

2. SYNTEZA MOWY I JEJ RODZAJE ... **23**

2.1. FORMANTOWA SYNTEZA MOWY ... 25
2.2. ARTYKULACYJNA SYNTEZA MOWY ... 26
2.3. KONKATENACYJNA SYNTEZA MOWY ... 27
2.4. KORPUSOWA SYNTEZA MOWY ... 29
2.5. STATYSTYCZNA SYNTEZA MOWY (HTS) ... 29

3. ROZPOZNAWANIE MOWY ... **31**

3.1. ANALIZA ORAZ PARAMETRYZACJA SYGNAŁU MOWY (EKSTRAKCJA CECH) ... 34
3.1.1. Analiza spektralna (widmowa) ... *36*
3.1.2. Analiza cepstralna (homomorficzna) ... *41*
3.2. METODY ROZPOZNAWANIA MOWY ... 42
3.2.1. Metody statystyczne (HMM) ... *43*
3.2.2. Sieci neuronowe (ANN) ... *51*
3.2.3. Hybrydy (NN-HMM) ... *53*

4. SEGMENTACJA AKUSTYCZNEJ BAZY JĘZYKOWEJ ... **55**

4.1. BAZA JĘZYKOWA (KORPUS) ... 56
4.2. NAGRANIA ... 59
4.3. AUTOMATYCZNA SEGMENTACJA NAGRAŃ (ALIGNER, HTK) ... 60
4.3.1. Wybór modeli HMM oraz jednostki akustycznej ... *63*
4.4. KOREKTA AUTOMATYCZNEJ SEGMENTACJI ... 70
4.4.1. Ręczna korekta błędów automatycznej segmentacji ... *74*
4.4.2. Opracowanie skryptu korygującego oraz weryfikacja jego działania ... *80*
4.4.3. Zakłócenia sieci elektrycznej (AC) ... *84*
4.4.4. Wstępna weryfikacja segmentacji w testowym syntezatorze ... *85*
4.5. DOSTOSOWANIE BAZY AKUSTYCZNEJ ORAZ OSTATNIE KOREKTY ... 87
4.6. TESTOWANIE (GENEROWANIE ZDAŃ TESTOWYCH) ... 95

5. ZAKOŃCZENIE ... **98**

PODZIĘKOWANIA ... **101**

BIBLIOGRAFIA ... **102**

ZAŁĄCZNIK A. KOD ŹRÓDŁOWY SKRYPTU KORYGUJĄCEGO ... **104**

ZAŁĄCZNIK B. SPIS RYSUNKÓW I TABEL ... **109**

ZAŁĄCZNIK C. ZAWARTOŚĆ PŁYTY ... **110**

Wstęp

Niniejsze opracowanie przedstawia proces segmentacji nagrań z wykorzystaniem mechanizmów automatycznego rozpoznawania mowy, w kontekście powstawania akustycznej bazy językowej. Przedmiotem pracy było stworzenie posegmentowanej i dopasowanej w dziedzinie czasu bazy akustycznej dla języka polskiego. Jej zastosowanie jest ściśle związane z korpusową syntezą mowy oraz tworzeniem systemów ASR (*Automatic Speech Recognition*). Ręczna segmentacja bazy akustycznej jest zadaniem bardzo czasochłonnym. Duże ilości materiału dźwiękowego wymagającego podzielenia na segmenty akustyczne sprawiły, iż konieczna stała się automatyzacja tego procesu. Pomimo, iż istnieją narzędzia realizujące to zadanie w sposób automatyczny, ich precyzja jest niewystarczająca z punktu widzenia wymogów zarówno syntezy, jak i rozpoznawania mowy. Wygenerowanie granic jednostek akustycznych z pomocą wspomnianych narzędzi wydaje się jednak stanowić bardzo dobry wstęp do prac nad dużą ilością nagrań, pozwalając zaoszczędzić sporo czasu. Ponadto założono, iż także korekta wyznaczonych ten sposób segmentów, może w pewnym stopniu zostać zautomatyzowana, umożliwiając przetworzenie dużej ilości materiału dźwiękowego w rozsądnym czasie.

Czynnikiem decydującym o podjęciu tematu, a także określającym kształt i zawartość opracowania, była konieczność dokonania segmentacji bazy akustycznej, przeznaczonej na potrzeby realizacji polskiego korpusowego syntezatora mowy w metasystemie Festival[1]. Celem niniejszej pracy stała się więc automatyczna segmentacja bazy akustycznej, będąca procesem wyznaczania granic wybranych jednostek akustycznych (segmentów) zarejestrowanej mowy. Dodatkowym celem było wykorzystanie do tego zadania mechanizmów automatycznego rozpoznawania mowy. Istotną przesłanką było także dostosowanie wspomnianej bazy akustycznej do wymogów systemu Festival oraz powstającego w PJWSTK syntezatora mowy. Wymogi stawiane przed projektem dotyczyły poprawności (w tym dokładności) wyznaczonych granic segmentów oraz ich transkrypcji fonetycznej, a także formatu danych. W związku z tym koniecznym okazało się opracowanie

[1] http://www.cstr.ed.ac.uk/projects/festival/, 12-2007

słownika transkrypcji fonetycznej wyrazów, dla których przyjęte reguły transkrypcji nie były poprawne.

Wyżej zaznaczona wieloaspektowość problematyki znalazła odzwierciedlenie w układzie pracy. Pierwszy rozdział poświęcono zagadnieniom wprowadzającym, dotyczącym analizy mowy, transkrypcji fonetycznej, językowych baz danych, a przede wszystkim problemu segmentacji sygnału mowy. W drugim rozdziale, przedstawiona została wiedza związana z generowaniem mowy ludzkiej w sposób sztuczny, czyli jej syntezą. Opisano ogólną budowę syntezatorów mowy oraz różne sposoby generowania mowy syntetycznej, w tym metodę korpusową (*unit selection*), z którą związana jest baza akustyczna poddana procesowi segmentacji w ramach praktycznej części niniejszej pracy. Natomiast w trzecim rozdziale, poddano szczegółowej analizie istotne z punktu widzenia segmentacji kwestie z zakresu automatycznego rozpoznawania mowy (ASR – *Automatic Speech Recognition*). Dotyczą one parametryzacji sygnału, jego analizy (widmowej oraz cepstralnej), a także różnych metod rozpoznawania mowy. Istotnym elementem tego rozdziału jest analiza widmowa (spektralna), stanowiąca także fundament analizy sygnału mowy w procesie segmentacji. Główny nacisk położono na przedstawienie statystycznych metod rozpoznawania mowy, wykorzystujących ukryte (niejawne) modele Markowa (HMM – *Hidden Markov Model*), gdyż na nich opierają się narzędzia wykorzystane w praktycznej części pracy. Czwarty rozdział zawiera opis przebiegu prac nad segmentacją akustycznej bazy językowej, a także dostosowaniem jej do wymogów systemu Festival oraz syntezatora mowy, w którym zostanie wykorzystana. W rozdziale tym, na wstępie przedstawiono cele i założenia tej części projektu. Następnie omówiono etapy powstawania wykorzystywanej bazy akustycznej, a konkretnie budowę korpusu językowego oraz przeprowadzenie na jego podstawie nagrań. Kolejnym elementem tej części pracy jest opis sposobu wygenerowania wstępnej automatycznej segmentacji oraz wykorzystanych w tym celu narzędzi, a także modeli HMM, wraz z procesem ich testowania i wyboru spośród udostępnionych wersji. Następnie przedstawiono metodę automatycznego wyszukiwania błędów oraz kilkuetapowy proces korekty błędów automatycznej segmentacji. Etapy tego procesu to kolejno ręczna korekta segmentacji dotycząca większości nagrań, opracowanie i realizacja skryptu korygującego, weryfikacja wraz z korektą błędów wprowadzonych przez skrypt oraz filtracja zakłóceń sieci elektrycznej (DC) występujących w stosunkowo niewielkiej części nagrań. Następnie omówiono wstępną weryfikację

poprawności segmentacji w testowym syntezatorze. Ponadto, zaprezentowany został przebieg dostosowywania posegmentowanej bazy akustycznej do wymogów systemu Festival oraz powstającego syntezatora, a także proces wprowadzania końcowych poprawek. Na tym etapie przedstawiono także budowę słownika transkrypcji fonetycznej dla występujących w nagraniach wyrazów, które nie podlegały przyjętym regułom fonologicznym. Ostatnią część rozdziału poświęcono procesowi testowania poprawności segmentacji we wczesnej wersji powstającego syntezatora, na podstawie wygenerowanych 100 zdań testowych oraz charakterystyce korpusu testowego. W zakończeniu przedstawiono podsumowanie uzyskanych wyników. W uzupełnieniu są załączniki ilustrujące praktyczną część pracy oraz bibliografia prezentująca wybór źródeł.

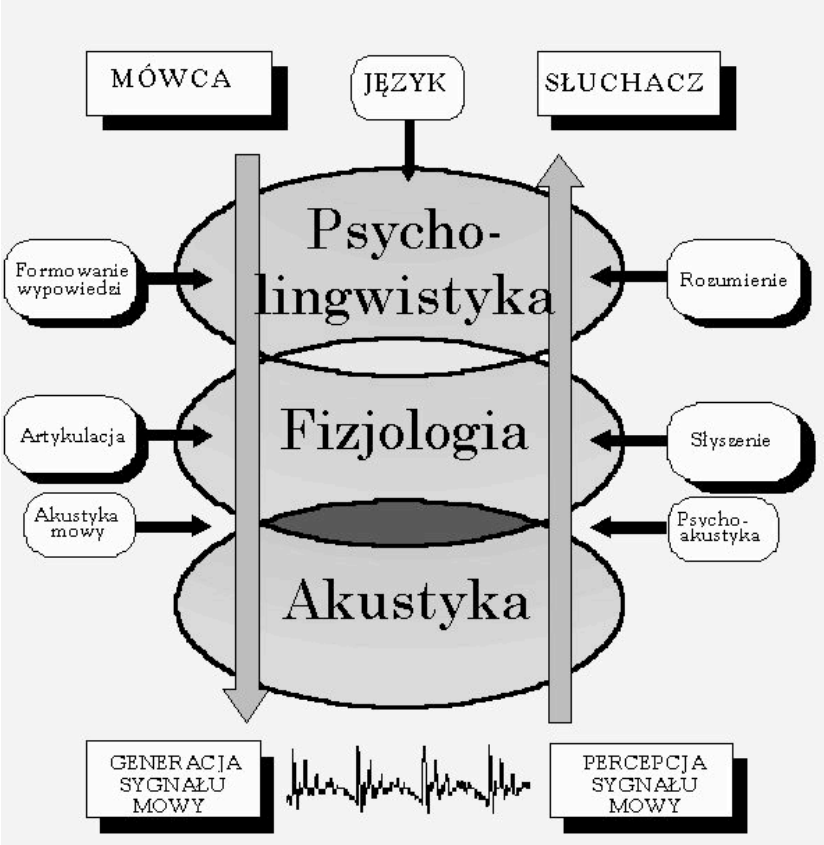
1. Wprowadzenie

”Mowa jest jednym z wielu sposobów przekazywania informacji. Specyfiką mowy jest to, że ma postać dźwiękową. Jest zazwyczaj kodowana w postaci ciągu dźwięków o określonych charakterystykach. Kod jest specyficzny dla danego języka, co powoduje, że każdy język ma określony dla siebie zbiór dźwięków mowy.” (w/g Gubrynowicz [1])

Badanie oraz analiza mowy, a ściślej fal dźwiękowych, jakie generuje ludzki narząd mowy w celu komunikacji z otoczeniem, jest domeną fonetyki akustycznej. Jest to techniczny dział nauki o języku, jaką jest lingwistyka. Samą fonetykę można podzielić na działy według obszarów badań, które niejednokrotnie przenikają się z innymi dziedzinami wiedzy jak np. fizjologią czy akustyką:

- fonetyka akustyczna – zajmująca się cechami fizycznymi (akustycznymi) dźwięków mowy,
- fonetyka artykulacyjna – zajmująca się sposobem wytwarzania dźwięków przez narządy mowy, czyli artykulacją,
- fonetyka audytywna - zajmująca się analizą percepcji tychże dźwięków,
- fonetyka psycholingwistyczna (percepcyjna) – zajmująca się reakcjami, jakie owe dźwięki wywołują w psychice człowieka,
- do tego należy dodać także inne cechy foniczne związane z głosem takie jak akcent, intonacja (prozodyczne) z pogranicza fonetyki i fonologii.

Schemat obrazujący dziedziny wiedzy związane z komunikacją werbalną przedstawia rysunek 1.1.



Rysunek 1.1: Dziedziny wiedzy związane z mową (w/g [1]).

Powstający w procesie mowy ciąg dźwięków niesie ze sobą wiele informacji, wśród których można wyróżnić:

- Informacje lingwistyczne,
- Informacje artykulacyjne,
- Informacje emocjonalne,
- Informacje osobnicze (społeczne, kulturowe, informacje o zaburzeniach).

Według podziału, jakiego dokonał J. Laver w 1991 roku, wyróżniamy trzy poziomy komunikacji w przypadku mowy:

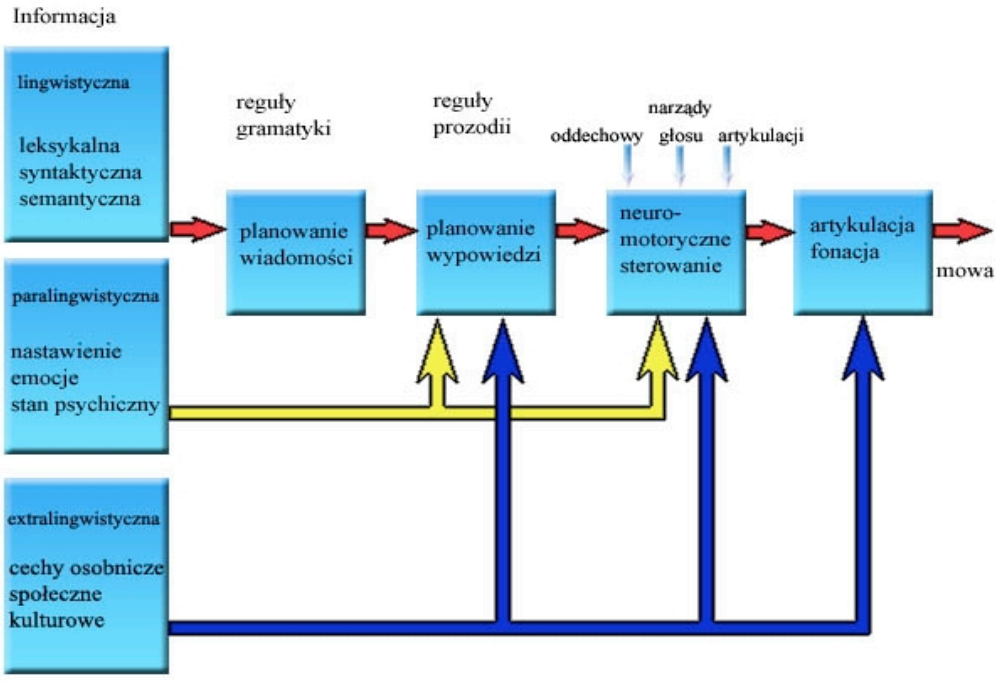
- Lingwistyczny
- Paralingwistyczny
- Extralingwistyczny

Poziom lingwistyczny obejmuje treść wypowiedzi wraz ze składnią leksykalną, syntaktyczną, a także semantyczną danego języka. Oprócz zagadnień gramatyki dotyczy również fonetycznej reprezentacji wypowiedzi.

Na poziomie paralingwistycznym (pozawerbalnym) przekazywane są informacje dotyczące nastawienia mówcy, jego stanu emocjonalnego czy psychicznego.

Extralingwistyczna część komunikacji werbalnej dotyczy przede wszystkim cech osobniczych mówcy (zarówno fizycznych jak i fizjologicznych), czyli głosu, płci, wieku, zaburzeń mowy, a nawet nawyków czy uwarunkowań społecznych, bądź kulturowych.

Omówione poziomy komunikacji werbalnej przedstawiono na rysunku 1.2.



Rysunek 1.2: Poziomy komunikacji werbalnej (częściowo w/g [19]).

1.1. Jednostki akustyczne

Analiza mowy wymaga, by w sygnale będącym ciągłą sekwencją dźwięków, wyodrębnić charakterystyczne, stosunkowo niewielkie segmenty, czyli zróżnicowane jednostki akustyczne. Istnieje kilka takich grup jednostek wykorzystywanych najczęściej w analizie mowy na potrzeby segmentacji, syntezy i rozpoznawania:

- fonemy (głoski),
- alofony,
- difony,
- trifony,
- półsylaby,
- sylaby.

Jak pisze Wierzchowska [9]: *„W strukturze postaci dźwiękowej języka polskiego szczególna rola przypada fonemom, które są najkrótszymi elementami dźwiękowymi pełniącymi funkcję dystynktywną.”* Innymi słowy fonemy są najprostszymi elementami dźwiękowymi mowy rozróżnianymi słuchowo przez użytkowników danego języka i odróżniającymi od siebie formy językowe mające różne wartości semantyczne. Inwentarz fonemów, zarówno jak i każdej innej jednostki akustycznej zależny jest od języka. W przedkładanej pracy przyjęto, iż w języku polskim występuje 37 fonemów (alfabet fonetyczny SAMPA), co odpowiada liczbie rozróżnianych głosek, a każdy z 37 fonemów jest zbiorem cech dystynktywnych odpowiadającej mu głoski. Należy jednak zaznaczyć, że spotyka się różne poglądy na liczbę rozróżnianych w języku polskim fonemów, sposoby ich wyznaczania, jak i na ilość występujących głosek. Dlatego w innych opracowaniach liczby te mogą być różne od przedstawionej w niniejszej pracy. Przykładowo cytując Wierzchowską [9]: „*Stosując kazańsko-praską procedurę wyróżniania fonemów, dla języka polskiego ustala się inwentarz fonemów obejmujący 41 pozycji.*” Przytoczona odmienna liczba fonemów wynika z faktu, iż kazańsko-praska procedura nie wyróżnia głoski „i” jako osobnego fonemu w przypadku jego występowania po spółgłosce, wyróżnia natomiast jako osobne fonemy

głoski zmiękczone przez następującą po nich głoskę „i” (np. zbitka głosek „pi” uznawana jest za osobny fonem - /p’/).

Zgodnie ze stanowiskiem Wierzchowskiej [9] należy zauważyć, iż: *„Poszczególne realizacje tych samych głosek, nawet wymawianych w takim samym kontekście fonetycznym, w tych samych formach wyrazowych, nie są nigdy zupełnie takie same; w różnych wykonaniach tych samych ruchów artykulacyjnych obserwuje się zawsze pewien naturalny rozrzut charakteryzujący wszelkie, najbardziej nawet zautomatyzowane czynności człowieka.”* Oznacza to, iż fonem stanowi pewne uogólnienie dostatecznie podobnych dźwięków, będących jego realizacjami. Konkretne realizacje fonemów także mogą być rozróżniane i nazywa się je alofonami.

Difonem (*diphone*) nazywa się przejście (tranzjent) pomiędzy dwoma fonemami. Rozpoczyna się w połowie jednego fonemu (tzw. części stacjonarnej), a kończy w połowie następnego (także w części stacjonarnej). Difony są wygodną jednostką w przypadku syntezy mowy, dzięki nim miejsca połączeń fragmentów mowy znajdują się w części stacjonarnej głosek, która nie ulega „zniekształceniom” związanym z płynnymi ruchami narządów artykulacyjnych (koartykulacją). W wyniku koartykulacji[2] wymawiane fonemy płynnie przechodzą jeden w drugi, a ich koniec i początek, często upodobniają się odpowiednio do poprzedniego i następnego fonemu. Dlatego fonemy wycięte z nagrań i umieszczone w innym kontekście, często wnoszą zniekształcenia podyktowane upodobnieniami na ich krańcach, wynikającymi z ich oryginalnego otoczenia. Połączenia difonów są bardziej naturalne i wymagają mniejszych modyfikacji źródłowego sygnału, niż w przypadku fonemów (o ile system przewiduje modyfikacje), co przekłada się na bardziej naturalne brzmienie syntezowanej mowy. Ponadto, rozmiar bazy danych jest stosunkowo niewielki (1443 difony dla języka polskiego). Należy zaznaczyć, iż granice difonów są dość łatwe do zlokalizowania, gdyż wspomniane części stacjonarne głosek, w których znajdują się owe granice, ulegają w znacznie mniejszym stopniu koartykulacji i są najbardziej charakterystycznymi elementami głosek. Dodatkowo granice difonów mogą być dość arbitralnie wyznaczane, w stosunkowo szerokim przedziale czasowym, np. 20% czasu trwania części stacjonarnej. Dlatego

[2] „Nakładanie się na siebie ruchów artykulacyjnych właściwych sąsiadującym ze sobą głoskom.” Zgodnie z Wierzchowska [9]

segmentacja nagrań z wykorzystaniem difonów jest łatwiejsza niż w przypadku fonemów, dla których niejednokrotnie trudno, czy wręcz niemożliwe jest dokładnie określić początek i koniec danej głoski (np. dla spółgłosek płynnych, takich jak /j/).

Obok fonemów i difonów istotną jednostką akustyczną są także trifony (*triphone*). Trifonami nazywa się fonemy wraz z określonym lewym i prawym kontekstem (sąsiedztwem). Oznacza to, iż trifony dzielą fonemy na grupy alofonów ze względu na ich lewe i prawe sąsiedztwo, modelując w ten sposób zależność fonemów od ich kontekstu (koartykulację). Dlatego trifony bardzo dobrze nadają się do syntezy mowy i pozwalają uzyskać dość naturalne brzmienie. Warto zauważyć, że choć trifony stanowią dobrą alternatywę dla difonów, wymagają oczywiście znacznie większej bazy akustycznej.

Definicja sylaby (zgłoski) jest w fonetyce zagadnieniem spornym. Jak pisze Wierzchowska [9] *„Problem sylaby rozpatrywany bywa bądź ze stanowiska artykulacyjnego, bądź ze stanowiska audytywnego, bądź w obu tych aspektach jednocześnie.”* Poniżej przedstawiono cytat tej samej pozycji definiujący sylabę ze stanowiska artykulacyjnego i audytywnego (akustycznego) jednocześnie, gdyż ta właśnie definicja wydaje się najpełniejsza. Jest to definicja tzw. sylaby fonetycznej.

„Fonetycy, którzy opisują sylabę w obu aspektach, tj. i w aspekcie artykulacyjnym, i w aspekcie akustycznym, kładą nacisk na jednoczesność zmian w układzie narządów mowy, ciśnieniu powietrza w tchawicy oraz donośności dźwięków (postrzegalności słuchowej). Ośrodkami sylab są te odcinki ciągu mownego, na które przypada maksymalne rozwarcie kanału głosowego i maksymalna donośność; na pograniczach i na krańcach sylab donośność dźwięków mowy jest najniższa, stopień zaś zbliżenia narządów mowy - największy.” [9]
Podobnie kwestię sylaby fonetycznej ujmuje także Roudet [11].

Przytoczone jednostki akustyczne są fundamentalne dla opisanych w kolejnych rozdziałach pracy zagadnień: segmentacji nagrań, rozpoznawania mowy oraz jej konkatenacyjnej syntezy, w tym w najefektywniejszej jej wersji, to jest metody korpusowej.

1.2. Transkrypcja fonetyczna

Opis sygnału mowy wymaga nadania etykiet poszczególnym jego segmentom. Tekst ortograficzny często nie określa jednoznacznie wymowy i nie jest dobrym sposobem jej reprezentacji. Te same znaki ortograficzne mogą odpowiadać różnym dźwiękom, podczas gdy ten sam dźwięk może odpowiadać różnym znakom. Przykładem może być litera „w" w wyrazach „woda" i „wtedy", w pierwszym wypadku czytana jest jako „w", w drugim jako „f". Inne przykłady to litery „u" i „ł" w wyrazach „euro" i „dług", obie czytane jako „ł" czy mniej oczywiste różnice w wymowie litery „n", np. w wyrazach „mina" i „bank". Dlatego opracowany został szeroko stosowany międzynarodowy alfabet fonetyczny IPA (*International Phonetic Alphabet*), zawierający reprezentację dźwięków mowy wszystkich języków. Mankamentem kodu IPA jest fakt, iż zawiera on znaki diakrytyczne niestosowane w standardowym kodzie ASCII. Wygodniejszy w przetwarzaniu komputerowym jest alfabet SAMPA (*Speech Assessment Methods Phonetic Alphabet*) w pełni kompatybilny z ASCII. Warto podkreślić, iż kod ten stanowi odwzorowanie alfabetu IPA na znaki ASCII, ale nie jest tak uniwersalny. Opracowywane równolegle były i wciąż są niezależne notacje dla różnych języków. Proces przekształcania tekstu ortograficznego na kod fonetyczny, opiera się o określone reguły i nazywa się transkrypcją fonetyczną. Opanowanie reguł transkrypcji fonetycznej w kodzie SAMPA dla języka polskiego[3] jest niezbędne w procesie segmentacji, będącej celem niniejszego projektu. Poniżej przedstawiono tabele ogólnych odwzorowań znaków ortograficznych (odpowiadających im głosek) na kod SAMPA dla języka polskiego. Dodatkowo opisane zostały reguły precyzujące odstępstwa i wyjątki specyficzne dla języka, w głównej mierze zależne od otoczenia danego znaku.

Poniższa tabela 1.2.1 przedstawia reprezentację fonetyczną 8 polskich samogłosek. Litery „ó" i „u" nie mają różnic w wymowie, dlatego posiadają wspólną reprezentację. Znak „~"oznacza nazalizację (unosowienie), czyli dodatkową nosową artykulację danego fonemu.

[3] www.phon.ucl.ac.uk/home/sampa/polish.htm, 12-2007

Symbol ortograficzny	Symbol SAMPA	Np. w wyrazie (transkr. SAMPA)
i	i	bit /bit/
y	I	byk /bIk/
e	e	bek /bek/
a	a	bak /bak/
o	o	bok /bok/
u	u	buk /buk/
ę	e~	tę /te~/
ą	o~	tą /to~/

Tabela 1.2.1: Transkrypcja fonetyczna samogłosek (w/g [1]).

Kolejne tabele (1.2.2, 1.2.3, 1.2.4 oraz 1.2.5) przedstawiają fonetyczną reprezentację spółgłosek. Składa się ona z 29 fonemów. Znak ’ symbolizuje zmiękczenie spółgłoski twardej (palatalizacja).

Symbol ortograficzny	Symbol SAMPA	Np. w wyrazie (transkr. SAMPA)
f	f	fakt /fakt/
w	v	waga /vaga/
s	s	syk /sIk/
z	z	zbir /zbir/
sz	S	szyk /SIk/
ż	Z	żyto /ZIto/
ś	s’	świt /s’fit/
ź	z’	źle /z’le/
h, ch	x	hak /xak/

Tabela 1.2.2: Transkrypcja fonetyczna spółgłosek trących (w/g [1]).

Symbol ortograficzny	Symbol SAMPA	Np. w wyrazie (transkr. SAMPA)
p	p	puk /puk/
b	b	bat /bat/
t	t	test /test/
d	d	dym /dIm/
k	k	kat /kat/
g	g	gen /gen/

Tabela 1.2.3: Transkrypcja fonetyczna spółgłosek zwartych, czyli plozyjnych (w/g [1]).

Symbol ortograficzny	Symbol SAMPA	Np. w wyrazie (transkr. SAMPA)
m	m	mysz /mIS/
n	n	nasz /naS/
ń	n’	koń /kon’/
n(k,g)	N	bank /baNk/*
ł	w	łyk /wIk/
j	j	jak /jak/
l	l	luk /luk/
r	r	ryk /rIk/

* Spółgłoska nosowa /N/ występuje w języku polskim tylko przed spółgłoskami /k, g/.

Tabela 1.2.4: Transkrypcja spółgłosek zwanych sonorantami lub rezonantami (w/g [1]).

Symbol ortograficzny	Symbol SAMPA	Np. w wyrazie (transkr. SAMPA)
c	ts	coś /tsos'/
dz	dz	dzwon /dzvon/
cz	tS	czapka /tSapka/
dż	dZ	dżem /dZem/
ć	ts'	ćwicz /ts'fitS/
dź	dz'	dźwiga /dz'viga/

Tabela 1.2.5: Transkrypcja fonetyczna spółgłosek zwarto-trących (w/g [1]).

Powyższe tabele określają jedynie odwzorowania symboli i wymagają uściślenia dodatkowymi regułami, które przedstawiono poniżej (zgodnie z Gubrynowicz [1]):

Literę „i" przed spółgłoską notuje się jako /i/, natomiast przed samogłoską jako:

- /j/ po spółgłoskach zwartych, trących /f, v, x/, nosowej /m/ i głoskach /l, r/
- /i/ na końcu wyrazu

Niniejsze grupy spółgłosek z następującą po nich samogłoską /i/ odpowiadają fonemom:

- „si" -> /s'/
- „ci" -> /ts'/
- „zi" -> /z'/
- „dzi" -> /dz'/
- „ni" -> /n'/ (wyjątek „Dania" -> /dan'ja/)

Podwójne „ii" po spółgłoskach zwartych, trących /f, v, x/, nosowej /m/ i głoskach /l, r/ wymawiane i notowane są jako /ji/

Samogłoski nosowe „ę, ą” odpowiadają fonemom:

- /e~, o~/ na końcu wyrazu
- /em, om/ przed /p, b/
- /en, on/ przed /t, d, ts, tS, dz, dZ/
- /en’, on’/ przed /ts’, dz’/
- /eN, oN/ przed /k, g/
- /e, o/ przed /l, w/

Głoski zwarte /b, d, g/, zwarto-trące /dz, dz’, dZ/ i trące /v, z, z’, Z/ wymówione przed głoskami bezdźwięcznymi czy przerwą także stają się bezdźwięcznymi. Ich wymowa i notacja fonetyczna jest identyczna, jak ich bezdźwięcznych odpowiedników, czyli /p, t, k/, /ts, ts’, tS/ czy /f, s, s’, S/. To samo występuje u zbiegu wyrazów wymówionych bez przerwy.

O ubezdźwięcznieniu lub udźwięcznieniu całej sekwencji powyższych spółgłosek o różnym typie pobudzenia, decyduje w zasadzie ostatnia w sekwencji głoska („liczba” -> /lidZba/, „rzadszy” -> /Zat_SI/). Od powyższej zasady jest wyjątek, gdy przed literą „w” lub sekwencją „rz” stoi głoska bezdźwięczna. Cała sekwencja staje się wtedy bezdźwięczna, np. „kwiat” -> /kfjat/, „szwaczka” -> /SfatSka/. Spółgłoski bezdźwięczne przed końcówką czasownikową „my” także pozostają bezdźwięczne, np. „kupmy” -> /kupmy/

Istnieją nieregularności w wymowie sekwencji „trz”, „drz”, „dż”, „dz” w obrębie wyrazu, np. „trzech” -> /tSSex/, „wodze” -> /vodze/, „odzew” -> /od_zef/. Spółgłoski „j”, „l”, „ł” (przymknięte) wymówione w środku dłuższych sekwencji spółgłoskowych, wymawiane są tak słabo, że często ulegają całkowitej redukcji, a ich otoczenie najczęściej staje się bezdźwięczne. Np. „jabłko” -> /japko/, „rzemieślnik” -> /Zemjes’n’ik/.

Omówioną reprezentację wykorzystano podczas segmentacji. Wraz z kolejnymi etapami prac, wprowadzano jednak pewne modyfikacje do przedstawionej transkrypcji. Związane były one z wymogami syntezatora i systemu Festival, a także wymową konkretnego mówcy. Modyfikacje te przedstawiono przy okazji omawiania praktycznej części pracy.

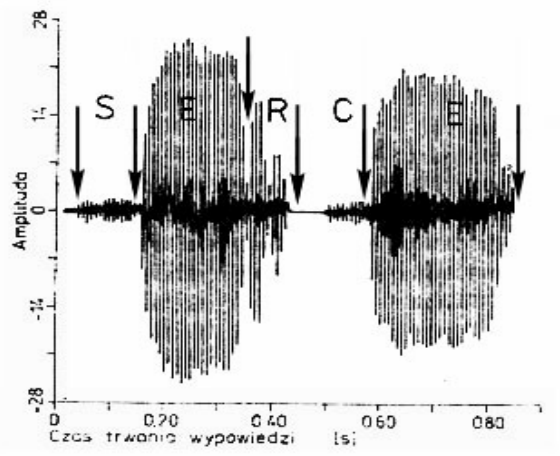
1.3. Językowa baza danych

Językowa baza danych jest uporządkowanym zbiorem różnych elementów i form językowych, wraz z ich naturalnym kontekstem. Bazy językowe występują zarówno w wersji tekstowej, jak i akustycznej, czyli w postaci zarejestrowanej mowy. Ich główne przeznaczenie to tzw. korpusowe metody przetwarzania języka i mowy. Zawartość, rozmiar, format oraz sposób tworzenia baz językowych uzależnione są od ich potencjalnych zastosowań. Przykładowo, zupełnie inna będzie baza przeznaczona do badań lingwistycznych danego języka (np. dotyczących częstości występowania form wyrazowych), w porównaniu z bazą zbudowaną na potrzeby systemu dialogowego dla informacji (rezerwacji) kolejowej. Rozmiar i zawartość bazy dobierane są tak, by była ona wystarczająco reprezentatywna dla danego języka lub jego modelowanego wycinka (np. wypowiedzi polityków, komentatorów sportowych czy komendy umożliwiające sterowanie i dialog z zabawką).

Tekstowe bazy językowe najczęściej wykorzystywane są w systemach automatycznego tłumaczenia tekstów na inny język oraz do badań lingwistycznych. Bazy akustyczne początkowo powstawały w celu utworzenia wspólnego zbioru danych testowych dla systemów rozpoznawania mowy. Obecnie są one niezbędne do badań fonetycznych poszczególnych języków, projektowania, uczenia i weryfikacji systemów rozpoznawania mowy i mówców, a także w konkatenacyjnych metodach syntezy mowy. Bazy akustyczne, poza zarejestrowanymi próbkami mowy, muszą zawierać także opis warunków technicznych nagrań oraz ich anotację. Powinna ona zawierać co najmniej ortograficzny zapis każdego nagrania i słownik transkrypcji fonetycznej zarejestrowanych wyrazów, zwykle zawiera też opis zdarzeń akustycznych (np. hałasów, czy nieartykułowanych dźwięków wydawanych przez mówce). W większości zastosowań znacznie dogodniejszą formą od słownika jest transkrypcja fonetyczna każdego nagrania osobno, wraz z oznaczonymi granicami każdego fonemu (lub innej jednostki akustycznej). Proces wyznaczania granic jednostek akustycznych, nazywany segmentacją nagrań, przedstawiony został w następnym rozdziale.

1.4. Segmentacja sygnału mowy

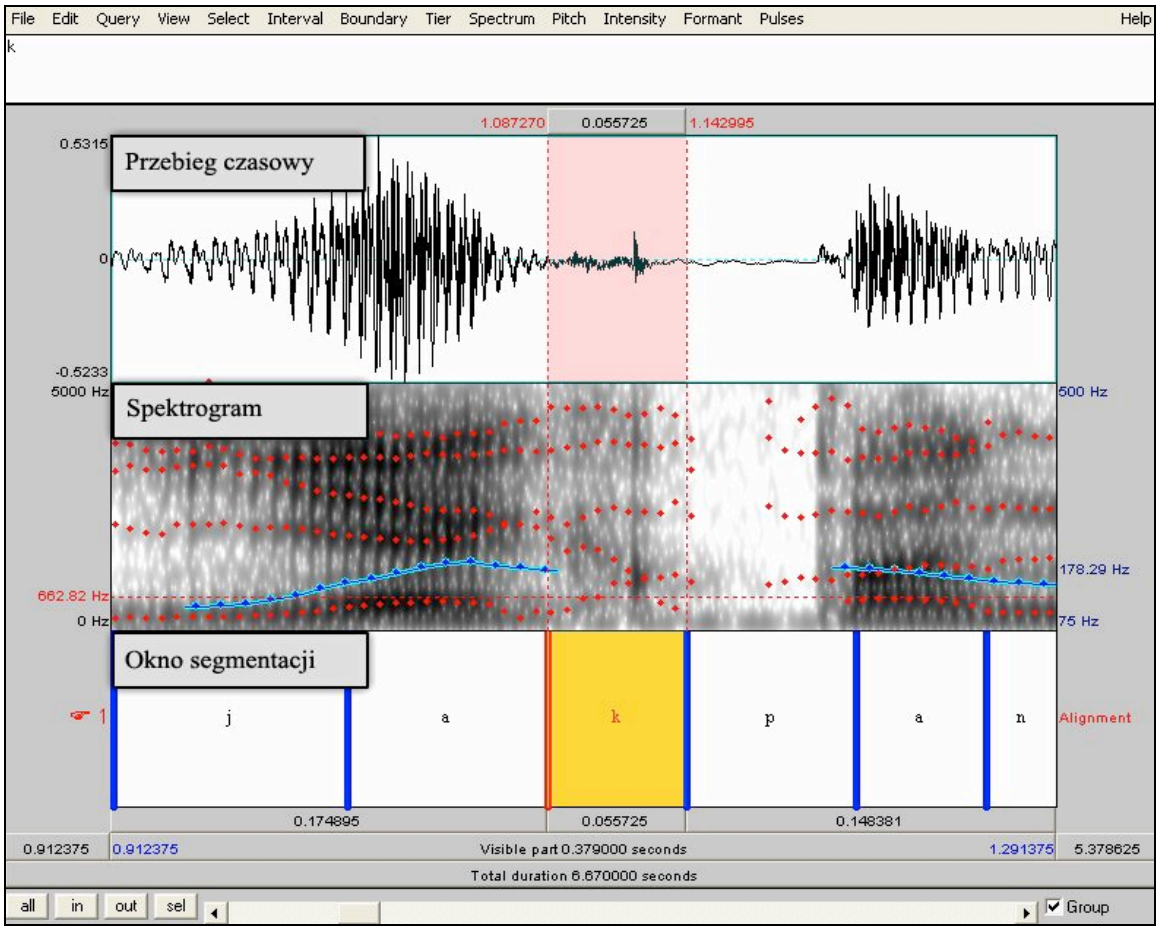
Segmentacja zarejestrowanej mowy, będąca tematem niniejszej pracy, ma na celu precyzyjne określenie granic (początku i końca) występujących w sygnale jednostek akustycznych danego typu. Dodatkowo, proces ten umożliwia szczegółową weryfikację wszystkich nagrań, ich kompletności, a także oznaczenie błędnych wypowiedzi czy zakłóceń. Jest to niezbędny element opisu baz akustycznych, przeznaczonych na potrzeby uczenia systemów rozpoznawania mowy, lub do jej konkatenacyjnej syntezy (przede wszystkim metody korpusowej). Poprawne oznaczenie wspomnianych granic, w często i płynnie zmieniającym się sygnale mowy, jest czasochłonne i w wielu przypadkach niejednoznaczne. Dlatego dokładna segmentacja, oprócz odsłuchu, wymaga znajomości fonetyki, reprezentacji akustycznej głosek, pomocnych w tym zadaniu fizycznych parametrów dźwięku, a także praktyki w obróbce sygnału mowy. Na rysunku 1.4.1 przedstawiono przykładowy przebieg czasowo-amplitudowy wypowiedzi „serce". Za pomocą strzałek oznaczone są orientacyjne granice fonemów, pomiędzy nie wpisane są ortograficzne odpowiedniki każdego fonemu.



Rysunek 1.4.1: Wypowiedź „serce", z zaznaczonymi granicami fonemów (w/g [7]).

Podstawowe parametry dźwięku pomocne w procesie segmentacji to przebieg czasowo-amplitudowy oraz spektrogram, umożliwiający analizę formantową (widmową). Analiza widmowa sygnału przedstawiona została w rozdziale 3.1.1 wraz z krótkim opisem pojęcia formantu i spektrogramu. W skrócie, widmo jest to wykres przedstawiający rozkład amplitud

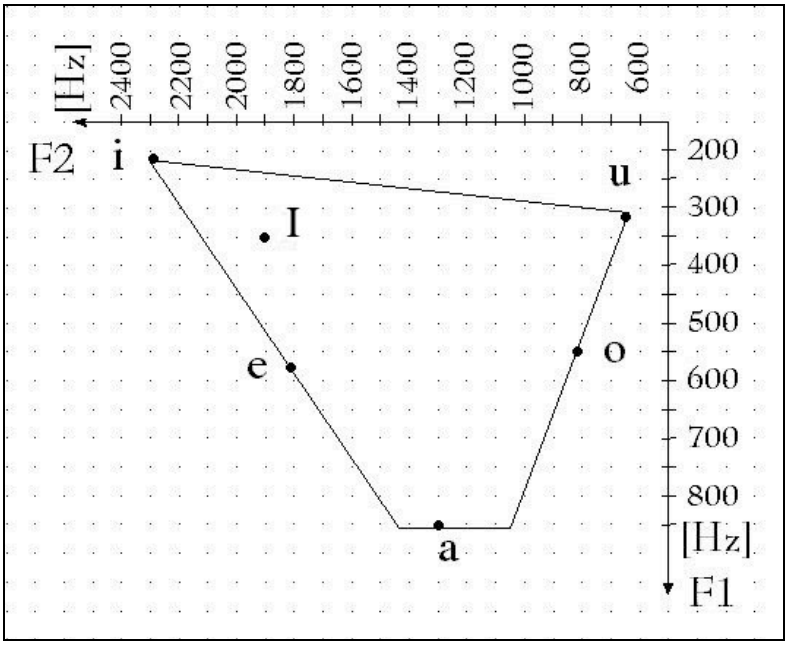
(energii) składowych częstotliwościowych niewielkiego fragmentu czasowego analizowanego sygnału (ramki). Formanty są to częstotliwości rezonansowe kanału głosowego, których estymatami są lokalne maksima obwiedni przekroju widma wyznaczonego dla zadanej chwili czasowej. Należy tutaj zauważyć, że wraz z ruchami narządów mowy zmieniają się wnęki rezonansowe, w związku z czym formanty pojawiają się, znikają, zmienia się ich liczba, rozmiar i położenie. Spektrogram jest natomiast graficzną reprezentacją zmian widma sygnału w czasie. Jest wizualną, trójwymiarową formą zapisu wypowiedzi, umożliwiającą śledzenie zachodzących w niej zmian (np. formantów). Warto zauważyć, iż pierwsza harmoniczna w widmie, nazywana „F0" lub „*Pitch*" reprezentuje ton krtaniowy, kształtujący melodię głosu oraz intonacje wypowiedzi. Powstaje on w wyniku modulacji strumienia powietrza wypływającego z płuc przez wiązadła głosowe. Zakres zmian tonu krtaniowego zależy od płci, wieku, oraz cech osobniczych mówcy.



Rysunek 1.4.2: Przebieg czasowy, spektrogram i okno segmentacji w programie Praat.

Rysunek 1.4.2 przedstawia przykładowe okno darmowego programu Praat[4] opracowanego na uniwersytecie w Amsterdamie, zawierające przebieg czasowy sygnału, spektrogram oraz część przeznaczoną dla segmentacji. Dodatkowo, program zaznaczył na spektrogramie połączonymi niebieskimi kropkami przebieg intonacji (F0, Pitch) oraz za pomocą czerwonych kropek - zmiany pierwszych pięciu formantów.

Trudno jest jednoznacznie sprecyzować uniwersalne reguły segmentacji. Różnorodność wymowy tych samych jednostek akustycznych sprawia, iż każdy przypadek należy rozpatrywać osobno. Poniżej przedstawiono dane i ogólne wytyczne pomocne w procesie identyfikacji granic fonemów. Nacisk położono na fonemy, gdyż łatwo na ich podstawie wyznaczyć granice innych jednostek, np. początek difonu /tI/ znajduje się w połowie fonemu /t/ (części stacjonarnej, którą jest zwarcie), a koniec w połowie fonemu /I/. Na rysunku 1.4.3 przedstawiono tzw. czworobok samogłosek, określający uśrednione położenie samogłosek na płaszczyźnie pierwszego i drugiego formantu (F1 i F2). Inną wersję, przedstawiającą obszary skupień samogłosek na tej płaszczyźnie, ilustruje rysunek 3.1.1, w rozdziale 3.1.



Rysunek 1.4.3: Czworobok samogłosek (w/g [1]).

[4] http://www.fon.hum.uva.nl/praat, 12-2007

Tabela 1.4.4 przedstawia krótką charakterystykę przebiegu czasowego oraz częstotliwościowego poszczególnych klas głosek.

Typ głoski	Przykład	Dziedzina czasu	Dziedzina częstotliwości
Samogłoski	/a/ /e/ /i/	- quasi-periodyczne - nośnik dużej energii	- od 3 do 6 widocznych formantów - wyraźne maksima intonacji (peak pitch) - główna energia sygnału zawiera się w niskich częstotliwościach
Wybuchowe dźwięczne	/b/ /d/ /g/	- nagły skok amplitudy	- formant w dolnych częstotliwościach - krótkotrwały skok energii we wszystkim pasmach częstotliwości
Wybuchowe bezdźwięczne	/p/ /t/ /k/	- nagły i wysoki skok amplitudy	- brak struktury formantów - energia skoncentrowana w wysokich częstotliwościach
Trące dźwięczne	/v/ /z/ /z'/	- szeroki quasi-niezmienny obszar - nośnik małej ilości energii	- formanty - pierwszy formant podobny do formantu samogłoskowego - energia skoncentrowana w wysokich częstotliwościach
Trące bezdźwięczne	/f/ /s/ /s'/	- charakterem podobna do szumu - nośnik małej ilości energii	- szerokie spektrum
Nosowe	/m/ /n/	- quasi-periodyczne - podobne do samogłosek - mniejsza ilość energii niż w samogłoskach	- minimum w zasięgu spektrum - formanty podobnie jak w samogłoskach
Płynne	/l/ /r/ /j/	- brak wyraźnego, niezmiennego początku -mało widoczne przejście do sąsiadującej samogłoski	- pierwszy formant o mniejszej częstotliwości

Tabela 1.4.4: Charakterystyka poszczególnych klas głosek (w/g [4]).

Podczas segmentacji bazy akustycznej na potrzeby syntezy mowy istotne jest, aby wyznaczone granice znajdowały się, jeśli to możliwe, na początku okresu krtaniowego oraz w przejściu sygnału przez zero (w niniejszej pracy przyjęto przejście z minusa na plus). W przeciwnym przypadku mogą pojawiać się trzaski w generowanej przez syntezator mowie.

Segmentacja dużej ilości materiału jest procesem bardzo czasochłonnym, wymagającym dużej uwagi. Dlatego w wielu laboratoriach powstały narzędzia umożliwiające automatyczne wyznaczanie granic wybranych jednostek akustycznych w zarejestrowanej mowie. Wykorzystują one na ogół mechanizmy automatycznego rozpoznawania mowy. Rozwiązania tego typu cechuje jednak dość niska precyzja oraz pojawiające się błędy, szczególnie w przypadku długich lub szybkich wypowiedzi, których wymowa jest mało dokładna. Dokładność segmentacji ma kluczowy wpływ na przydatność bazy akustycznej, przeznaczonej na potrzeby rozpoznawania lub syntezy mowy. W przypadku syntezy mowy wpływ, jaki wywiera precyzja oznaczonych granic na jej naturalne i spójne brzmienie, zależy od konkretnego systemu, jednak zawsze jest on bardzo duży. Dlatego automatycznie posegmentowane nagrania rzadko są użyteczne z punktu widzenia przytoczonych zastosowań; wydają się jednak stanowić bardzo dobry wstęp do ręcznej korekty segmentacji dużej ilości materiału. Znacznie łatwiej i szybciej można poprawić już istniejące, opisane i wstępnie dopasowane granice, niż tworzyć je od podstaw, jednocześnie wypełniając powstałe segmenty transkrypcją fonetyczną. Warto zauważyć, iż korekta automatycznie wyznaczonych granic, szczególnie dla dużej ilości nagrań, też może w niewielkim stopniu zostać zautomatyzowana (wyszukiwanie i poprawa niektórych błędów). W takim przypadku ręczne poprawki, choć nadal niezbędne, mogą być znacznie ograniczone.

2. Synteza mowy i jej rodzaje

Syntezą mowy nazywa się sztuczne generowanie mowy ludzkiej. Syntezatorem mowy (*speech synthesizer*) określa się system, przetwarzający tekst ortograficzny na mowę (TTS *Text-To-Speech*). Wyznacznikiem jakości syntezatora jest płynność i naturalność brzmienia generowanej mowy. Celem, który przyświeca obecnym projektom jest generowanie mowy o takiej jakości, aby jak najmniej różniła się od naturalnej.

Generalizując, w systemie TTS można wyróżnić cztery podstawowe moduły:

- Moduł analizy tekstu
- Moduł analizy fonetycznej
- Moduł analizy prozodycznej
- Moduł syntezy mowy

Moduł analizy tekstu dostarcza informacji umożliwiających poprawne działanie modułu fonetycznego, w bardziej rozbudowanych systemach analiza lingwistyczna wspomaga także działanie modułu prozodycznego. Innymi słowy moduł ten spełnia dwa główne zadania: normalizuje tekst oraz przeprowadza analizę lingwistyczną. W najprostszych systemach działanie tego modułu ogranicza się jedynie do normalizacji tekstu, czyli konwersji symboli, cyfr, skrótów czy innych znaków nieortograficznych do ich pełnej postaci umożliwiającej wygenerowanie odpowiadającego im zapisu fonetycznego (transkrypcji fonetycznej). Przeprowadzenie poprawnej metodologicznie analizy lingwistycznej (badaniom poddawane są wybrane elementy syntaktyczne i semantyczne, takie jak słowo, fraza, zdanie, wypowiedź) umożliwia określenie wpływu elementów składowych analizy na cechy prozodyczne.

Moduł fonetyczny dokonuje konwersji wyrazów w postaci ortograficznej na zapis (kod) fonetyczny, czyli odpowiada za transkrypcję fonetyczną tekstu. Ponadto zapis fonetyczny może zawierać dodatkowe informacje określające wymowę, przykładowo dotyczące akcentu. Konwersja odbywa się w oparciu o słownik zawierający wersję ortograficzną oraz

transkrypcje fonetyczną wyrazów, oraz/lub zestaw reguł pozwalających na konwersję wyrazów, które nie znalazły się w słowniku.

Moduł analizy prozodycznej jest odpowiedzialny za utworzenie prozodii[5] dla poszczególnych sekwencji fonemów. Ma on wpływ na brzmieniowe właściwości mowy nakładające się na głoskowy, sylabiczny i wyrazowy ciąg wypowiedzi, związane np. z akcentem, intonacją, czy choćby odmiennym brzmieniem zapytania i oznajmienia.

Moduł syntezy mowy generuje sygnał mowy, w oparciu o uzyskaną wcześniej sekwencję fonemów oraz analizę prozodyczną określającą wzorce iloczasowe, kontur melodyczny i obwiednie amplitudy.

Przedstawiony schemat jest tylko ogólnym zarysem budowy syntezatorów mowy, podział na moduły zarówno jak i ich funkcjonalność, a przede wszystkim sposób generowania dźwięków różnią się w różnych systemach. Ogólnie generowanie mowy w systemach TTS opiera się o dwie metody. Pierwsza z nich polega na łączeniu i odtwarzaniu zarejestrowanych wcześniej segmentów mowy naturalnej, poddawanych ewentualnym modyfikacjom podnoszącym jakość wygenerowanego sygnału mowy. Technika ta jest mało skomplikowana i gwarantuje dość szybkie osiągnięcie zwykle zadowalających efektów. Na tej zasadzie działa większość komercyjnych systemów syntezy mowy. Alternatywna metoda zakłada, iż system powinien mieć parametry i możliwości zbliżone do traktu głosowego człowieka (w dziedzinie generowania częstotliwości formantowych głosek, czy też modelowania całego narządu artykulacyjnego człowieka). Generowanie odpowiednich głosek odbywa się według reguł (*by-rule*) opracowywanych przez ekspertów. Jest to trudniejsza droga, głównie ze względu na skomplikowane sterowanie parametrami syntezatora i jednak nie dająca zbyt dobrych wyników jeśli chodzi o naturalność brzmienia mowy.

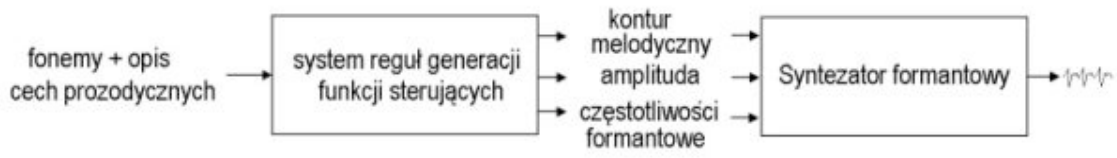
[5] Termin prozodia odnosi się do pewnych właściwości sygnału mowy, które mają wpływ na zmianę głośności, długość sylab (akcent) czy intonację. Cechy prozodyczne odgrywają duże znaczenie w komunikacji werbalnej. Odpowiednie zaakcentowanie sylaby czasem zmienia znaczenie całej wypowiedzi.

Rozróżniamy następujące rodzaje syntezy mowy:

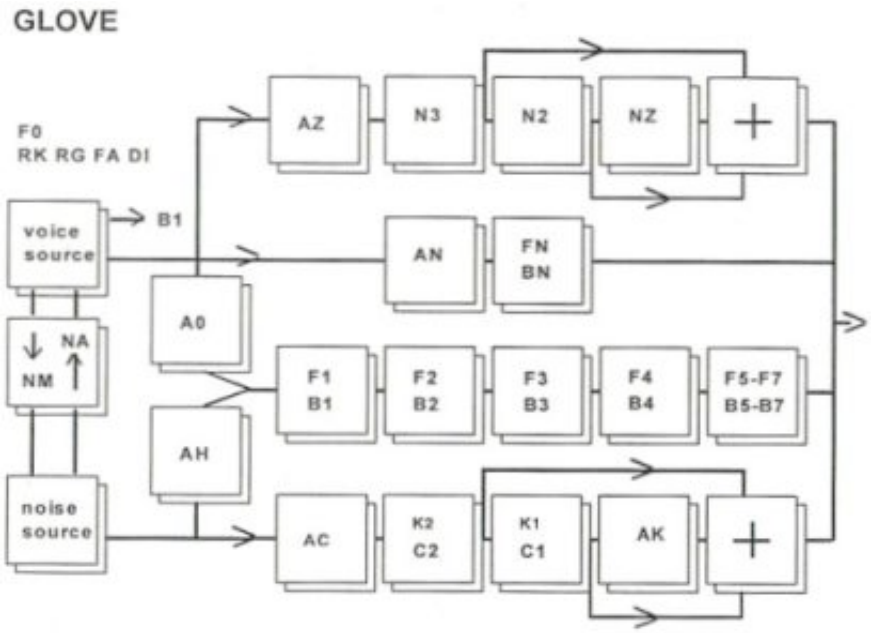
- Synteza formantowa (oparta na regułach ”*by-rule*”)
- Synteza artykulacyjna (oparta na regułach ”*by-rule*”)
- Synteza konkatenacyjna (konkatenacja jednostek):
    - Oparta na fonemach
    - Oparta na difonach
    - Oparta na trifonach
    - Oparta na mikro-segmentach
    - Oparta na sylabach
    - Unit selection (korpusowa, różne jednostki)
- Synteza statystyczna HTS (oparta na ukrytych modelach Markowa - *HMM-based speech synthesis system*)

2.1. Formantowa synteza mowy

Formantowa synteza mowy polega na modelowaniu funkcji przenoszenia toru głosowego w dziedzinie częstotliwości, za pomocą filtrów cyfrowych. Filtry te połączone są ze sobą szeregowo i/lub równolegle i generują dźwięk o charakterystycznych dla mowy częstotliwościach. Sygnał ten odzwierciedla charakterystyczne formanty głosek. Generowanie odpowiednich głosek odbywa się wedle pewnych reguł (*by rule)*, tworzonych dla danego języka przez ekspertów, np. dla AE (*American English*) autorstwa Dennisa Klatta [16]. Schemat obrazujący sterowanie syntezatorem formantowym przedstawiono na rysunku 2.1.1, natomiast rysunek 2.1.2 przedstawia schemat syntezatora formantowego Dennisa Klatta.



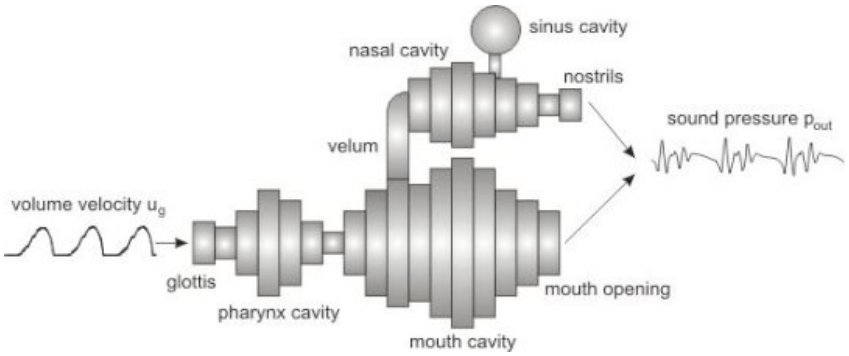
Rysunek 2.1.1: Sterowanie syntezatorem formantowym (w/g [1]).



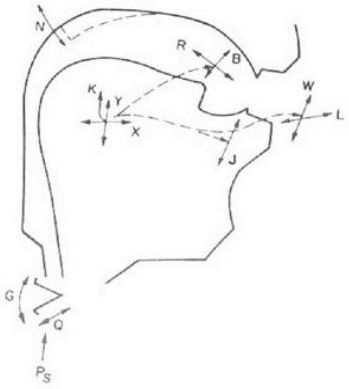
Rysunek 2.1.2: Schemat formantowego syntezatora mowy Dennisa Klatta (w/g [1]).

2.2. Artykulacyjna synteza mowy

Artykulacyjna synteza mowy polega natomiast na modelowaniu rzeczywistego narządu artykulacyjnego w oparciu o zarejestrowane obrazy jego przekrojów. W modelu tym parametry opisują ruchy artykulacyjne, które wpływają na zmianę kształtu toru głosowego. Model ten opisuje bardzo dużo parametrów i oparty jest (podobnie jak w przypadku syntezy formantowej) na generowaniu mowy za pomocą reguł (*by rule*) opracowywanych przez ekspertów. Rysunek 2.2.1 przedstawia konfigurację toru głosowego w danym momencie czasu, przy określonych wartościach parametrów artykulacyjnych (wymienionych na Rysunku 2.2.2).



Rysunek 2.2.1: Przykładowy model toru głosowego zbudowany (na podstawie przekrojów) w oparciu o odcinki rur cylindrycznych.[6]



L - oś stopnia wysunięcia warg
W - oś stopnia otwarcia warg
J - oś położenia szczęki dolnej
X - pozioma oś położenia masy języka
Y - pionowa oś położenia masy języka
K - (0,1) zamknięcie toru
N - otwarcie wlotu do nosa
B - podniesienie czubka języka
R - przesunięcie czubka języka

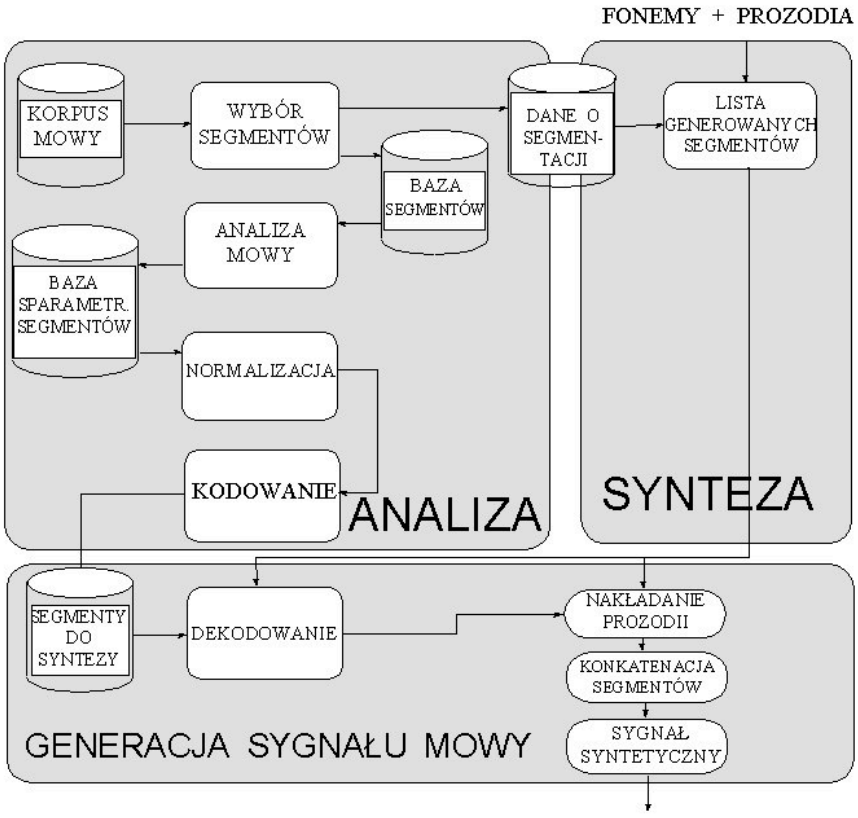
Rysunek 2.2.2: Uproszczone modelowanie ruchów artykulacyjnych (w/g [1]).

2.3. Konkatenacyjna synteza mowy

Konkatenacyjna wersja syntezy generuje mowę poprzez łączenie ze sobą zarejestrowanych wcześniej w bazie danych elementów akustycznych mowy naturalnej (fonemy, difony, trifony, sylaby, itp.), odpowiednio do przetwarzanego tekstu. Innymi słowy system zastępuje zapis ortograficzny prawdziwymi próbkami głosu. W bazie danych przechowywane są wszystkie rodzaje wybranej jednostki akustycznej dla danego języka. Jakość syntezy zarówno jak i rozmiar bazy danych zależy od wyboru jednostki akustycznej.

[6] Źródło: http://wwwicg.informatik.uni-rostock.de/~piet/speak_main.html, 12-2007

Dlatego też syntezę konkatenacyjną można podzielić ze względu na typ użytych jednostek (istnieją syntezatory posiadające w bazie akustycznej jednostki o różnej rozciągłości – rozdział 2.4). Często używane są difony, gdyż umożliwiają dość dobrą jakość syntezy mowy, przy stosunkowo niewielkiej bazie akustycznej (ok. 1500 - 3000 jednostek). Atrakcyjność metody konkatenacyjnej wynika przede wszystkim z tego, że dzięki operowaniu na naturalnej mowie, łączone fragmenty zachowują dość naturalne brzmienie. Mankamentem tej metody jest natomiast fakt, iż łączone („sklejane") ze sobą jednostki są nagrywane w różnych kontekstach. Powoduje to często nienaturalną intonację i rytm syntetycznej mowy. Można częściowo je wyeliminować, zmieniając na podstawie informacji o prozodii wysokość i długość trwania poszczególnych jednostek, nadając głosowi bardziej naturalne ("ludzkie") brzmienie. Ogólny schemat konkatenacyjnej wersji syntezy mowy przedstawiono na rysunku 2.3.1.



Rysunek 2.3.1: Schemat konkatenacyjnej syntezy mowy (w/g [1]).

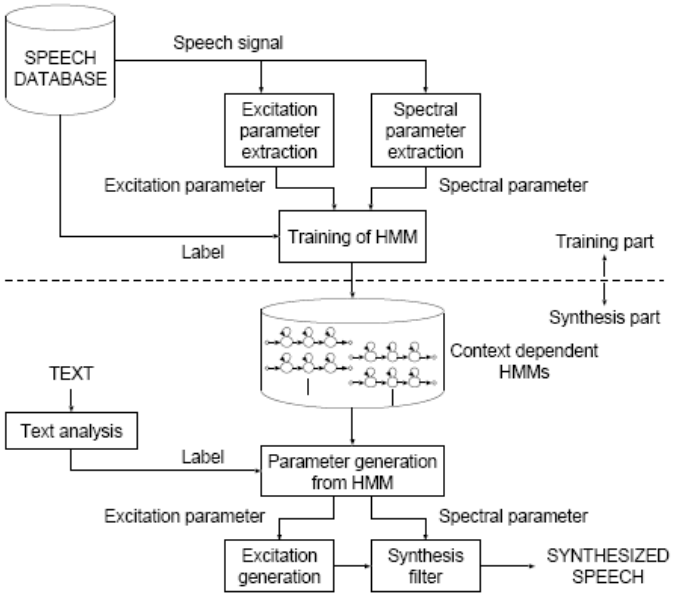
2.4. Korpusowa synteza mowy

Korpusowa synteza mowy (*unit selection*), jest jedną z wersji syntezy konkatenacyjnej. Metoda „*unit selection*" zakłada, iż baza językowa (korpus) jest znacznie większa i zawiera posegmentowane wypowiedzi, a pośrednio różne segmenty akustyczne o różnej rozciągłości (np. fonemy, difony, trifony, sylaby, wyrazy, całe zdania). Oznacza to, iż w korpusie ta sama jednostka (np. fonem) może i powinna wystąpić wiele razy (kilka instancji danej jednostki użytej w różnych kontekstach). Ponadto, aby wygenerować zadaną wypowiedź łączone są ze sobą możliwie jak najdłuższe fragmenty mowy zarejestrowanej w bazie akustycznej, a nie tylko pojedyncze jednostki. Jako kryterium najbardziej optymalnego doboru odpowiednich fragmentów, może być stosowana funkcja kosztu. Oszacowuje ona „koszt" (wagę) każdego z możliwych sposobów wygenerowania zadanej wypowiedzi. Każdy z tych sposobów wykorzystuje różne fragmenty zarejestrowanej w bazie akustycznej mowy. Wspomniana funkcja kosztu porównuje, która z wersji wypowiedzi będzie brzmiała najbardziej naturalnie (koszt wygenerowania mowy będzie wówczas najmniejszy). Pod uwagę brane są różne czasy trwania jednostek akustycznych tworzących łączone fragmenty, ich intonacja czy też kontur widma i dobierane są tak, aby odpowiadały przyjętemu modelowi intonacyjnemu. Na ogół modyfikacje prozodyczne sygnału nie są przy takim podejściu konieczne, co przekłada się na dużą naturalność brzmienia generowanej mowy. W chwili obecnej metoda selekcji jednostek (unit selection) jest najbardziej efektywną i popularną metodą syntezy konkatenacyjnej, co powoduje duże zapotrzebowanie na posegmentowane i transkrybowane bazy akustyczne o stosunkowo dużych rozmiarach..

2.5. Statystyczna synteza mowy (HTS)

Stosunkowo nową i wartą uwagi wersją syntezy mowy jest metoda statystyczna HTS[7] oparta na ukrytych modelach Markowa (*HMM-based speech synthesis system)*, opisanych w rozdziale 3.2.1. Jest to rozwiązanie w pewnym sensie zbliżone do metody konkatenacyjnej, jednak w omawianym przypadku w procesie syntezy nie wykorzystuje się fragmentów mowy naturalnej, lecz zależne od kontekstu modele HMM wytrenowane na podstawie reprezentatywnej bazy akustycznej. Modele te łączone są odpowiednio do przetwarzanego tekstu, a wygenerowane przez nie wektory cech (obserwacje) są podstawą do syntezy mowy realizowanej przez odpowiedni filtr. Należy zaznaczyć, iż osobno modelowane są parametry dotyczące widma (lub cepstrum) i parametry dotyczące tonu krtaniowego (F0, dźwięczność). Dzięki rozdzieleniu tych przebiegów w dość łatwy sposób można modelować emocje, czy też zupełnie zmieniać charakterystykę głosu, wykorzystując techniki adaptacji modeli, opracowane na potrzeby rozpoznawania mowy. Jest to duża zaleta w porównaniu z metodą korpusową, gdzie modyfikacja głosu nie jest możliwa i wymagałaby przeprowadzenia nowych nagrań. Istotną zaletą metody statystycznej jest też niewielki rozmiar wykorzystywanych w trakcie syntezy danych (np. 1MB pomijając moduły analizy tekstu) oraz duża szybkość działania. Oceniając jakość generowanej mowy należy zauważyć, że brzmi ona dobrze, choć mniej naturalnie niż w przypadku metody korpusowej (typowy głos ”wokodera”). Metodę statystyczną cechuje wysoka zrozumiałość, stabilność i dobre modelowanie cech prozodycznych. Ogólny schemat syntezy mowy HTS przedstawiono na rysunku 2.5.1. Więcej na temat syntezy HTS znaleźć można w opracowaniu na temat implementacji tej metody odnośnie języka angielskiego w systemie Festival [18].

[7] http://hts.sp.nitech.ac.jp/?Home, 12-2007



Rysunek 2.5.1: Schemat syntezy mowy HTS (w/g [18])

3. Rozpoznawanie mowy

Technologia automatycznego rozpoznawania mowy (ASR - *Automated Speech Recognition*) umożliwia komputerowi przetwarzanie mowy ludzkiej do postaci tekstowej oraz jej ograniczoną interpretację. Zagadnienie rozumienia mowy jest zdecydowanie bardziej złożonym problemem niż jej artykulacja i to zarówno z punktu widzenia opisu ludzkiego analizatora słuchowego, jak i w zakresie technicznych systemów rozpoznających mowę. Technologia ta wiąże się z najróżniejszymi dziedzinami wiedzy: od akustyki, poprzez lingwistykę, ścisłą matematykę i statystykę, aż po sztuczną inteligencję. Wykorzystywana jest najczęściej jako metoda interakcji człowieka z komputerem czy robotem, poprzez wydawanie komend głosowych. Umożliwia nawet dyktowanie. Często w połączeniu z syntezą mowy tworzy systemy dialogowe. Może też jednak służyć do automatyzacji żmudnego procesu segmentacji zarejestrowanej wcześniej mowy.

Automatyczne rozpoznawanie mowy nie jest zadaniem trywialnym. Wbrew pozorom, dwie osoby wymawiające tę samą wypowiedź, nigdy nie robią tego dokładnie w ten sam sposób. Nawet ta sama osoba różnie wymawia tę samą wypowiedź. Różnice wynikają z budowy anatomicznej "narządu" głosowego, wychowania, nawyków, stanu emocjonalnego,

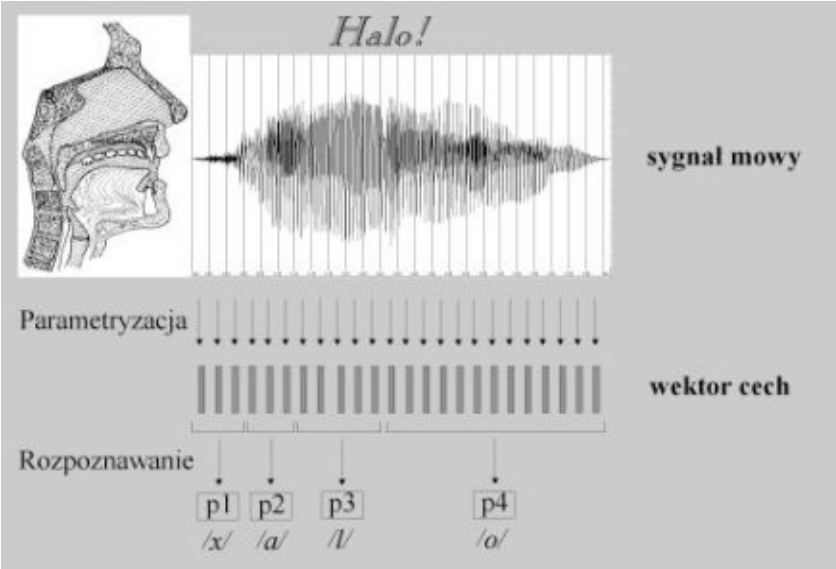
temperatury ciała, zmęczenia czy kataru. Ludzie często mówią niewyraźnie, popełniając błędy. Ponadto mowa zakłócana jest dźwiękami otoczenia czy echa. Choć percepcja tak zróżnicowanej i często ”zaburzonej” mowy dla człowieka nie stanowi problemu, to w przypadku automatycznego rozpoznawania mowy wciąż pozostaje to dużym wyzwaniem.

Ludzie wymawiając tę samą wypowiedź, różnią się przede wszystkim:

- głośnością wymawiania poszczególnych części wyrazu,
- wysokością głosu i przebiegiem zmian tej wysokości przy wymawianiu,
- położeniem częstotliwości formantowych i ich przebiegiem w czasie wymawiania,
- szerokością formantów,
- iloczasem.

Proces rozpoznania mowy, niezależnie od metody jaka zostanie zastosowana, można podzielić na kilka etapów. Etapy te przedstawiono i omówiono poniżej, wraz z najczęściej stosowanymi metodami:

- Digitalizacja, obejmująca przetwarzanie fali akustycznej do postaci cyfrowej (kwantowanie sygnału w dziedzinie czasu i amplitudy)
- *Preprocessing*, czyli wstępne przetwarzanie dźwięku (normalizacja, usuwanie częstotliwości poniżej 50 Hz (DC), preemfaza, detekcja mowy)
- Analiza oraz parametryzacja sygnału mowy (ekstrakcja cech)
    - Analiza widmowa (spektralna)
    - Analiza homomorficzna (cepstralna)
- Rozpoznawanie mowy
    - Metoda analizy statystycznej (HMM – *Hidden Marcov Model*)
    - Sieci neuronowe (ANN – *Artificial Neural Network*)
    - Hybrydy (HMM + sieci neuronowe)



Rysunek 3.1: Etapy prowadzące do automatycznego rozpoznawania mowy (w/g [2]).

Pominięto zagadnienia związane z przetwarzaniem sygnału mowy do postaci cyfrowej (digitalizacja). Warto jedynie zauważyć, iż przetworzony sygnał musi spełniać kryterium Nyquista. Mówi ono, że częstotliwość próbkowania powinna być (co najmniej) dwukrotnie większa niż najwyższa składowa częstotliwościowa w sygnale. Niespełnienie tego kryterium prowadzi do powstania niepożądanego efektu *aliasingu*, czyli nakładania się prążków w widmie podczas dalszej analizy sygnału.

Wstępne przetwarzanie sygnału mowy (*preprocessing*) obejmuje najczęściej następujące elementy: normalizację, obcięcie częstotliwości poniżej 50 Hz, preemfazę oraz detekcję mowy.

Normalizacja polega na równomiernym podniesieniu lub ewentualnym obniżeniu amplitudy całego nagrania tak, by najwyższa jej wartość osiągnęła zadany poziom. Pozwala na uzyskanie zawsze takiej samej maksymalnej amplitudy, bez przesterowania, przy zachowaniu pełnego zakresu dynamiki sygnału.

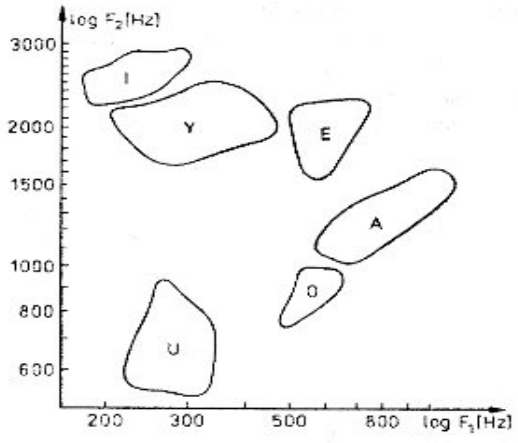
Obcinanie częstotliwości poniżej 50 Hz, realizowane za pomocą filtra górnoprzepustowego (*high-pass filter*), ma na celu wyeliminowanie zakłóceń sieci elektrycznej (*DC*).

Kolejnym etapem wstępnego przetwarzania mowy jest wzmocnienie (podbicie) wysokich częstotliwości sygnału z wykorzystaniem filtra preemfazy (preemphasis). Proces ten ma na celu zniwelowanie niedogodności związanych z faktem, iż obwiednia widma mowy (charakterystyka) opada 6 dB na oktawę (podwojona częstotliwość), czyli amplituda dźwięku maleje o połowę wraz z dwukrotnym wzrostem częstotliwości.

Ostatnim krokiem, który można zaklasyfikować jako wstępne przetwarzanie jest detekcja mowy. Precyzyjne określenie początku i końca wypowiedzi jest konieczne do realizacji prawidłowego rozpoznawania. Najczęściej, aby określić granice mowy, w przebiegu czasowym badany jest poziom energii i częstość przejść sygnału przez zero lub w przebiegu częstotliwościowym cechy charakterystyczne jego widma.

3.1. Analiza oraz parametryzacja sygnału mowy (ekstrakcja cech)

Sygnał, jakim jest mowa, nawet w postaci cyfrowej i wstępnie przetworzony, nie stanowi dobrej podstawy dla algorytmów rozpoznawania. Problemem jest zarówno zbyt duża objętość sygnału, jak i jego niedogodna do analizy struktura (przebieg czasowo-amplitudowy). Dlatego też sygnał przetwarzany jest do postaci dogodniejszej w kontekście jego analizy, a następnie wyszukiwane są w nim cechy charakterystyczne, czyli te, które najlepiej opisują mowę. Ze względu na znaczne ograniczenie informacji, najlepszą formą danych dla potrzeb rozpoznawania mowy, są opisujące sygnał wektory cech. Wektory takie, opisujące wybrane cechy sygnału, o określonej i na stałe przyjętej wymiarowości, zawierają informacje pozwalające na prawidłowe rozpoznawanie mowy. Najłatwiej ocenić to używając kryteriów geometrycznych. W odpowiedniej przestrzeni cech, w której osie odpowiadają oddzielnym cechom, obiekty takie same pod względem fonetycznym powinny skupiać się w ustalonym obszarze przestrzeni, możliwie najbardziej odległym od skupisk odpowiadającym innym klasom.[7] Obrazuje to poniższa ilustracja (3.1.1), na której obszary skupień odpowiadające poszczególnym samogłoskom języka polskiego przedstawiono na płaszczyźnie, której osie wyznaczają pierwszy i drugi formant.



Rysunek 3.1.1: Położenie obszarów skupień samogłosek na płaszczyźnie 1 i 2 formantu (w/g [7]).

Wyraźne rozdzielenie zwartych skupień występowania samogłosek na powyższej ilustracji dowodzi, iż dwa pierwsze formanty dostarczają wystarczającej informacji w kontekście rozpoznawania izolowanych samogłosek.[7] Formanty te nie wystarczają jednak do rozpoznawania wszystkich głosek, na przykład głosek szumowych (trących).

Dobór parametrów zawartych w opisującym sygnał wektorze cech, oraz jego wymiarowość, zależne są od konkretnego systemu i jego przeznaczenia, a co za tym idzie - metody użytej w procesie analizy sygnału. Istotne jest, czy system ma rozpoznawać mowę ciągłą, czy izolowane wyrazy, czy ma rozpoznawać mowę niezależnie od mówcy, czy powinien rozpoznawać większy zakres wypowiedzi, ale dla konkretnego mówcy (system speaker dependent). Kolejne możliwości to rozpoznawanie mówcy niezależnie od wypowiedzi, czy sprawdzanie poprawności wymowy. Dlatego parametry dobiera się w taki sposób, aby ich objętość informacyjna była znacząco mniejsza od objętości informacyjnej źródłowego sygnału, przy jednoczesnym zachowaniu cech istotnych w konkretnym przypadku. Ponadto uzyskane parametry można poddać procesowi optymalizacji, aby ograniczyć ich ilość.

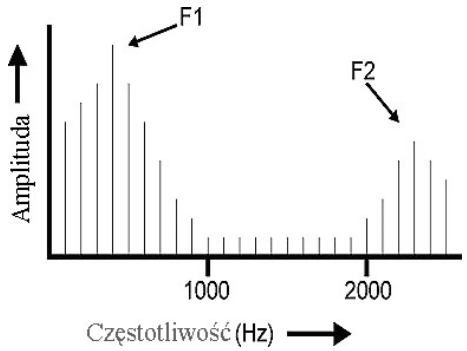
Podsumowując, parametryczny opis sygnału mowy umożliwia pogodzenie dwóch pozornie sprzecznych wymagań, czyli maksymalnie zwięzłej reprezentacji i jednoczesnego zachowania istotnych w konkretnym zastosowaniu szczegółów.

Wśród parametrów najczęściej wykorzystywanych w systemach rozpoznawania mowy wyróżnić należy:

- Parametry widmowe (spektralne), wynikające z charakterystyk amplitudowo-częstotliwościowych
- Parametry widmowo - czasowe, uwzględniające czasową zmienność źródła sygnału
- Parametry cepstralne

3.1.1. Analiza spektralna (widmowa)

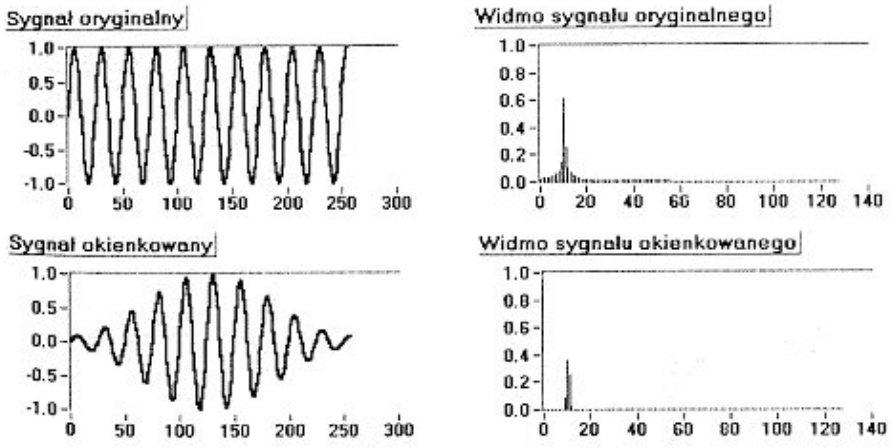
Analiza widmowa oznacza badanie sygnału mowy w dziedzinie częstotliwości. Widmo (spektrum) sygnału otrzymuje się, wykorzystując transformatę Fouriera. Powszechnie stosowaną metodą obliczeniową w przypadku badania mowy jest szybka transformata Fouriera - FFT *(Fast Fourier Transform*). Przykładowe widmo wraz z zaznaczonymi pierwszym i drugim formantem[8] ilustruje rysunek 3.1.1.1.



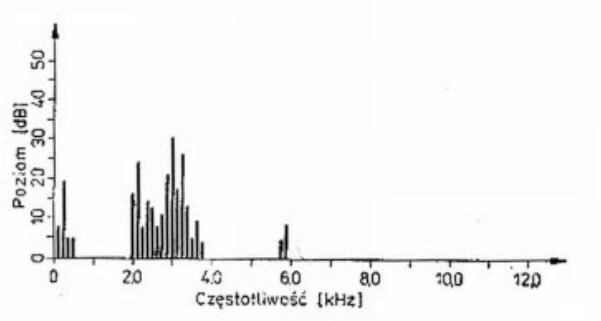
[8] Formant – Lokalne maksimum obwiedni widma, związane z rezonansem powstającym w trakcie głosowym podczas artykulacji. Częstotliwość, przy której występuje to maksimum, nazywa się częstotliwością formantową. *„Ruchy narządów mowy zmieniają rozmiary i proporcje tworzących się wnęk rezonansowych, wobec tego formanty podczas mowy zmieniają swoje położenie, pojawiają się, znikają, zmienia się ich liczba, wielkość i lokalizacja."* [7].

Rysunek 3.1.1.1: Przykład chwilowego widma sygnału mowy, z oznaczonym pierwszym i drugim formantem.

Płaszczyzna amplitudy i częstotliwości wymaga, by badaniu były poddane stany ustalone sygnału, czyli takie, których charakterystyka nie zależy od czasu. W przypadku mowy stany takie są krótkotrwałe i zmieniają się zgodnie ze zmianami konfiguracji toru artykulacyjnego. Mówiąc ściślej, można przyjąć, że sygnał mowy jest stacjonarny w przedziałach czasowych ok. 10 - 30ms. Dlatego, aby osiągnąć stacjonarność w czasie pomiaru, sygnał dzieli się na fragmenty (ramki) o stałej długości (10 – 30ms). Ramki nie mogą być jednak zwyczajnie wycięte z sygnału. Możliwe niezerowe wartości amplitudy na ich krańcach wprowadzałyby silne zakłócenia, w postaci przecieku energii w widmie z jednego prążka do sąsiednich. Dlatego kolejne ramki zachodzą na siebie, a wartości próbek na ich krańcach są tłumione w wyniku pomnożenia sygnału źródłowego przez odpowiednio dobraną funkcję okna (np. okno Hanninga). Należy zauważyć, iż w wyniku okienkowania pojawia się efekt uboczny w postaci spadku energii prążków widma. Wielkość tego efektu, jak i siła tłumienia skrajnych próbek w ramce, zależy od doboru funkcji okna. Widmo chwilowe (Rysunek 3.1.1.1), będące podstawą dla parametryzacji mowy, jest wynikiem transformaty Fouriera obliczonej (FFT) dla konkretnej, okienkowanej ramki sygnału. Na rysunku 3.1.1.2 przedstawiono porównanie ramki oryginalnego przebiegu czasowego wraz z widmem chwilowym oraz ramki okienkowanej metodą Hanninga wraz z jej widmem. Natomiast na rysunku 3.1.1.3 przedstawiono przykład widma chwilowego, z amplitudą wyrażoną w powszechnie stosowanej skali logarytmicznej (dB).

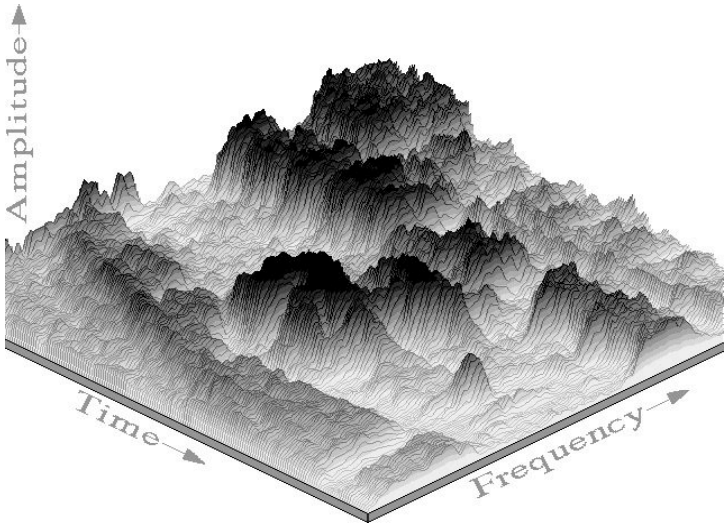


Rysunek 3.1.1.2: Przykładowy sygnał o niecałkowitej liczbie cykli i jego widmo, bez okienkowania i z zastosowaniem okna Hanninga (w/g [14]).

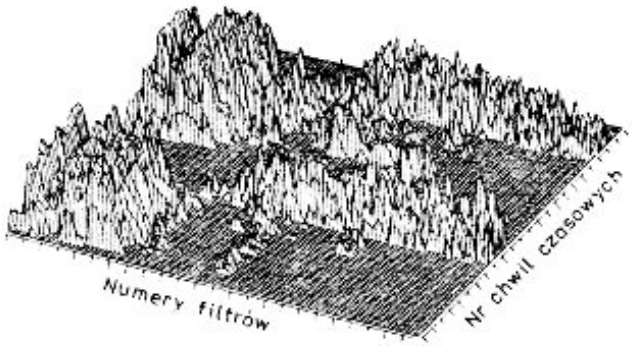


Rysunek 3.1.1.3: Widmo chwilowe stacjonarnej części samogłoski ”i” - amplituda w skali logarytmicznej (w/g [7]).

Widmo chwilowe oblicza się dla każdej ramki sygnału. W ten sposób powstaje seria widm, pozwalająca na obserwację ich zmian w czasie. Powszechną reprezentacją serii widm chwilowych jest trójwymiarowy spektrogram reprezentowany na płaszczyźnie. Aby jednak lepiej zobrazować zagadnienie, przykładowy przebieg widma w czasie przedstawiono na rysunku w rzucie trójwymiarowym. 3.1.1.4 (Spektrogram 3D) i rysunku 3.1.1.5.

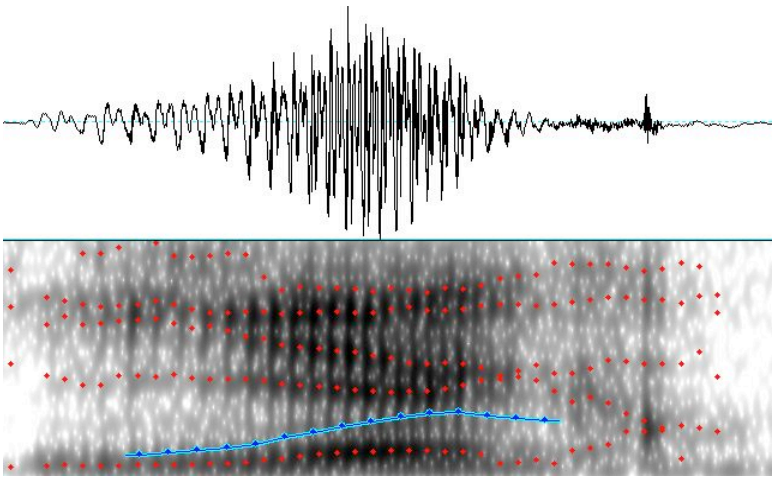


Rysunek 3.1.1.4: Seria widm chwilowych w rzucie trójwymiarowym (Spektrogram 3D)[9].



Rysunek 3.1.1.5: Seria widm chwilowych, widok 3D. Wypowiedź ”serce” (w/g [7]).

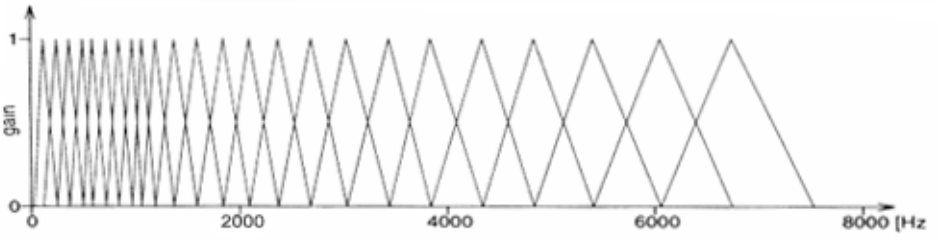
Trójwymiarowy spektrogram, powszechnie stosowany jako reprezentacja przebiegu zmian widma w czasie, także podczas segmentacji nagrań, odpowiada widokowi z góry na serię widm chwilowych. Oś amplitudy w tym spektrogramie reprezentuje kolor w postaci odcieni szarości; im jest ciemniejszy, tym wyższa amplituda danej składowej częstotliwościowej. Zależność tę można zaobserwować także na rysunku 3.1.1.4. Przykładowy spektrogram (wraz z przebiegiem czasowym) przedstawiony został na Rysunku 3.1.1.6. Linie czerwonych kropek to zaznaczone przez program Praat zmiany formantów, niebieska krzywa łącząca niebieskie kropki to wykres intonacji (*Pitch*).



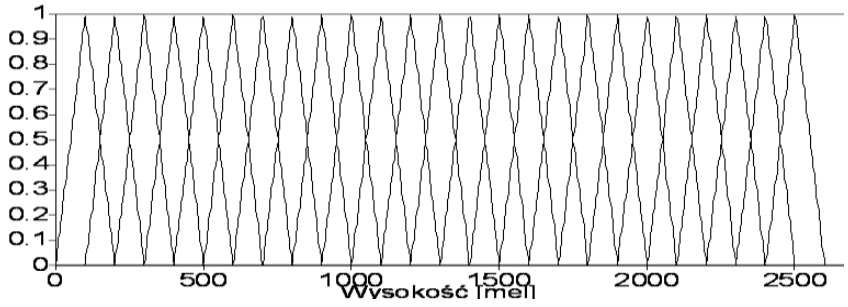
[9] Źródło: http://www.anglistika.upol.cz/phonetics/acph/ls/l3/l3.htm, 11-2006

Rysunek 3.1.1.6: Przykładowy spektrogram sygnału mowy razem z przebiegiem czasowym.

Pomocna w kontekście parametryzacji sygnału mowy na potrzeby jej rozpoznawania jest filtracja nieliniowo przekształcająca skalę częstotliwości tak, aby odpowiadała subiektywnemu odbiorowi częstotliwości przez ludzki słuch. Zwykle używa się w tym celu zestawu ~20 filtrów symulujących działanie ucha. Filtry te rozmieszczone są na osi częstotliwości w skali melowej. Oznacza to, iż częstotliwości środkowe filtrów poniżej 1kHz rozmieszczone są liniowo, natomiast powyżej tego progu rozmieszczone są w skali logarytmicznej. Pasma przepuszczania kolejnych filtrów częściowo się nakładają, a powyżej 1kHz wraz ze wzrostem częstotliwości rośnie ich szerokość. Wzrost szerokości pasma przepuszczania filtrów powyżej 1kHz, a także zagęszczenie filtrów poniżej tego progu wynika z własności percepcyjnych ludzkiego słuchu. Na rysunku 3.1.1.7 przedstawiono przybliżony kształt filtrów melowych w skali częstotliwościowej. Zestaw 25 filtrów w skali melowej przedstawia rysunek 3.1.1.8.



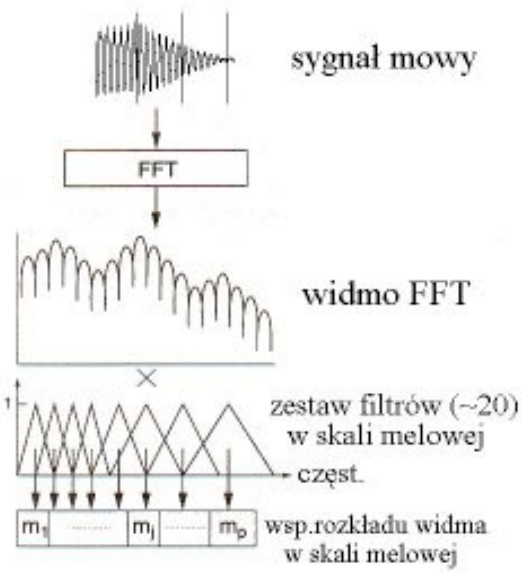
Rysunek 3.1.1.7: Filtry melowe (24) przedstawione w skali częstotliwościowej (w/g [2]).



Rysunek 3.1.1.8: Zestaw 25 trójkątnych filtrów w skali melowej (wysokości).

Parametry widmowe opisujące sygnał mowy uzyskiwane są w wyniku przekształcenia widm chwilowych, najczęściej w skali melowej, na wektory cech, opisujących rozkład energii

w wydzielonych pasmach częstotliwości każdej ramki sygnału. Uproszczony schemat ekstrakcji cech widmowych (spektralnych) sygnału mowy przedstawiono na rysunku 3.1.1.9.



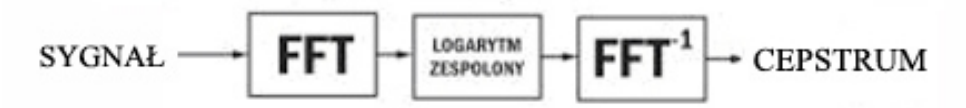
Rysunek 3.1.1.9: Etapy ekstrakcji parametrów spektralnych mowy (w/g [2]).

W procesie analizy (parametryzacji) widmowej otrzymuje się sekwencję wektorów cech opisujących widmowy przebieg sygnału mowy. Mogą one być podstawą dla algorytmów rozpoznawania mowy, jednak lepsze rezultaty można uzyskać stosując parametry cepstralne. Omówienie analizy cepstralnej prowadzącej do uzyskania najczęściej obecnie stosowanych współczynników cepstralnych w skali melowej (MFCC) przedstawiono w poniższym podrozdziale.

3.1.2. Analiza cepstralna (homomorficzna)

Analiza cepstralna jest w uproszczeniu rozwinięciem opisanej w poprzednim rozdziale analizy spektralnej (widmowej). W myśl teorii, sygnał mowy jest splotem funkcji pobudzenia i odpowiedzi impulsowej kanału głosowego. Analiza cepstralna prowadzi do rozdzielenia tych przebiegów. Pozwala przekształcić splot sygnałów w sumę, a co za tym idzie rozdzielić składowe addytywne za pomocą filtracji liniowej. Umożliwia separacje dźwięku na jego

podstawowe składowe i ekstrakcję bardzo przydatnych parametrów do automatycznego rozpoznawania mowy. Podstawowym składnikiem teorii homomorficznego przetwarzania splecionych sygnałów jest przekształcenie cepstralne. Jest to wynik odwrotnej transformaty Fouriera, obliczonej dla amplitudy widma, poddanego wcześniej operacji logarytmowania. Analizę dodatkowo wspomaga tzw. wygładzanie cepstralne, będące wynikiem transformaty Fouriera obliczonego wcześniej cepstrum. Schemat blokowy analizy przedstawiono na rysunku 3.1.2.1.



Rysunek 3.1.2.1: Schemat blokowy analizy cepstralnej (homomorficznej).

Wynikiem analizy są współczynniki cepstralne, najczęściej w skali melowej. Początkowe współczynniki związane są z ogólnym charakterem widma. Jest to wektor parametrów opisujących jego obwiednię. Ułatwiają estymację częstotliwości formantowych (współrzędne lokalnych maksimów wygładzonego cepstrum). Pozostałe (wysokie) współczynniki mogą służyć do stwierdzenia czy istnieje, oraz ewentualnego określenia częstotliwości tonu krtaniowego (dla głosek dźwięcznych). Optymalna liczba współczynników opisujących każdą ramkę sygnału wynosi 39. Pierwszym współczynnikiem jest logarytm poziomu energii. Kolejne, to wektor 12 współczynników MFCC, opisujących charakterystykę cepstrum w skali melowej (MFCC – *Mel Frequency Cepstral Coefficients*). Pozostałe, to pochodne pierwszego i drugiego stopnia obliczone zarówno dla energii jak i 12 podstawowych współczynników, obrazujące dynamiczne zmiany w sygnale. (E, 12MFCC, ΔE, 12ΔMFCC, ΔΔE, 12ΔΔMFCC)

3.2. Metody rozpoznawania mowy

Ostatnim i zarazem kluczowym etapem automatycznego rozpoznawania mowy jest klasyfikacja. Na tym etapie zapada decyzja o rozpoznaniu w oparciu o wektory cech, będące

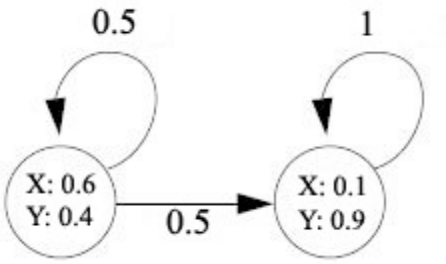
wynikiem analizy i parametryzacji zarejestrowanego wcześniej sygnału mowy. Powszechnie stosowaną technologią do realizacji tego zadania jest metoda statystyczna wykorzystująca ukryte modele Markowa (HMM – *Hidden Markov Model*). Alternatywne rozwiązania opierają się o technikę sieci neuronowych (ANN – *Artificial Neural Network*), a ostatnio coraz większą popularność zdobywają hybrydy obu wspomnianych rozwiązań (NN-HMM). Ukryte modele Markowa, a także metody związane z sieciami neuronowymi opisane zostały w następnych rozdziałach.

W obu przytoczonych technologiach wymagane jest przeprowadzenie procesu uczenia poprawnej klasyfikacji. Wykorzystuje się w tym celu zarejestrowane wzorcowe realizacje różnych elementów mowy, na ogół w posegmentowanej postaci. Należy zaznaczyć, że jakość, ilość i różnorodność wzorcowych nagrań wykorzystanych w procesie uczenia ma decydujący wpływ na dokładność rozpoznawania systemu.

3.2.1. Metody statystyczne (HMM)

Pierwszą z przedstawionych technologii rozpoznawania mowy są ukryte modele Markowa (HMM - *Hidden Markov Model*), będące statystyczną metodą klasyfikacji sekwencji zdarzeń. Ściślej ukryte modele Markowa są zmieniającymi się w czasie modelami stochastycznymi (automaty skończone). Każdy z modeli HMM jest zbiorem połączonych ze sobą stanów, z określonym stanem początkowym. W kolejnych dyskretnych chwilach czasu model zmienia swój aktualny stan przechodząc z jednego stanu do innego, z którym jest on połączony. Model może pozostawać w tym samym stanie, jeśli stan ten jest połączony z sobą samym. W każdym stanie emitowany jest pewien losowy symbol (obserwacja). Wybór przejścia między stanami oraz emisji konkretnej obserwacji są losowe i opisane pewnym rozkładem prawdopodobieństw. Stany, w jakich znalazł się model, pozostają ukryte; rejestrowane są tylko emitowane przez niego obserwacje. Dlatego nazywa się je modelami ukrytymi (niejawnymi). Przykładowy dwustanowy model (HMM) przedstawiono na rysunku 3.2.1.1. Węzły przedstawionego grafu odpowiadają stanom modelu Markowa. Strzałki

obrazują połączenia między stanami i ich kierunek, opisane są prawdopodobieństwami ich wyboru. W symbole odpowiadające stanom wpisane są prawdopodobieństwa emisji obserwacji X i Y w danym stanie.



Rysunek 3.2.1.1: Prosty ukryty model Markowa (HMM), posiadający dwa stany i dwie możliwe obserwacje: X i Y.

Formalnie, ukryty model Markowa opisuje zestaw parametrów V = (P, A, B), gdzie:

- P jest rozkładem prawdopodobieństw stanu początkowego (wektor początkowy),
- A jest to macierz prawdopodobieństw przejść pomiędzy stanami, gdzie a_{ij} jest prawdopodobieństwem przejścia ze stanu i do stanu j,
- B jest zestawem prawdopodobieństw emisji obserwacji, gdzie b_i jest rozkładem prawdopodobieństwa emisji każdej możliwej obserwacji j w stanie i. Rozkład prawdopodobieństw może być ciągły lub dyskretny, zależnie od modelowanych danych. W przypadku rozkładu dyskretnego B przyjmuje postać macierzy, gdzie b_{ij} jest prawdopodobieństwem emisji obserwacji j w stanie i.

Jako że p, a i b są prawdopodobieństwami, muszą spełniać następujące założenia:

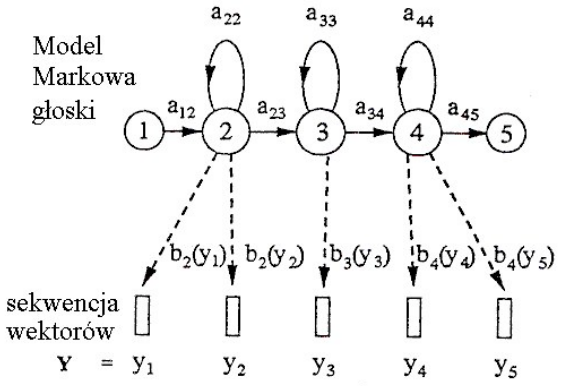
$0 \le p_i \le 1,\ 0 \le a_{ij} \le 1,\ \ 0 \le b_{ij} \le 1,\ \ \forall i, j$,p

$$\sum_i p_i = 1, \quad \sum_j a_{ij} = 1, \quad \sum_j b_{ij} = 1, \ \forall i$$

Jednorodność łańcuchów Markowa orzeka, iż prawdopodobieństwo przebywania procesu w stanie i_{n+1} w chwili n+1, pod warunkiem, że w chwili n przebywał w stanie i_n, ma określoną wartość niezależną od n. Oznacza to, iż prawdopodobieństwa a i b łańcuchów Markowa I rodzaju zależą tylko od stanu poprzedniego.

Ukryte modele Markowa są aktualnie dominującą technologią automatycznego rozpoznawania mowy. Ich stany interpretowane są jako modele akustyczne, wskazujące, jakie dźwięki mogą być zarejestrowane (obserwowane) podczas odpowiadających im fragmentów mowy. Połączenia stanów determinują topologię modelu, czyli, w jaki sposób i w jakiej kolejności stany (dźwięki) mogą następować po sobie w kolejnych chwilach czasu. Połączenia stanów modeli HMM wykorzystywanych w rozpoznawaniu mowy przebiegają od lewej do prawej lub łączą pętlą dany stan z samym sobą, pozwalając by miał arbitralny czas trwania (*left–to–right topology*). Oznacza to wg przyjętej notacji, iż prawdopodobieństwo przejścia pomiędzy stanami $a_{ij}=0$, dla $j<i$. Wadą ukrytych modeli Markowa jest przytoczone wyżej założenie, iż wszystkie prawdopodobieństwa zależą tylko od aktualnego stanu. Założenie to nie jest prawdziwe dla sygnału mowy, którego elementy zależą od swojego otoczenia. Przekłada się to na dość słabe modelowanie koartykulacji, zwłaszcza wstecznej mocno wpływającej na ostateczną postać dźwięków mowy i konieczność dodawania nadmiarowych parametrów do każdej ramki sygnału, opisujących jej otoczenie.

Przykładowy model HMM fonemu przedstawiono na rysunku 3.2.1.2. Trzy główne stany reprezentują nagłos, śródgłos oraz wygłos danej głoski i emitują obserwacje. Pozostałe 2 (nieemitujące), to stan początkowy i końcowy. Modele fonemów mogą łączyć się ze sobą, według określonej gramatyki generatywnej (np. modele głosek połączone w model słowa).



Rysunek 3.2.1.2: Statystyczny model HMM głoski – fonemu (w/g [2]).

Systemy ASR, niezależnie od użytej technologii, wykorzystują jako dane wejściowe wektory cech, uzyskane podczas parametryzacji sygnału mowy. Systemy oparte na HMM wyszukują model lub sekwencję modeli, która z największym prawdopodobieństwem wyemituje obserwacje, odpowiadające danym wejściowym.

Podstawowymi algorytmami w systemach ASR opartych na ukrytych modelach Markowa, są:

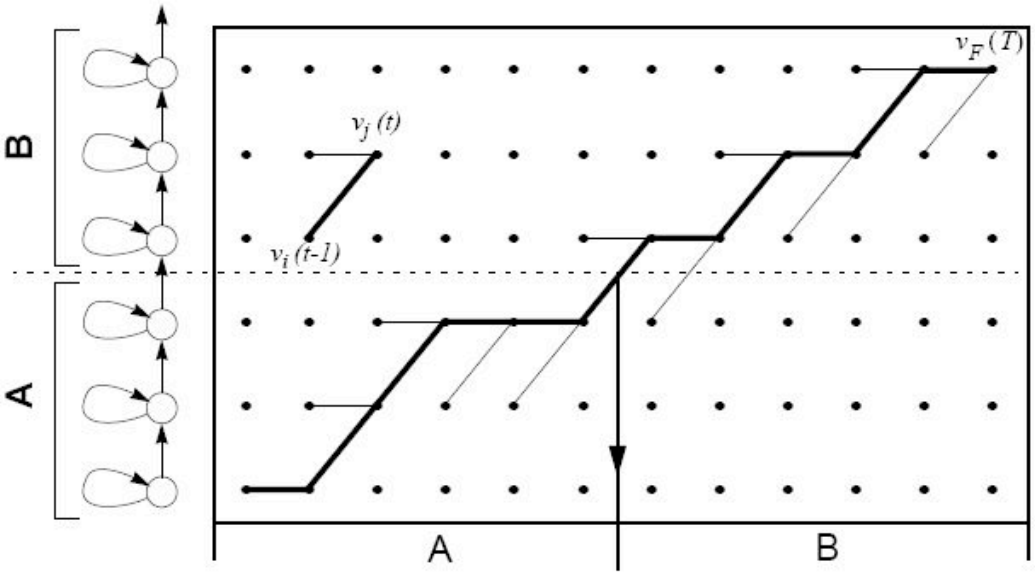
- algorytm Viterbiego, powszechnie stosowany do rozpoznawania mowy
- algorytm "*Forward-Backward*" (Baum-Welch), używany do trenowania modeli HMM

W przypadku rozpoznawania niewielkiej liczby izolowanych słów poszczególne wytrenowane modele HMM mogą reprezentować nawet całe słowa (np. komendy), które mają być rozpoznane. W celu poprawnej klasyfikacji wystarczy wtedy jedynie obliczyć prawdopodobieństwo, że konkretny model HMM wyemitował sekwencję obserwacji, odpowiadającą wejściowej sekwencji wektorów cech. Pozwala to na porównanie wyników modeli różnych słów i wybranie tego o największym prawdopodobieństwie emisji danych wejściowych. Najprostszym rozwiązaniem byłoby wypisanie wszystkich możliwych kombinacji stanów danego modelu o długości równej sekwencji wejściowej i policzenie dla każdej z nich prawdopodobieństwa emisji wejściowych wektorów cech. Jednak algorytm ten ze względu na złożoność obliczeniową jest zupełnie niepraktyczny. Przy tak zdefiniowanych założeniach znacznie wydajniejszym i wystarczającym rozwiązaniem jest algorytm

”Forward”, którego złożoność obliczeniowa jest liniowa. Opiera się on o rekursywne wyliczanie prawdopodobieństwa wygenerowania częściowej sekwencji obserwacji y^t, kończącej się w stanie *j* w czasie *t*, opisanego jako $\alpha_j(t)$. W pierwszym kroku wartość $\alpha_j(t=0)$ inicjowana jest wartością 1 w stanie początkowym i 0 w pozostałych stanach. W następnych krokach do obliczenia $\alpha_j(t)$ wykorzystywane są wartości $\alpha_i(t-1)$ obliczone w kroku poprzednim t-1, dla wszystkich *i* (stanów, z których możliwe jest przejście do stanu j => $a_{ij}>0$). Wartość $\alpha_j(t)$ obliczana jest poprzez zsumowanie iloczynów wartości $\alpha_i(t-1)$, prawdopodobieństwa przejścia ze stanu *i* do stanu *j* oraz prawdopodobieństwa emisji obserwacji y_t w stanie *i*. Formalnie $\alpha_j(t) = \sum_i \alpha_i(t-1)a_{ij}b_i(y_t)$. W celu uzyskania prawdopodobieństwa, iż model HMM danego słowa wyemitował całą sekwencję obserwacji y^T kończąc w stanie J, należy obliczyć $\alpha_J(T)$.

Pomimo iż algorytm ”Forward” może być wystarczający przy rozpoznawaniu niewielkiej ilości izolowanych słów, nie nadaje się do rozpoznawania mowy ciągłej. Wymagałby utworzenia osobnych modeli HMM dla każdej możliwej wypowiedzi. Dlatego powszechnie wykorzystuje się modele HMM reprezentujące pojedyncze fonemy. Dodatkowo stosuje się modele języka (np. statystyczne), które określają, jak modele fonemów mogą łączyć się w wyrazy, zaś te w całą wypowiedź. Przy tak sformułowanych założeniach celem staje się odtworzenie sekwencji stanów wielu różnych modeli, które wyemitują dane wejściowe. Niestety sekwencja stanów z definicji jest ukryta i nie może być jednoznacznie zidentyfikowana, ponieważ istnieje wiele sekwencji mogących wyemitować podobne dane wejściowe. Dlatego poszukuje się takiej sekwencji stanów, która wyemituje dane obserwacje z największym prawdopodobieństwem. Algorytmem wydajnie realizującym to zadanie jest powszechnie stosowany algorytm Viterbiego, znajdujący ścieżkę z najlepszym wynikiem w skierowanym grafie z ważonymi połączeniami. Jego działanie zbliżone jest do algorytmu ”Forward”, jednak zamiast obliczać łączne prawdopodobieństwo różnych sekwencji stanów danego modelu, wyszukuje się jedną, najbardziej prawdopodobną sekwencję stanów różnych modeli. Przekłada się to przede wszystkim na obliczanie w każdym kroku maksimum a nie sumy. $v_j(t) = MAX_i[v_i(t-1)a_{ij}b_j(y_t)]$ Według tego wzoru, w kolejnych krokach dla wybranego stanu wyznacza się najbardziej prawdopodobny stan poprzedni, aż do osiągnięcia

stanu początkowego. Poprzez zachowywanie wskaźników do obliczonych stanów poprzednich możliwe staje się odtworzenie całej sekwencji stanów. Rysunek 3.2.1.3 obrazuje ten proces.



Rysunek 3.2.1.3: Przykład obrazujący działanie algorytmu Viterbiego (w/g [13]).

Oba przytoczone powyżej algorytmy wymagają wcześniejszego przygotowania odpowiednich modeli. Proces ten, zwany trenowaniem lub uczeniem modeli HMM, polega w uproszczeniu na określeniu prawdopodobieństw przejść między stanami oraz rozkładu prawdopodobieństw emisji obserwacji dla każdego modelu. Wykorzystuje się w tym celu wzorcowe, najczęściej posegmentowane, realizacje modelowanego wycinka mowy. Formalnie, aby wytrenować dany model HMM, optymalizuje się wartości a i b, by z założonym prawdopodobieństwem emitował on obserwacje, odpowiadające każdej jego realizacji w danych treningowych (wzorcowych). W tym celu należy zacząć z inicjującymi wartościami zmiennych a i b, najczęściej oszacowanymi na podstawie wstępnych danych, następnie iteracyjnie je modyfikować, aż do osiągnięcia zamierzonych rezultatów dla danych wzorcowych. Algorytmem powszechnie stosowanym do trenowania modeli HMM jest algorytm ”*Forward-Backward*”, znany także jako algorytm ”Baum-Welch”.

Istnieje kilka różnic w architekturze ukrytych modeli Markowa w systemach ASR. Największe dotyczą sposobu modelowania przestrzeni akustycznej, czyli rozkładu prawdopodobieństw emisji obserwacji. Przestrzeń ta opisana może być w sposób dyskretny, ciągły lub mieszany.

Pierwsza w systemach ASR pojawiła się reprezentacja dyskretna. W tym podejściu cała przestrzeń akustyczna podzielona zostaje na skończoną ilość regionów, za pomocą algorytmu klastrującego (grupującego) „*Vector Quantization*" (VQ). Środek ciężkości (centroid) każdego klastra (grupy) reprezentowany jest przez tzw. słowo kodowe (*codeword*), będące indeksem tzw. książki kodowej (*codebook*). Słowa kodowe reprezentują odpowiadające im wektory cech akustycznych, opisujące sygnał mowy. W celu klasyfikacji, kolejne ramki sygnału przekształcane są na słowa kodowe, poprzez znajdowanie w książce kodowej wektora o najmniejszej odległości od wektora cech danej ramki. Obserwacje modeli HMM także są słowami kodowymi, dzięki czemu rozkład ich prawdopodobieństw przybiera postać listy z prawdopodobieństwami emisji każdego słowa kodowego. Nie jest to jednak dobry sposób modelowania dla tak bardzo zróżnicowanego sygnału mowy.

Obecne systemy modelują przestrzeń akustyczną w sposób ciągły. Rozkład prawdopodobieństw emisji obserwacji każdego ze stanów modeli HMM opisuje się za pomocą ciągłych funkcji gęstości, zakładając, iż da się go przedstawić parametrycznie. Powszechnie stosowanym sposobem ciągłej reprezentacji są mikstury Gaussowskie (*Gaussian Mixture Model* – GMM), będące złożeniem (ważona suma) kilku funkcji rozkładu Gaussa, zwanego też rozkładem normalnym. Ilość użytych funkcji (mikstur) jest kluczowa i wymaga pewnego kompromisu. Nie może być zbyt mała, by przestrzeń akustyczna mogła być dobrze modelowana, jednak z każdą kolejną miksturą rośnie ilość parametrów.

Reprezentacja mieszana jest kompromisem pomiędzy dwoma przedstawionymi powyżej rozwiązaniami. Tak jak w reprezentacji dyskretnej, tak i tu istnieje książka kodowa opisująca klastry (obszary skupień) w przestrzeni akustycznej, współdzielona przez wszystkie stany. Jednak zamiast odległości od centroidów (środków ciężkości), przynależność do klastrów

określają w tym przypadku ciągłe funkcje gęstości (powszechnie mikstury rozkładów Gaussa – *Tied Mixture Model*).

Inne różnice w systemach ASR bazujących na HMM dotyczą interpretacji danych wejściowych (wektorów cech). W wielu rozwiązaniach wejściowy strumień danych dzieli się na kilka osobno rozpatrywanych strumieni. Można w naturalny sposób wyodrębnić ich aż do sześciu: współczynniki cepstralne, ich pochodne pierwszego stopnia, pochodne drugiego stopnia, energię i jej dwie pochodne. Wszystkie te dane zawarte są w wektorach cech opisujących każdą ramkę sygnału mowy, co przedstawione zostało w rozdziale 3.1.2. Rozdzielenie tych danych pozwala na osobną interpretację każdego ze strumieni, z osobnymi rozkładami prawdopodobieństw, bądź książkami kodowymi. Przekłada się to na lepsze modelowanie przestrzeni akustycznej i koartykulacji.

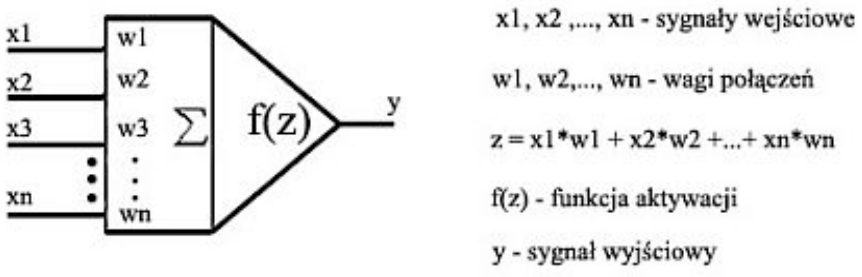
Ilość wypowiedzi, które mogą zostać rozpoznane przez system ASR, w większości zastosowań (pomijając systemy do dyktowania) jest znacznie ograniczona. Taki stan rzeczy stwarza ryzyko, iż wypowiedź mówcy nie znalazła się w słowniku, bądź nie podlega regułom przyjętej gramatyki. Algorytmy bazujące na HMM zaklasyfikują jednak taką wypowiedź jako jedną z przewidzianych w systemie. Wybiorą model lub modele o możliwie największym prawdopodobieństwie emisji danych wejściowych, choćby było ono niewielkie. Aby zmniejszyć ryzyko błędnej klasyfikacji, można wykorzystać tzw. *garbage* model. Jest to model HMM reprezentujący szumy otoczenia oraz wszelkie wypowiedzi, które nie znalazły się w modelowanej gramatyce, także te nieartykułowane. W ten sposób system wybierając *garbage* model, jest w stanie zaklasyfikować sygnał jako nierozpoznany, zamiast klasyfikować go błędnie.

Podsumowując, ukryte modele Markowa, a ściślej ich modyfikacja, to najskuteczniejsza dostępna technologia, użyteczna w każdym zadaniu rozpoznawania mowy. Niezależnie od tego, czy jest to mowa ciągła czy dyskretna, zależna od mówcy czy nie. Dlatego też narzędzia użyte w niniejszym projekcie do wstępnej segmentacji nagrań wykorzystują ukryte modele Markowa.

3.2.2. Sieci neuronowe (ANN)

Zgodnie ze stanowiskiem Tadeusiewicza [8] odnośnie sieci neuronowej: *„Składa się ona z dużej liczby (od kilkuset do kilkudziesięciu tysięcy) elementów przetwarzających informację. Elementy te nazywane są neuronami, chociaż w stosunku do rzeczywistych komórek nerwowych ich funkcje są bardzo uproszczone, by nie powiedzieć prymityzowane. Neurony są powiązane w sieć za pomocą połączeń o parametrach (tak zwanych wagach) modyfikowanych w trakcie tak zwanego procesu uczenia. Topologia połączeń oraz ich parametry stanowią program działania sieci, zaś sygnały pojawiające się na jej wyjściach w odpowiedzi na określone sygnały wejściowe są rozwiązaniami stawianych jej zadań."*

Podstawowym składnikiem sieci neuronowych (ANN - *Artificial Neural Network*) są proste jednostki równolegle przetwarzające ich lokalne sygnały. Dopiero odpowiednio ze sobą połączone są w stanie rozwiązywać skomplikowane zadania, dzięki „wiedzy" zawartej w wagach połączeń. Matematyczny model pojedynczego neuronu opracowany został już w roku 1943 (*binary threshold unit*) przez McCullocha i Pittsa. Opisywał on urządzenie, do którego dochodziła dowolna ilość sygnałów wejściowych, a wychodził jeden. Sygnał na wyjściu przyjmował wartość 1 lub 0, w zależności od tego, czy suma ważona sygnałów wejściowych oraz przypisanych im wag przekroczyła wartość progową. Schematyczny przykład pojedynczego neuronu z dowolną funkcją aktywacji przedstawia Rysunek 3.2.2.1



Rysunek 3.2.2.1: Przykładowy schemat sztucznego neuronu.

Jednym z często spotykanych zastosowań dla sieci neuronowych jest szeroko pojęte rozpoznawanie i klasyfikacja wzorców (przydzielanie wzorcom kategorii). Zadanie rozpoznawania mowy w uproszczeniu sprowadza się właśnie do klasyfikacji wzorców, dlatego naturalnym wydaje się, iż sieci neuronowe znalazły zastosowanie także w tej dziedzinie. Pomimo iż nie spotyka się komercyjnych systemów ASR opartych o sieci neuronowe, w pewnych przypadkach udało się osiągnąć dokładność rozpoznawania na poziomie systemów opartych o HMM. ANN dobrze radzą sobie z rozpoznawaniem izolowanych fonemów, cyfr czy ograniczonego słownika wyrazów. Fakt ten czyni je użytecznymi w przypadku sterowania, np. robotem, czy aplikacją poprzez wydawanie prostych komend głosowych, czy np. rozpoznawania mówcy. Ich główną zaletą jest dobre modelowanie przestrzeni akustycznej, są też odporne na zakłócenia.

Istnieją dwa odmienne podejścia do zagadnienia klasyfikacji przy rozpoznawaniu mowy z wykorzystaniem sieci neuronowych: statyczne i dynamiczne. W przypadku klasyfikacji statycznej na wejście sieci trafia od razu cały wycinek mowy, a sieć podejmuje pojedynczą decyzję. Podejście dynamiczne zakłada, iż danymi wejściowymi dla sieci jest pojedyncza ramka sygnału mowy (lub kilka ramek), a sieć podejmuje serię lokalnych decyzji, podczas gdy ramka przesuwa się po wycinku mowy. Dopiero później, na podstawie wcześniejszych decyzji, sieć klasyfikuje całą wypowiedź.

Oba rozwiązania równie dobrze sprawdzają się przy rozpoznawaniu pojedynczych fonemów. Jednak przy rozpoznawaniu izolowanych słów klasyfikacja dynamiczna wykazuje lepsze przystosowanie do zmienności czasowej sygnału mowy w trakcie trwania słowa. Dalsze skalowanie zadania rozpoznawania mowy do większych jej wycinków pokazało, iż nawet klasyfikacja dynamiczna nie rozwiązuje problemu, jaki dla sieci neuronowych stanowi zmienność czasu trwania sygnału mowy. Rozważając rozpoznawanie mowy ciągłej, problem ten zyskuje całkiem nowy wymiar związany z regułami kompozycji wypowiedzi dyktowanymi przez gramatykę. W efekcie, dla celów rozpoznawania mowy ciągłej wykorzystuje się ukryte modele Markowa (HMM). Nowych rozwiązań poszukuje się, łącząc

metody statystyczne z sieciami neuronowymi (NN-HMM), o czym będzie mowa w następnym rozdziale.

3.2.3. Hybrydy (NN-HMM)

W poprzednich rozdziałach przedstawione zostały dwie odmienne metody, wykorzystywane przy automatycznym rozpoznawaniu mowy. Jedna z nich, to sieci neuronowe (ANN), dobrze modelujące przestrzeń akustyczną, jednak bardzo słabo przystosowane do czasowej zmienności sygnału mowy. Druga metoda, oparta o ukryte modele Markowa (HMM), wydaje się dobra pod każdym względem, jednak i ona ma swoje słabe strony, np. modelowanie przestrzeni akustycznej. Problemem jest też założenie, iż wszystkie prawdopodobieństwa zależą tylko od poprzedniego stanu (*First-Order Assumption*), a kolejne ramki sygnału (wektory cech) nie są ze sobą skorelowane. Założenie to nie jest prawdziwe w przypadku sygnału mowy, w wyniku czego modelowanie koartykulacji jest niewystarczające i konieczne jest uwzględnienie dodatkowych informacji o otoczeniu do każdej ramki sygnału. Dlatego obecnie próbuje się łączyć obie te metody w różne rozwiązania hybrydowe, by zminimalizować niedogodności każdej z nich, a wykorzystać ich mocne strony.

Jednym ze sposobów na połączenie obu metod jest np. implementacja elementów modeli Markowa przy pomocy sieci neuronowych. Przykład takiego rozwiązania Lippman i Gold przedstawili w 1987 roku, a jest nim ”*Viterbi Net*”, sieć neuronowa implementująca przytoczony w rozdziale 3.2.1 algorytm Viterbiego. Danymi wejściowymi dla tej sieci są kolejne ramki sygnału mowy, a danymi wyjściowymi obliczone według algorytmu Viterbiego prawdopodobieństwo $v_j(t) = MAX_i[v_i(t-1)a_{ij}b_j(y_t)]$. Choć algorytm Viterbiego używany jest do rozpoznawania mowy ciągłej, w przypadku *Viterbi Net* jest to niemożliwe, gdyż nie da się odtworzyć sekwencji stanów, która wyemitowała dane obserwacje. Rozwiązanie takie pozwala jedynie na rozpoznawanie izolowanych słów poprzez porównywanie wyników różnych sieci *Viterbi Net* działających równolegle. Podobnym przykładem jest sieć *Alphanet*

przedstawiona w 1990 r. przez Johna S. Bridle, implementująca przedstawiony w rozdziale 3.2.1 algorytm ”*Forward*”.

Pomysłem bardziej popularnym niż implementowanie modeli Markowa w sieciach neuronowych, a także bardziej pasującym do określenia rozwiązanie hybrydowe, jest zawarcie w systemie ASR obu rozwiązań współpracujących ze sobą jako osobne warstwy systemu. Najczęściej spotykanym rozwiązaniem i najlepiej wykorzystującym zalety obu metod jest wykorzystanie sieci neuronowych do modelowania przestrzeni akustycznej, czyli prawdopodobieństw emisji obserwacji dla modeli Markowa (warstwa akustyczna). Oznacza to, iż w większości systemów hybrydowych sieć neuronowa klasyfikuje kolejne ramki sygnału z określonym prawdopodobieństwem jako konkretne fonemy, natomiast modele HMM modelują strukturę fonetyczną słów oraz czasową zmienność sygnału mowy (dekodowanie ciągu obserwacji – warstwa słów). Dodatkowo wykorzystywane są, podobnie jak w przypadku systemów w pełni opartych na HMM, modele języka lub gramatyki określające możliwe sekwencje słów (warstwę językową). Zaletą tego rozwiązania jest przede wszystkim wyeliminowanie istniejącego w HMM założenia o parametrycznym kształcie funkcji gęstości.

Poszukiwania w tej dziedzinie doprowadziły jednak do powstania także innych, mniej popularnych podejść, różniących się pod względem podziału zadań pomiędzy obie technologie. W alternatywnej metodzie przykładowo sieć neuronowa dokonuje rozpoznania na podstawie sygnału mowy, posegmentowanego wcześniej przez modele Markowa. W tym rozwiązaniu na wejście sieci, zamiast pojedynczych ramek, trafiają wycięte z sygnału całe fonemy. Jeszcze inna metoda, będąca rozwinięciem poprzedniej zakłada, iż segmentami, które trafiają na wejście sieci neuronowej, są całe wyrazy. Ponieważ czas trwania wyrazów nie jest stały, każdy wyraz musi zostać nieliniowo przeskalowany do określonej wartości algorytmem DTW (*Dynamic Time Warping*). Warto jeszcze wspomnieć o metodzie polegającej na wspomaganiu przez sieci neuronowe parametryzacji dźwięku. W rozwiązaniu takim danymi wejściowymi dla modeli Markowa są parametry uzyskane dzięki połączeniu analizy dźwięku i działania sieci neuronowych.

Niezależnie od przyjętego sposobu, próby łączenia sieci neuronowych z modelami Markowa mają na celu optymalizacje istniejących rozwiązań pod kątem wydajności lub poziomu dokładności rozpoznawania. Często optymalizacja dotyczy konkretnych zastosowań i nierzadko znacznie komplikuje cały system. Dlatego też rozwiązania hybrydowe największym zainteresowaniem cieszą się, jak na razie, przede wszystkim w środowisku akademickim.

4. Segmentacja akustycznej bazy językowej

Celem niniejszego projektu była segmentacja językowej bazy akustycznej na potrzeby realizacji polskiej korpusowej syntezy mowy. Dodatkowo, konieczne było dostosowanie posegmentowanej bazy do wymogów systemu Festival oraz powstającego syntezatora. Wykorzystany w projekcie korpus językowy oraz nagrania, przygotowane i udostępnione zostały przez Krzysztofa Szklanny, autora powstającego syntezatora. W procesie segmentacji wykorzystano narzędzia do rozpoznawania mowy, umożliwiające znaczną automatyzację tego żmudnego i czasochłonnego zadania. Z punktu widzenia algorytmów dokładne określenie granic jednostek akustycznych jest często zadaniem trudniejszym niż samo rozpoznawanie mowy. Nawet, gdy dysponuje się fonetycznym zapisem wypowiedzi. Automatyczna metoda na ogół nie jest zbyt dokładna, szczególnie w przypadku dłuższych fragmentów mowy

ciągłej. Precyzja segmentacji nagrań ma jednak kluczowy wpływ na użyteczność bazy akustycznej. Dlatego wymagane było opracowanie metod wyszukiwania, a następnie poprawiania błędów automatycznej segmentacji. Ostatnim etapem było dostosowanie reprezentacji danych do konkretnych założeń syntezatora oraz systemu. Wymagało to przede wszystkim dostosowania formatu danych i transkrypcji fonetycznej do wymogów i reguł syntezatora (np. brak możliwości używania znaku ‘ w transkrypcji, czy polskich znaków diakrytycznych w ortograficznym zapisie wypowiedzi), zależnych także od systemu Festival. Dodatkowo utworzony został słownik transkrypcji fonetycznej dla wyrazów, które nie podlegały przyjętym regułom fonologicznym obowiązującymi w języku polskim. Ostatecznie wygenerowane zostały pliki w formacie systemu Festival, opisujące granice fonemów każdego nagrania. Pozwoliło to wykorzystać powstałą bazę do syntezy mowy, a także wygenerować zdania testowe, by zweryfikować jakość segmentacji.

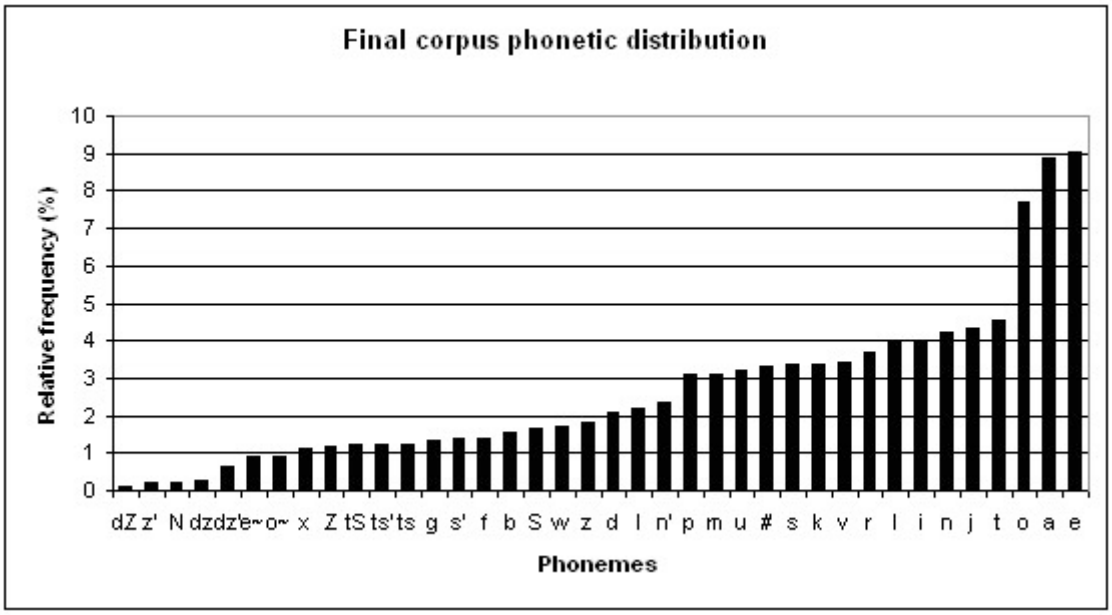
W następnych rozdziałach przedstawiono cały proces powstawania akustycznej bazy językowej dla korpusowej syntezy mowy. Na wstępie analizie poddano wykorzystaną bazę językową oraz nagrania, by móc przejść do zagadnień dotyczących procesu wstępnej automatycznej segmentacji, metody wyszukiwania błędów i ich poprawiania, weryfikacji oraz procesu dostosowania bazy akustycznej do wymogów syntezatora i systemu Festival.

4.1. Baza językowa (korpus)

Baza językowa (korpus), którą wykorzystano w projekcie, opracowana i udostępniona została przez Krzysztofa Szklanny. Teksty zostały tak dobrane, by uzyskać wystarczającą reprezentację poszczególnych głosek i jak największą ilość ich połączeń charakterystycznych dla języka polskiego. W większości przypadków teksty dobierane są pod kątem przyszłego zastosowania bazy. Ręczne dobieranie tekstów ma jednak sens tylko dla korpusu o bardzo zawężonej dziedzinie (*limited domain* – np. informacji kolejowej). W przypadku baz o bardziej ogólnym charakterze, podstawowym materiałem często są teksty z gazet czy radia, a ich dobór, przy wykorzystaniu odpowiednich narzędzi, jest zautomatyzowany. Podobnie było w przypadku bazy językowej użytej w projekcie.

Korpus powstał na bazie tekstów wypowiedzi sejmowych oraz tekstów recenzji z gazet. Najpierw przeprowadzono analizę istniejących baz językowych z tych dziedzin. Na podstawie analizy jako podstawowe źródło wybrano wypowiedzi sejmowe. Ważnym atutem tej bazy językowej był jej rozmiar, czyli 300 MB odpowiadające 5778460 zdaniom. Niezbędne było usunięcie wszelkich tagów oraz innych meta danych. Skróty, akronimy i arabskie cyfry zostały rozwinięte do pełnych form. Następnie automatycznie utworzono transkrypcję fonetyczną każdego zdania dla fonemów, difonów i trifonów w kodzie SAMPA. Pozwoliło to na analizę statystyczną wystąpień każdej z przytoczonych jednostek. Stworzony został słownik zawierający transkrypcję ponad 11000 słów. W dalszych etapach prac okazało się, iż automatyczna transkrypcja wymagała korekt, szczególnie w przypadku słów obcojęzycznych. Kolejnym etapem było tworzenie i balansowanie[10] przyszłej bazy językowej w programie CorpusCrt (Bailador, 1998). Celem balansowania było uzyskanie o wiele mniejszego niż źródłowe bazy językowe korpusu, dostatecznie bogatego pod względem fonetycznym i reprezentatywnego dla języka polskiego. Wynikiem pierwszego etapu balansowania było 12 różnych korpusów, po 2500 zdań każdy. Kryteria doboru zdań dotyczyły minimalnej i maksymalnej długości zdań oraz ilości wystąpień fonemów, difonów i trifonów w każdym z korpusów. Kolejnym krokiem było połączenie 12 uzyskanych korpusów w jeden wspólny oraz jego dalsza redukcja. Celem była optymalizacja ilości zdań i częstości występowania różnych jednostek akustycznych. Następnie przygotowano i zbalansowano według tych samych kryteriów bazę językową z recenzjami z gazet, zawierającą 21135 zdań. Przygotowany wcześniej korpus sejmowy połączono z bazą zawierającą recenzje z gazet, a także dodano 90 rzadkich wyrazów i przeprowadzono ostateczne balansowanie połączonego korpusu. Wynikowa baza językowa liczy 2150 promptów. 2060 to zdania, z czego 1970 to zdania oznajmujące, reszta to pytania i zdania rozkazujące. Pozostałe 90 wpisów są to pojedyncze, rzadko występujące wyrazy. Przy takiej ilości wypowiedzi, udało się osiągnąć 1200 różnych difonów i 14505 trifonów. Poniżej (Rysunek 4.1.1) przedstawiono wykres częstotliwości występowania poszczególnych fonemów w korpusie. Dodatkowy 38 fonem, oznaczony jako „#”, reprezentuje ciszę międzywyrazową. Należy zaznaczyć, iż przedstawiony rozkład częstości występowania poszczególnych fonemów w korpusie jest zgodny z częstością występowania tych fonemów w mowie polskiej.

[10] Balansowanie (bazy językowej), to w uproszczeniu wyszukiwanie (dobór) pewnych wyrazów, zdań czy innych jednostek językowych spełniających pewne kryteria, spośród dużych zbiorów tekstowych. Kryterium doboru podczas balansowania może dotyczyć także, tak jak w opisywanym przypadku, występowania różnych jednostek akustycznych, oczywiście pod warunkiem posiadania zapisu fonetycznego źródłowych tekstów.



Rysunek 4.1.1: Częstotliwość występowania poszczególnych fonemów w bazie językowej (w/g [5]).

Wykorzystany korpus jest zbiorem 2150 wypowiedzi, zapisanych ortograficznie w pliku tekstowym, w następującej konwencji: każda linia pliku to jedno zdanie/wypowiedź.

Każde zdanie zapisane jest w następującej formie:

Numer zdania w postaci sxxxx, gdzie xxxx jest kolejnym numerem zdania. Numer oddzielony jest dwukropkiem i tabulacją od samego zdania zakończonego kropką, znakiem zapytania, bądź wykrzyknikiem. Brak zakończenia zdania jakimkolwiek znakiem traktowany był jako zdanie zakończone kropką. Przyjęty zapis czterocyfrowy pozwolił uniknąć kłopotów z kolejnością wyświetlania plików. W dalszych etapach prac, by ułatwić identyfikację, każde ze zdań zapisane będzie w osobnym pliku o nazwie takiej jak numer zdania.

Poniżej zamieszczono przykładowe wpisy użytego korpusu:

s0001: jak pan ją ocenia jako celnik , jako osoba która nadzoruje sprawy celne i sprawy graniczne ?

s0002: czy dlatego że dyrektorką jest pani glemp gazownicy robią co kilka miesięcy jej prezent ?

s0003: pytałem jeszcze dziś czy może przyszła jakaś odpowiedź ?

s0004: co będzie wtedy podstawą do dalszych prac czy projekt pierwotny czy to co wypracowała podkomisja ?

s0005: zapytajmy jak to jest w szwajcarii jak to jest w austrii ?

….

s2147: buzia

s2148: w optymiźmie

s2149: zipnąć

s2150: ziąb

Wraz z bazą językową na potrzeby projektu udostępniony został także słownik transkrypcji fonetycznej. Został on wykorzystany podczas automatycznej segmentacji.

4.2. Nagrania

Drugim etapem powstawania akustycznej bazy językowej było nagranie wypowiedzi zawartych w korpusie językowym. To bardzo ważny etap mający kluczowy wpływ na brzmienie przyszłego syntezatora. Nagrania przeprowadzone zostały przez Krzysztofa Szklanny. Następnie udostępnione zostały wraz z korpusem językowym i słownikiem transkrypcji fonetycznej do dalszych prac nad segmentacją (format RAW).

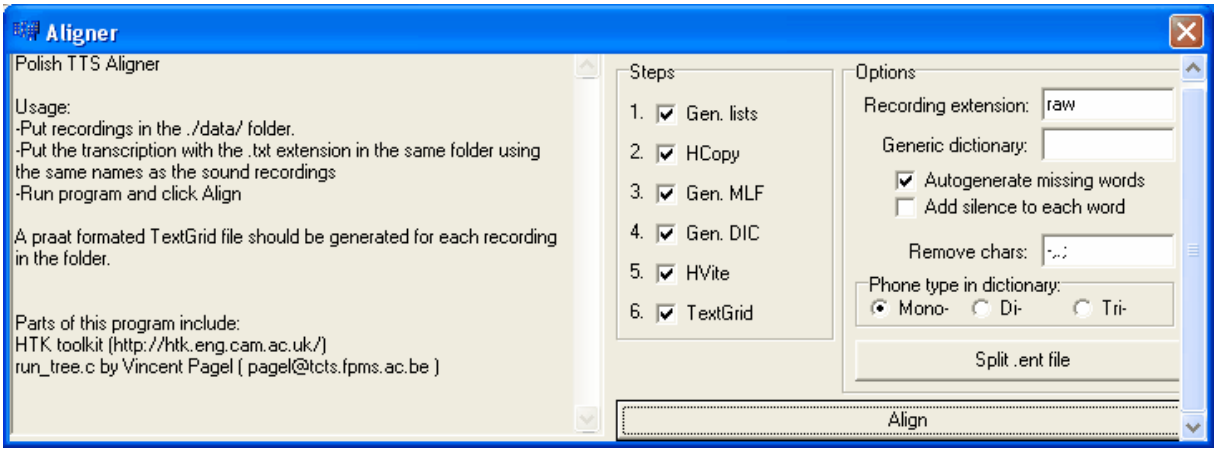
Przed mówcą stało trudne zadanie, gdyż trzeba było zarejestrować 2150 wypowiedzi. Przy czym w każdej chwili musiał on mieć na uwadze dokładność i reguły wymowy, a także starać się mówić monotonnie. Monotonność oznacza tutaj wymowę w jak największym stopniu jednostajną, pozbawioną emocji, o ograniczonej intonacji i jak najmniejszych zmianach tempa czy głośności. Ma to bardzo duże znaczenie w przypadku nagrań przeznaczonych do konkatenacyjnej syntezy mowy, gdyż ich fragmenty będą łączone ze sobą i aby nie powstawały zniekształcenia muszą być jak najbardziej zbliżone do siebie pod kątem prozodii. Warto zaznaczyć, iż największą spójność nagrań uzyskuje się przeprowadzając je podczas jednej sesji. Powoduje to jednak zmęczenie narządów artykulacyjnych mówcy, prowadzące do mniej dokładnej wymowy i specyficznej chrypki (glotalizacji). Ze względów

technicznych zarejestrowanie wszystkich wypowiedzi za jednym razem było niemożliwe. Nagrania przeprowadzano w sesjach trwających od 1 do 4 godzin, a cały proces trwał miesiąc. Na „brzmienie głosu" ma wpływ wiele czynników, takich jak np. stan emocjonalny, temperatura ciała czy zmęczenie mówcy. Dlatego trudno było zachować taką spójność głosu i wymowy podczas wielu sesji. By osiągnąć jak największe podobieństwo, prowadzone były odsłuchy poprzednich sesji, by „dostroić się" z wymową w bieżącej sesji nagraniowej do ich stylu. Ponadto używano tego samego mikrofonu, ustawionego w tym samym miejscu i pozycji. Był to mikrofon dynamiczny Rode NT1000. Nagrania przeprowadzono w studiu Polsko-Japońskiej Wyższej Szkoły Technik Komputerowych. Głosu użyczył Krzysztof Szklanny. Częstotliwość próbkowania to 48 kHz i 16 bitowa reprezentacja. Zapewniło to możliwość łatwej zmiany częstotliwości próbkowania na 16 lub 8 kHz, najczęściej używanych w systemach TTS. Do odsłuchu i weryfikacji jakości posłużyły nauszne (zamknięte) słuchawki. Podczas nagrań napotkano problemy związane np. z zakłóceniami sieci elektrycznej (*AC*), przez co konieczne było ponowne zarejestrowanie ponad 25% materiału. Wynikowa baza zawiera 2150 wypowiedzi, trwających od 2 do 13 sekund, oraz przykładowe chrząknięcia, mlaśnięcia czy śmiech. Rozmiar danych to 1,2 GB.

4.3. Automatyczna segmentacja nagrań (Aligner, HTK)

Trzeci etap powstawania bazy akustycznej, a zarazem główny cel niniejszej pracy, to segmentacja nagranych wcześniej wypowiedzi. Proces ten powinien być przeprowadzany z jak największą uwagą i dokładnością. Od jego poprawności zależy bowiem przydatność wynikowej bazy akustycznej, a w opisywanym przypadku - jakość generowanej przez syntezator mowy. Największą dokładność wciąż osiągnąć można przez ręczne wyznaczanie granic wybranych jednostek akustycznych. Ogrom materiału, w postaci 2150 wypowiedzi, wymusił jednak częściową automatyzację tego zadania. Polegała ona przede wszystkim na wstępnym dopasowaniu sygnału i granic transkrypcji fonetycznej (*alignment*), przy pomocy narzędzi opartych o mechanizmy rozpoznawania mowy. Dodatkowo częściowo zautomatyzowane zostały: wyszukiwanie oraz korekta niektórych błędów.

Do automatycznej segmentacji posłużył opracowany przez prof. Krzysztofa Maraska program Aligner, przy czym wykorzystana wersja została zmodyfikowana przez Danijela Korzinka. Program ten powstał na podstawie elementów ogólnie dostępnego oprogramowania HTK[11], będącego kompleksowym zestawem współdziałających narzędzi, umożliwiających przetwarzanie, a przede wszystkim rozpoznawanie mowy, z wykorzystaniem ukrytych modeli Markowa (HMM). Narzędzia te, pozwalają na modelowanie gramatyk, parametryzację oraz rozpoznawanie mowy, a także tworzenie i trenowanie modeli HMM. Elementy HTK stanowią użyteczne moduły dla programów wykorzystujących mechanizmy rozpoznawania mowy, razem obejmując cały zakres związanych z tą dziedziną zagadnień. Danymi wejściowymi dla programu Aligner są pliki dźwiękowe z nagraniami oraz ich ortograficzny zapis. Wynikiem jego działania są pliki o rozszerzeniu Textgrid, w formacie programu Praat, opisujące granice jednostek akustycznych. Transkrypcję fonetyczną wypowiedzi w kodzie SAMPA, Aligner może wygenerować automatycznie, może też korzystać z zewnętrznego słownika odwzorowań. Transkrypcja, zarówno jak i segmentacja może być dokonana na podstawie fonemów, difonów lub trifonów. Ponadto, użytkownik może zadecydować, które etapy procesu mają być wykonane (etapy parametryzacji, transkrypcja, segmentacja). Okno programu Aligner przedstawiono na rysunku 4.3.1.



Rysunek 4.3.1: Okno programu Aligner.

[11] http://htk.eng.cam.ac.uk/, 12-2007

Należy zauważyć, iż poprawność segmentacji programu Aligner, a pośrednio także rodzaj użytej jednostki akustycznej, według której jest ona przeprowadzona, wynikają z zastosowanych modeli Markowa (HMM). Na potrzeby prac udostępnione zostało 5 różnych zestawów modeli HMM. W następnym rozdziale przedstawiono ich krótką charakterystykę, proces testowania, oraz wyboru modeli, a co za tym idzie jednostki akustycznej wykorzystanej w projekcie.

Podczas wstępnej segmentacji, pominięto proces generowania transkrypcji fonetycznej. Automatyczna wersja wymagałaby dodatkowej weryfikacji i korekty, gdyż wymowa konkretnego mówcy w pewnych przypadkach może odbiegać od ogólnych reguł dla języka polskiego. Ponadto reguły te z definicji byłyby błędne w przypadku słów obcojęzycznych, których dość duża ilość znalazła się w korpusie. Dlatego wykorzystano słownik z transkrypcją fonetyczną, udostępniony wraz z korpusem językowym. Pomimo, iż ten słownik także opierał się na automatycznie wygenerowanej transkrypcji, zawierał już jednak bardzo dużo ręcznie naniesionych poprawek. Przyjmując, iż większość błędów w słowniku została wyeliminowana oraz mając na uwadze fakt, że i tak niezbędna jest weryfikacja i korekta segmentacji, postanowiono nanosić poprawki w transkrypcji podczas wspomnianych prac. Nieprawidłowości tego typu na ogół powodują tak duże błędy w wyznaczaniu granic, iż stosunkowo łatwo jest je zlokalizować bezpośrednio na wizualizowanym przebiegu sygnału, nawet bez odsłuchu. Choć wymagają one dużych korekt, to ręczne sprawdzenie całej transkrypcji zajęłoby znacznie więcej czasu.

Przygotowanie, oraz konfiguracja programu Aligner wymagały:

- Utworzenia osobnych plików tekstowych dla każdego nagrania. Zadanie to zrealizował prosty skrypt (Perl) pobierający po jednym wierszu z bazy językowej i zapisujący go w pliku tekstowym o nazwie odpowiadającej numerowi wypowiedzi.
- Skopiowania do katalogu „data” plików z nagraniami oraz plików tekstowych.
- Ustawienia w pliku konfiguracyjnym HTK o nazwie hcopy.conf, parametru „sourcerate = 208”, określającego częstotliwość próbkowania nagrań. Wartość 208 odpowiada częstotliwości 48 kHz i oznacza, iż próbki mają być pobierane co 0.0208ms (1000ms / 48000 = 0.0208(3) ms).

- Przygotowania słownika, przekopiowania go do głównego katalogu programu oraz zmiany jego nazwy na „sampa.dic”, aby Aligner mógł z niego korzystać.
- Skopiowania do głównego katalogu wybranego pliku z modelami HMM i zmiany jego nazwy na „model.mmf”, by był on widoczny dla programu.
- Odznaczenia 4 kroku (Gen.Dic) w oknie programu Aligner, aby korzystał z zewnętrznego słownika transkrypcji fonetycznej i nie zastąpił go własnym, wygenerowanym automatycznie.

Dostosowanie słownika polegało na ustawieniu kodowania znaków na „ISO-latin 8859-2” oraz dodaniu na początku pliku 2 wersów przedstawionych poniżej, łącznie z kilkoma przykładowymi liniami słownika. Taki zapis oznaczał, iż cisza (na początku i końcu wypowiedzi) oznaczana ma być przez Aligner jako „_sil_”.

SIL [] _sil_
sil [] _sil_
…
Przykładowe wpisy słownika:

świąt	s' f j o n t
świątyni	s' f j o n t I n' i
świadczą	s' f j a t tS o~
świadczący	s' f j a t tS o n ts I

…

4.3.1. Wybór modeli HMM oraz jednostki akustycznej

Czynnikiem definiującym działanie programu Aligner i decydującym o powodzeniu etapu automatycznej segmentacji, były zastosowane ukryte modele Markowa, przedstawione bliżej

w rozdziale 3.2.1. Na użytek prac nad segmentacją udostępnione zostało przez Krzysztofa Szklanny pięć różnych, gotowych do użycia kompletów modeli HMM. Wszystkie stworzone i wytrenowane zostały z wykorzystaniem HTK (*Hidden Markov Model Toolkit*). Do parametryzacji każdego z modeli posłużono się 39 współczynnikami mel-cepstralnymi (E, 12 MFCC, ΔE, 12 ΔMFCC, ΔΔE, 12 ΔΔMFCC), opisanymi w rozdziale 3.1.2. Ich skład to: logarytm poziomu energii, 12 współczynników MFCC (*Mel Frequency Cepstral Coefficients*) opisujących charakterystykę cepstrum w skali melowej oraz pochodne pierwszego i drugiego stopnia obliczone zarówno dla energii, jak i 12 podstawowych współczynników. Różnice pomiędzy dostarczonymi kompletami modeli dotyczyły reprezentowanego rodzaju jednostek akustycznych oraz rozmiaru i cyklu pobierania kolejnych ramek sygnału. Cztery z dostarczonych zestawów modelowały fonemy, piąty natomiast difony. Krótką charakterystykę każdego z nich przedstawiono poniżej, wraz z procesem testowania i wyboru najskuteczniejszego. Brak ograniczeń ze strony systemu Festival, w którym zostanie wykorzystana powstająca baza akustyczna, oraz ze strony przyszłego syntezatora, zapewnił swobodę w wyborze modeli HMM, a co za tym idzie wykorzystanej jednostki akustycznej (fonemy, difony).

Wszystkie dostarczone komplety reprezentujące fonemy zawierały 38 modeli HMM, czyli o jeden więcej niż ilość fonemów w języku polskim. Dodatkowy HMM, wykorzystywany przez program Aligner, reprezentował ciszę. Modele składały się z trzech stanów, reprezentujących nagłos, śródgłos oraz wygłos danego fonemu. Na początku powstały trzy odmienne struktury HMM. Ich odmienność wynikała z różnych rozmiarów i innego cyklu pobierania kolejnych ramek parametryzowanej mowy. W pierwszym z kompletów wykorzystano okno analizy o szerokości 5ms, pobierane co 1ms. W drugim użyto ramkę o rozmiarze 15ms pobieraną co 5ms, natomiast w trzecim odpowiednio ramkę 25ms i przesunięcie 10ms. Każdy z 3 wspomnianych zestawów wstępnie wytrenowany został na podstawie 585 fonetycznie dobranych nagrań, zawierających pojedyncze wyrazy i zdania. Aby podnieść poziom dokładności rozpoznawania, do każdego stanu wstępnie wytrenowanych modeli HMM dodano mikstury Gaussowskie, po czym ponownie estymowano parametry. Ten proces we wszystkich przypadkach powtórzono 3 razy. Na zakończenie przeprowadzono ostateczną estymację parametrów wszystkich modeli HMM na podstawie zarejestrowanych nagrań 40 mówców (łącznie 585 wypowiedzi). Czwarty zestaw ukrytych modeli Markowa reprezentujących fonemy powstał w wyniku dodatkowej estymacji

trzeciego zestawu. Dodatkową estymację przeprowadzono na podstawie powstającej bazy akustycznej, posegmentowanej automatycznie z wykorzystaniem programu Aligner i trzeciego, bazowego kompletu modeli. Ostatni zestaw HMM wykorzystujący difony wytrenowany i estymowany został na podstawie bazy akustycznej Speecon (Marasek i Gubrynowicz, 2004), która zawiera zarejestrowane wypowiedzi 600 mówców, a konkretniej godzinę nagrań każdego z 600 mówców. Należy zauważyć, iż w przypadku modeli difonów trening w całości odbywał się na podstawie nieposegmentowanych nagrań.

Wraz z modelami HMM autorowi opracowania przekazane zostały wyniki przeprowadzonego testu, sprawdzającego poziom ich rozpoznawalności. Test przeprowadzony został z wykorzystaniem narzędzi HTK, a ściślej programu HResults. Program ten umożliwia porównanie sekwencji rozpoznanych jednostek mowy z transkrypcją nagrania zapisaną w pliku i określenie jaki procent zdań (całych wypowiedzi) i osobno pojedynczych wyrazów zostało poprawnie rozpoznanych. W przypadku zdań (wypowiedzi) poziom rozpoznawalności jest to wyrażony w procentach stosunek wszystkich zdań, do zdań, dla których rozpoznana sekwencja znaków odpowiadała transkrypcji fonetycznej. W przypadku pojedynczych wyrazów program porównując rozpoznaną sekwencję jednostek z transkrypcją fonetyczną liczy odległość między tymi dwoma ciągami znaków (odległość Levenshteina). Odległość ta wyrażona jest w liczbie operacji wstawienia, usunięcia lub zamiany symboli (jednostek, znaków), które należy wykonać, aby rozpoznana sekwencja jednostek była identyczna z transkrypcją fonetyczną danego nagrania. Natomiast wynikowy poziom rozpoznawalności jest to wyrażony w procentach stosunek wszystkich symboli (jednostek, etykiet) w transkrypcji, do symboli, które zostały poprawnie rozpoznane (nie wykonano na nich przytoczonych wcześniej operacji). Dlatego rozpoznawalność na poziomie zdań jest niższa niż na poziomie pojedynczych słów (pojedynczy błąd nie wyklucza całego zdania). Do testowania użyto 125 wypowiedzi o tematyce związanej z komputerami. Wyniki testu przedstawione zostały w tabeli 4.3.1.1.

Modele HMM	**Rozpoznane słowa (%)**	**Rozpoznane zdania (%)**
(fonemy) ramka 5ms, pobierana co 1ms	38,14	33,06

(fonemy) ramka 15ms, pobierana co 5ms	71,47	55,65
(fonemy) ramka 25ms, pobierana co 10ms	93,27	89,52
(fonemy) ramka 25ms, pobierana co 10ms, dodatkowa estymacja na bazie akustycznej	92,95	89,52
(difony) ramka 25ms, pobierana co 10ms	71,79	53,23

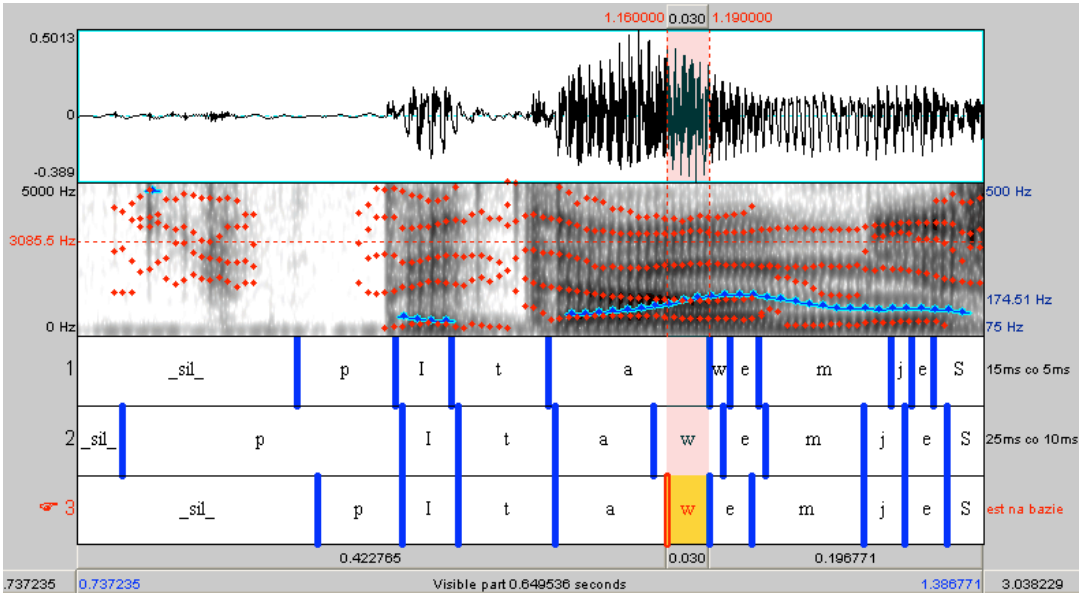
Tabela 4.3.1.1: Porównanie poziomu rozpoznawalności różnych modeli HMM.

Należy zaznaczyć, iż poziom rozpoznawalności danego zestawu modeli HMM nie określa poprawności wyznaczanych przez niego granic jednostek akustycznych. Można spodziewać się, iż modele, które osiągnęły w teście zdecydowanie wyższy wynik niż inny zestaw modeli będą popełniać mniej błędów, jednak kryterium doboru modeli HMM oparte o wyniki testu może być zawodne. Przykładowo, najlepszy wynik w postaci 93,27% prawidłowo rozpoznanych słów, uzyskano dla trzeciego zestawu modeli fonemów o ramce 25ms i kroku 10ms. Poziom rozpoznawalności niższy o 0.34%, w przypadku czwartego zestawu nie wykluczał jednak pewnej adaptacji do konkretnego mówcy i warunków nagrań, co powinno przełożyć się na mniejszą ilość błędów w segmentacji. Kryterium wyboru modeli na użytek projektu nie była rozpoznawalność jednostek mowy, lecz prawidłowe oznaczenie granic jednostek akustycznych w oparciu o transkrypcję fonetyczną, w konkretnych nagraniach konkretnego mówcy. Dlatego postanowiono w ramach niniejszego projektu przeprowadzić dodatkowo porównanie automatycznej segmentacji fragmentów bazy akustycznej, wygenerowanej z wykorzystaniem różnych kompletów modeli HMM. W teście tym postanowiono pominąć pierwszy komplet o ramce 5ms, pobieranej co milisekundę, ze względu na zbyt niski poziom rozpoznawalności (38,14%). Wygenerowanie granic segmentów w oparciu o różne zestawy modeli wymagało oprócz zamiany samych modeli, każdorazowej zmiany konfiguracji programu Aligner, a ściślej narzędzi HTK. W przypadku difonów dodatkowo zamiany wymagał plik „model.mlist", zawierający listę jednostek akustycznych (w tym przypadku listę difonów). Konfiguracja polegała na ustawieniu wartości dwóch parametrów w pliku „hcopy.conf". Mianowicie zmiennej „TARGETRATE" odpowiadającej za odstęp pomiędzy początkami kolejnych pobieranych ramek sygnału oraz „WINDOWSIZE" określającej ich rozmiar.

Fragment pliku „hcopy.conf” definiujący rozmiar i przesunięcie ramki (25ms, co 10ms):

```
#windowing
TARGETRATE      = 100000
WINDOWSIZE      = 250000
```

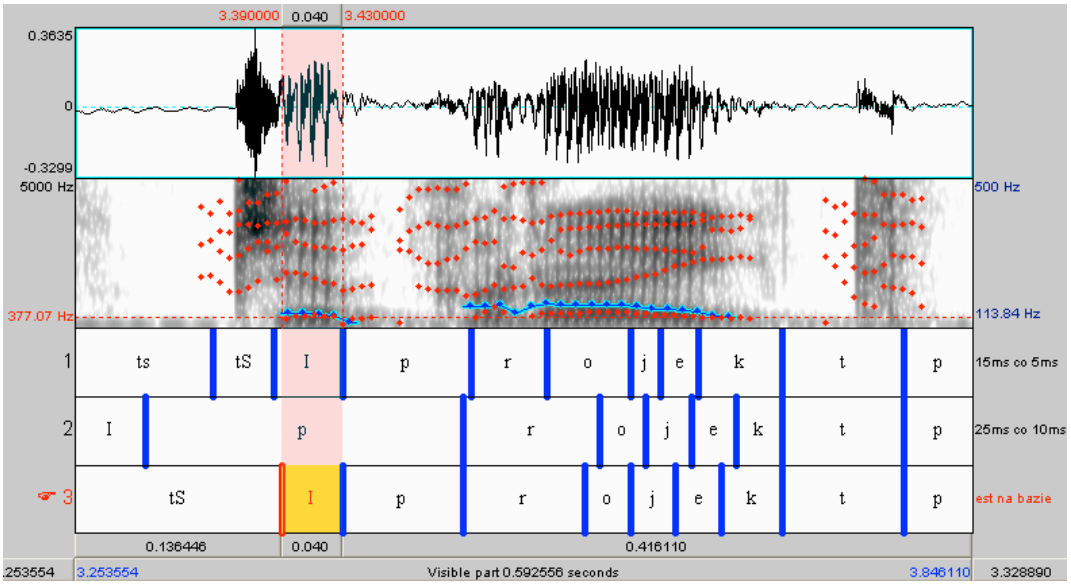
Porównanie poprawności segmentacji różnych kompletów modeli Markowa przeprowadzono w programie Praat. Umożliwia on umieszczenie w jednym oknie programu kilku wersji (warstw, poziomów) segmentacji wraz z przebiegiem czasowym i częstotliwościowym (spektrogramem), co znacznie ułatwiło cały proces. Najpierw porównano segmentację wygenerowaną przez różne modele fonemów i wybrano najlepsze z nich. Następnie, aby obiektywnie porównać wybrane wcześniej modele fonemów z modelami difonów, wymagane było przekonwertowanie segmentacji opartej na difonach do postaci opartej na fonemach. Jak już wspomniano, difon który określa przejście między dwoma fonemami, zaczyna się w połowie jednego, a kończy w połowie następnego. Dlatego, aby przekonwertować difony na fonemy, wystarczyło przeciąć każdy z nich w połowie i usunąć granice między sąsiadującymi ze sobą difonami. Powstałe w ten sposób przekształcenie difonów na fonemy nie zawsze jest idealne, jednak było wystarczające na potrzeby porównania automatycznej segmentacji z wykorzystaniem obu tych jednostek akustycznych. Poniżej przedstawiono „zrzuty ekranu” wykonane podczas porównań. Rysunki 4.3.1.2 i 4.3.1.3 ilustrują fragmenty segmentacji różnych kompletów modeli fonemów. Na obu rysunkach kolejne wersje segmentacji to zaczynając od góry: modele o ramce 15ms i kroku 5ms, modele o ramce 25ms i kroku 10ms oraz te same modele o ramce 25, pobieranej, co 10ms, dodatkowo estymowane na bazie akustycznej.



Rysunek 4.3.1.2: Porównanie segmentacji opartej na modelach (HMM) fonemów.

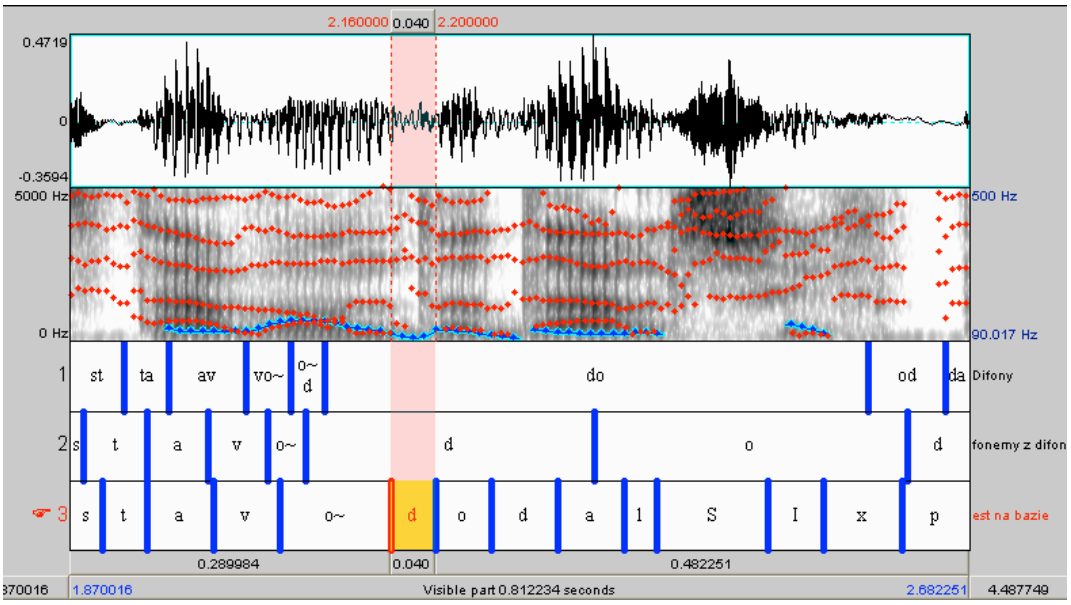
Zgodnie z wynikami testu rozpoznawalności najmniejszą poprawnością w określaniu granic fonemów wykazały się modele o ramce 15ms, pobieranej co 5ms, pomimo ich teoretycznie największej precyzji (rzędu 5ms). W ich przypadku najczęściej zdarzało się, iż dany fonem w całości znajdował się poza wyznaczonymi granicami. Wybór pomiędzy pozostałymi dwoma kompletami nie był tak oczywisty. Przykładowo, trudne w ocenie okazały się głoski płynne (/l/, /r/, /j/, /w/), których granice czasem nie są wyraźne i nawet ręcznie trudno jest je wyznaczyć. Dość często i niezależnie od zestawu modeli HMM, głoski te zostawały błędnie oznaczone (szczególnie w połączeniu z samogłoskami), dlatego trudno było wskazać zestaw modeli, który lepiej oznaczał ich granice. Generalnie w wielu sytuacjach podstawowe modele wykazywały większą precyzję, niż te po dodatkowej estymacji. Powodem była w pełni automatyczna segmentacja materiału wykorzystanego do dodatkowych obliczeń. Brak dodatkowej estymacji podstawowych modeli, oznaczający też brak adaptacji do konkretnego mówcy, powodował jednak częstsze występowanie poważnych błędów niż miało to miejsce w przypadku drugiego rozważanego kompletu. Problemem były np. sytuacje, w których szum związany z oddechem mówcy uznawany był za początek wyrazu i całe fonemy wyznaczane były w miejscu ciszy, co pociągało za sobą kolejne błędy. Dlatego pomimo dokładności rozpoznawania niższej o 0.34%, od uzyskanej dla modeli podstawowych, do dalszych prac nad segmentacją wybrano modele estymowane dodatkowo na powstającej bazie (ramka 25ms, krok 10ms). Ich dużą zaletą była największa spośród

wszystkich zestawów modeli przewidywalność i systematyczność popełnianych błędów, co ułatwiało ich wyszukiwanie i w wielu przypadkach ich korektę.

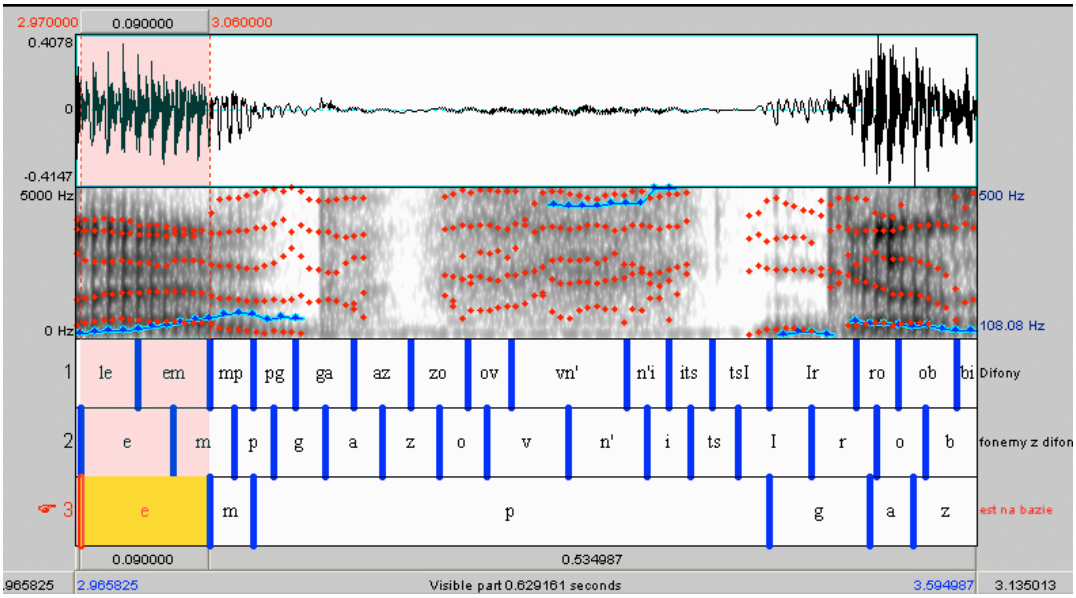


Rysunek 4.3.1.3: Porównanie modeli HMM fonemów inny przykład.

Porównanie segmentacji opartej na wybranych wcześniej modelach fonemów z wersją modeli fonemów wyznaczonych z modeli difonów potwierdziło przytoczone wcześniej wyniki testu rozpoznawalności poszczególnych zestawów modeli. Precyzja modeli difonów okazała się niższa, niż w przypadku drugiego porównywanego kompletu. Ponadto, w segmentacji opartej na modelach difonów częściej występowały poważne, nawarstwiające się błędy. Powodami mogły być: zbyt mała baza danych treningowych, trening w całości na podstawie nieposegmentowanych nagrań, czy też brak adaptacji do konkretnego mówcy i warunków nagrań. Dlatego ostatecznie wybrane zostały modele HMM fonemów estymowane na bazie akustycznej. Tym samym modele te zostały wykorzystane przy wstępnej automatycznej segmentacji. Porównanie segmentacji modeli fonemów z modelami difonów obrazują rysunki 4.3.1.4 i 4.3.1.5. Na ilustracjach tych kolejne wersje segmentacji to od góry: modele difonów, difony przekonwertowane na fonemy oraz wybrane wcześniej modele fonemów. Oba rysunki pokazują skrajne błędy, jakie zdarzyły się modelom difonów, przy czym rysunek 4.3.1.5 bardzo dobrze wskazuje jak wpływa SNR (*Signal to Nosie Ratio* - stosunek sygnału do szumu) na dokładność segmentacji.



Rysunek 4.3.1.4: Porównanie modeli fonemów z modelami difonów.



Rysunek 4.3.1.5: Porównanie modeli fonemów z modelami difonów inny przykład.

4.4. Korekta automatycznej segmentacji

Wstępna segmentacja nagrań, wygenerowana z wykorzystaniem programu Aligner oraz wybranych wcześniej modeli Markowa, stanowiła podstawę dla głównego etapu praktycznej części niniejszej pracy. Etap ten polegał na korekcie błędów automatycznej segmentacji, a

ściślej identyfikacji, wyszukaniu oraz poprawieniu wszelkich błędów. Ze względu na bardzo dużą ilość materiału, na wstępie postanowiono znacznie zautomatyzować proces wyszukiwania błędów. Stało się to możliwe dzięki skryptom opracowanym przez dr Dominikę Oliver, napisanym w języku Perl oraz języku skryptowym programu Praat. Skrypty te udostępnione zostały na potrzeby projektu przez Krzysztofa Szklanny. Ich działanie polegało na obliczeniu czasu trwania każdego fonemu w nagraniach, wyliczeniu globalnych średnich i odchylenia standardowego dla różnych fonemów oraz wyszukaniu i wypisaniu wystąpień, których czas trwania znacząco odbiegał od wyliczonych wcześniej średnich (2x odchylenie standardowe). Taka metoda, choć jest dosyć prosta, okazała się w miarę skuteczna w wyszukiwaniu istotnych błędów w segmentacji, czy błędnej transkrypcji fonetycznej. W wyniku działania skryptów otrzymano listę ok. 4500 wykrytych fonemów o nienaturalnym czasie trwania. Kilka przykładowych wpisów przedstawiono poniżej. Kolejne kolumny to od lewej: numer nagrania, symbol fonemu oraz czas końca danego fonemu w nagraniu.

Przykładowe wpisy wygenerowanej listy błędów:

…

s1000.phones	k	6.21000 dur
s1002.phones	s'	5.31000 dur
s1003.phones	p	2.02000 dur
s1003.phones	dz'	2.83000 dur

…

Przyjmując, iż najistotniejsze błędy zostały zidentyfikowane przez skrypty, postanowiono ograniczyć ręczną korektę do nagrań, których fonemy znalazły się na liście prawdopodobnych błędów. Należy zaznaczyć, iż większość nagrań zawierała przynajmniej jeden fonem, który znalazł się na liście wygenerowanej przez skrypty. Procedura korekty polegała na weryfikacji oraz poprawieniu zarówno transkrypcji fonetycznej, jak i segmentacji całego nagrania, a przede wszystkim granic fonemu wskazanego przez skrypty. Proces ten szerzej opisano w następnym rozdziale, wraz z najczęściej napotykanymi błędami. Jedna trzecia nagrań poprawiona została przez Krzysztofa Szklanny. Ręczna korekta nie stanowiła jednak całości prac związanych z poprawą automatycznej segmentacji. Poniżej przedstawiono kolejne etapy,

uwzględnienie których miało także istotny wpływ na sposób, w jaki przeprowadzono ręczną korektę.

Warto w tym miejscu przytoczyć pomijany do tej pory problem. Otóż ze względu na przyszłe zastosowanie bazy akustycznej, czyli syntezę mowy, granice wszystkich fonemów powinny zostać wyznaczone w miejscu dodatniego przejścia sygnału przez zero (z minusa na plus). W przeciwnym przypadku mogłyby powstać trzaski w syntezowanej mowie, wynikające z braku ciągłości zmian amplitudy. Wstępna automatyczna segmentacja nie spełniała tego wymogu ze względu na wykorzystane modele HMM, a także brak odpowiednich mechanizmów w programie Aligner. Dokładność automatycznie wyznaczonych granic wynosiła 10ms i wynikała z częstotliwości pobierania kolejnych ramek sygnału podczas parametryzacji. Fakt ten zaobserwować można na przedstawionych powyżej przykładowych wpisach listy wygenerowanej przez skrypty, zawierających czas końca danego fonemu w nagraniu. Taka dokładność okazała się w wielu przypadkach wystarczająca do poprawnego wyznaczenia granic z punktu widzenia odsłuchu i obserwacji spektrogramu, nawet dla fonemów o czasie trwania zaledwie 40ms. Jednak z punktu widzenia wymogów syntezy mowy oznaczała potrzebę poprawienia granic każdego automatycznie wyznaczonego fonemu. Konieczność poprawienia granic każdego fonemu poddawała niestety w wątpliwość sens automatycznego wyszukiwania błędów, a co za tym idzie - pominięcia ręcznej korekty części nagrań. Dlatego jeszcze przed rozpoczęciem prac postanowiono, iż niezbędne będzie opracowanie skryptu dla programu Praat, który w sposób automatyczny przesunie granice wszystkich fonemów do najbliższego przejścia amplitudy przez zero. Ponadto uznano, iż proces ten przeprowadzony zostanie już po ręcznej korekcie nagrań wskazanych przez skrypty autorstwa dr Dominiki Oliver. Zaletą przyjętego harmonogramu prac był brak konieczności czasochłonnego wyznaczania granic w przejściu przez zero podczas ręcznej korekty, a także możliwość wprowadzenia innych automatycznych poprawek, na podstawie obserwacji automatycznej segmentacji. W ten sposób także nagrania, które nie podlegały ręcznej korekcie, mogły zostać poprawione, a ręczne poprawki w dość dużym zakresie ograniczone. Istotną wadą przyjętej kolejności zadań była konieczność dodatkowej weryfikacji i kolejnych ręcznych poprawek, dokonywanych po automatycznie wprowadzonych modyfikacjach. Dlatego postanowiono umieścić w przytoczonym wcześniej skrypcie mechanizmy, które pozwoliły w sposób automatyczny weryfikować wszystkie

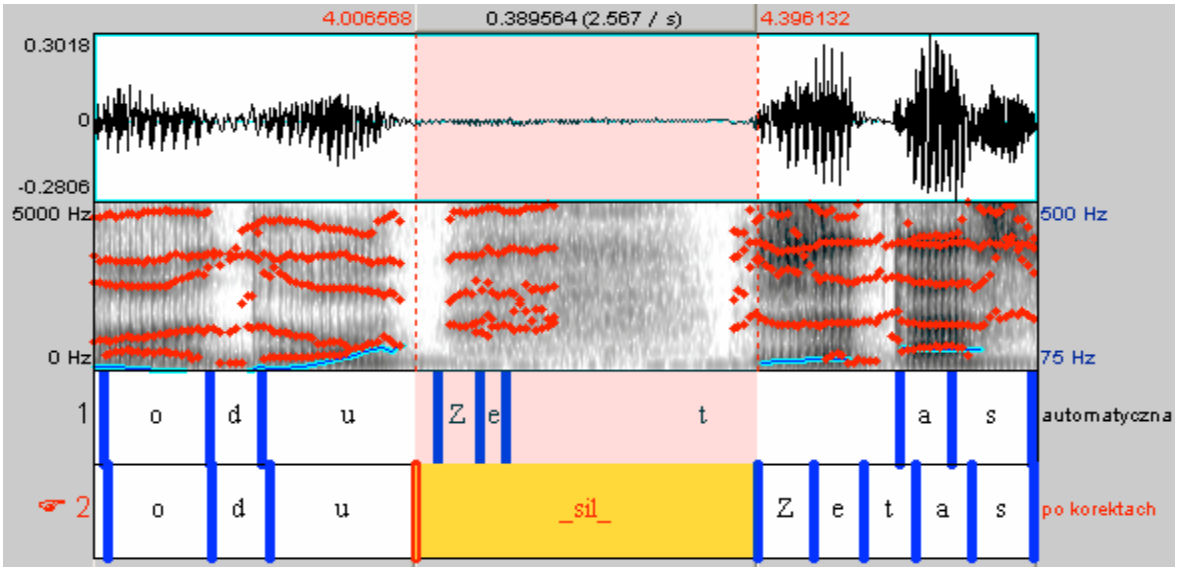
wprowadzane modyfikacje i raportować prawdopodobne lub pewne błędy. Głównym kryterium dla automatycznej weryfikacji była odległość, o jaką skrypt przesunął granice danego fonemu względem ich poprzedniego położenia. Im większe przesunięcie, tym większe prawdopodobieństwo zaistnienia błędu. Raport wygenerowany przez skrypt zawierał listę ponad 400 błędów oraz ponad 300 ostrzeżeń. Każdy z tych przypadków wymagał ręcznej weryfikacji oraz ewentualnych poprawek. Jednak w przeciwieństwie do przeprowadzonej wcześniej ręcznej korekty weryfikacja całego nagrania była bardzo pobieżna, a modyfikacje ograniczały się raczej do fonemu wskazanego przez skrypt. Więcej na temat wyznaczania granic fonemów w przejściu sygnału przez zero, skryptu korygującego oraz weryfikacji i korekty wprowadzonych przez niego zmian przedstawiono w rozdziale 4.4.2.

Podczas ręcznej korekty automatycznej segmentacji okazało się, że niewielka część nagrań zawierała dość silne zakłócenia sieci elektrycznej (niska częstotliwość, do 50Hz), powodujące miejscowe przesunięcia sygnału względem zera (brak przejścia sygnału przez zero). Sporządzono listę plików, w których zakłócenia te występowały w stopniu mogącym mieć wpływ na jakość syntezy mowy. Należy zauważyć, iż ręczna weryfikacja (korekta) nie dotyczyła wszystkich nagrań, jednak ze względu na miejscowe braki przejścia sygnału przez zero weryfikację pozostałych nagrań zapewnił wspomniany wcześniej skrypt korygujący. Zakłócenia sieci elektrycznej oraz filtrację nagrań, w których zostały one zlokalizowane omówiono szerzej w rozdziale 4.4.3. Segmentacja nagrań, w których znaleziono wspomniane zakłócenia, podlegała w całości ręcznej poprawie.

Poprawność segmentacji wstępnie zweryfikowano w udostępnionym syntezatorze, umożliwiającym wykorzystanie bazy akustycznej w jej ówczesnej postaci. Jego autorami są Danijel Korzinek oraz Łukasz Brocki. Wyznacznikiem poprawności, decydującym o zakończeniu omawianego etapu korekty, była jakość generowanej przez syntezator mowy, a ściślej - prawidłowe brzmienie jej poszczególnych „sklejanych" fragmentów. Poszukiwano błędnych segmentów oraz trzasków i innych zakłóceń, których źródłem mogły być niedokładnie wyznaczone granice fonemów. Proces wstępnej weryfikacji, wraz z testowym syntezatorem przedstawiono w rozdziale 4.4.4. Dodatkowa weryfikacja oraz kolejne korekty przeprowadzone zostały podczas dostosowywania bazy akustycznej do wymogów systemu Festival oraz syntezatora, co opisano szerzej w rozdziale 4.5. Końcowe testy, których

przedmiotem była ocena poprawności segmentacji, zrealizowano w prototypie korpusowego syntezatora powstającego w systemie Festival. Testom tym poświęcono rozdział 4.6.

Rysunek 4.4.1 przedstawia przykładowe porównanie automatycznie wygenerowanej segmentacji (góra) z wersją uwzględniającą opisane niżej etapy korekty (dół).



Rysunek 4.4.1: Porównanie automatycznej segmentacji oraz wersji po korektach.

4.4.1. Ręczna korekta błędów automatycznej segmentacji

Proces ręcznej korekty dotyczył nagrań wskazanych przez skrypty autorstwa Dominiki Oliver, które przedstawione zostały wcześniej. Segmentacja tych nagrań zawierała fonemy, których czas trwania znacząco odbiegał od wyliczonych wcześniej średnich (2x odchylenie standardowe). Omawiany etap prac obejmował weryfikację oraz korektę zarówno automatycznej segmentacji, jak i transkrypcji fonetycznej wspomnianych nagrań, a przede wszystkim fonemów wskazanych przez skrypty. Prace te wymagały dużej uwagi, gdyż stanowiły kluczową część całego projektu. To od nich w dużej mierze zależała przydatność wynikowej bazy akustycznej. Pewnym udogodnieniem był fakt, iż przeznaczeniem bazy jest korpusowa wersja syntezy mowy. W tym rodzaju syntezy generalnie wybierane są jak najdłuższe fragmenty zarejestrowanej mowy, co ogranicza ilość ich połączeń, a co za tym idzie ilość możliwych błędów spowodowanych nieprawidłową segmentacją. Pod uwagę brane są na ogół także czasy trwania łączonych ze sobą segmentów oraz ich intonacja, dzięki czemu dodatkowo maleje prawdopodobieństwo, iż w miejscu łączenia znajdzie się błędnie oznaczony fonem. Niedogodnością był natomiast brak możliwości modyfikacji łączonych fragmentów w przypadku korpusowej syntezy mowy w systemie Festival (multisyn). Oznaczało to, iż każdy ewentualny błąd segmentacji występujący w miejscu łączenia, który nie spowoduje odrzucenia danego segmentu przez funkcję kosztu, będzie słyszalny.

Rozważając proces segmentacji dużej bazy akustycznej przeznaczonej na potrzeby korpusowej syntezy mowy należy zauważyć, iż istotna w tym zastosowaniu jest konsekwencja w wyznaczaniu granic segmentów. Oznacza to, że granice tych samych fonemów (czy innych jednostek akustycznych) powinny być wyznaczane w możliwie taki sam sposób. Wspomniana konsekwencja ułatwia uzyskanie jednolitego brzmienia tych samych segmentów syntezowanej mowy oraz jej stałej, przewidywalnej jakości. Ponadto, nawet jeśli konsekwencja oznacza popełnianie tego samego błędu, czasem możliwe staje się opracowanie metod automatycznej korekty danej nieprawidłowości we wszystkich nagraniach. Przykładem może być przytoczone w poprzednim podrozdziale i szerzej opisane w następnym, automatyczne przesuwanie granic wszystkich fonemów do miejsc, w których są przejścia sygnału przez zero.

W związku z powyższym, podczas ręcznych korekt, a szczególnie na początku tego procesu, starano się zidentyfikować charakterystyczne błędy w automatycznej segmentacji,

które pojawiały się systematycznie. Działania te miały na celu jak najszybsze opracowanie listy wspomnianych błędów, umożliwiającej ograniczenie zakresu wprowadzanych w segmentacji poprawek, a także zbadanie możliwości automatycznej korekty niektórych z tych błędów. Potrzeba ograniczenia liczby nanoszonych poprawek wynikała z ogromu czasu, jakiego wymagałaby uważna korekta nagrań wskazanych przez skrypty (ok. 1400, czyli ok. 2/3 wszystkich promptów). Skrócenie średniego czasu potrzebnego na korektę jednego nagrania o minutę przekładało się na oszczędność prawie 24 godzin w ogólnym rozrachunku (ok. 1400 zaoszczędzonych minut). Należy też zauważyć, że bardzo dokładna korekta wszystkich błędów przyniosłaby umiarkowane korzyści, biorąc pod uwagę, iż 1/3 nagrań byłaby wykluczona z tego procesu. Dodatkowo ze względu na brak podobieństwa poprawionej w części nagrań segmentacji z wersją wygenerowaną automatycznie, jakość syntezy mowy byłaby zmienna i zależałaby od wykorzystanych nagrań. Dlatego postanowiono podczas nanoszenia ręcznych poprawek zachować pewną zgodność z automatycznie wygenerowaną segmentacją, co w wielu przypadkach oznaczało większą tolerancję na przybliżony niekiedy charakter wyznaczonych granic, a czasem nawet pozostawienie błędu. Częściowa zgodność z automatyczną segmentacją dotyczyła przede wszystkim najczęściej napotykanych błędów, które znalazły się na opracowanej liście oraz braku wyznaczania granic w przejściu przez zero. W ten sposób ograniczenie zakresu poprawek w niewielkim stopniu rekompensowała większa konsekwencja w segmentacji. Dodatkowo część zaoszczędzonego czasu mogła być poświęcona na próby opracowania metod automatycznej korekty wybranych błędów. Przykładem mogą być fonemy /p/, /t/, /k/ (plozyjne bezdźwięczne), które w automatycznej segmentacji prawie zawsze zaczynały się zbyt wcześnie, tzn. nie od ciszy, lecz od końcówki poprzedniego fonemu. Pomimo iż są to bardzo istotne błędy, ręcznie poprawiane były jedynie w skrajnych przypadkach, często do postaci zbliżonej do typowego zachowania wykorzystanych modeli HMM. W przypadku wspomnianych fonemów udało się zawrzeć w skrypcie korygującym mechanizmy, dzięki którym ich granice początkowe zostały automatycznie przesunięte tak, by znalazły się na początku ciszy. Najczęstsze błędy automatycznej segmentacji przedstawia tabela 4.4.1.1.

Całość prac związanych z ręczną korektą automatycznej segmentacji przeprowadzono w programie Praat, według zasad przedstawionych w rozdziale 1.4 oraz powyżej opisanych. Korekta 650 nagrań, czyli 1/3 wszystkich, przeprowadzona została przez Krzysztofa Szklanny.

Aby przyspieszyć prace, posługiwano się prostym skryptem, który po wpisaniu numeru nagrania otwierał okno z wskazanym nagraniem oraz jego segmentacją.

Największa część błędów wskazanych przez skrypty dotyczyła fonemów znajdujących się na końcach wypowiedzi lub dłuższych przerw między wyrazami, które nie były rozpoznawane i były dołączane do sąsiadujących fonemów, powodując ich nienaturalne wydłużenie. Należy zauważyć, iż nie były to najważniejsze błędy z punktu widzenia przyszłego zastosowania, a w przypadku ciszy dołączanej do ostatnich fonemów wypowiedzi, były wręcz mało istotne. Skrypty dobrze zlokalizowały najbardziej skrajne błędy, pomijały jednak część nieprawidłowości, niemożliwych do wykrycia na podstawie anormalnego czasu trwania fonemów. Niemniej dzięki opracowanym skryptom udało się w stosunkowo krótkim czasie wyeliminować największe nieprawidłowości, bez potrzeby dokładnej, ręcznej weryfikacji wszystkich nagrań.

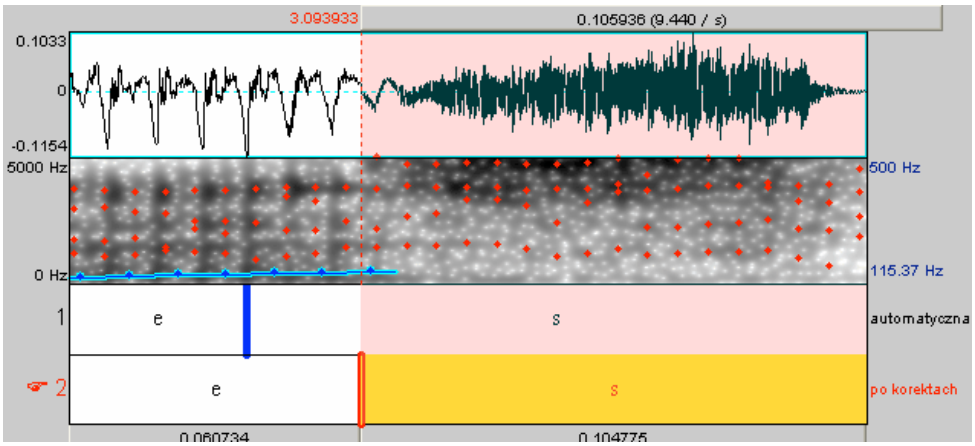
Jak się okazało transkrypcja fonetyczna zawierała dość dużo rozbieżności w stosunku do wymowy w nagraniach, w dużej mierze dotyczących obcojęzycznych słów, imion czy nazw własnych, których dość duża liczba znalazła się w korpusie. Zlokalizowano także przypadki różnej wymowy tych samych wyrazów wymagające poprawienia ich transkrypcji. Przykładem może być wyraz „mógłby", który wymawiany był zarówno /mugwbI/, jak i /mugby/ (obie formy są dopuszczalne). Błędy lub rozbieżności dotyczące transkrypcji fonetycznej poprawiane były zgodnie z wymową, poprawki nanoszono także w słowniku transkrypcji udostępnionym wraz z korpusem.

Przykłady kilku błędów oraz ręcznych korekt przedstawiono na rysunkach 4.4.1.2, 4.4.1.3 oraz 4.4.1.4. Na rysunkach tych górna wersja segmentacji przedstawia wersję wygenerowaną automatycznie, natomiast dolna uwzględnia wprowadzone korekty.

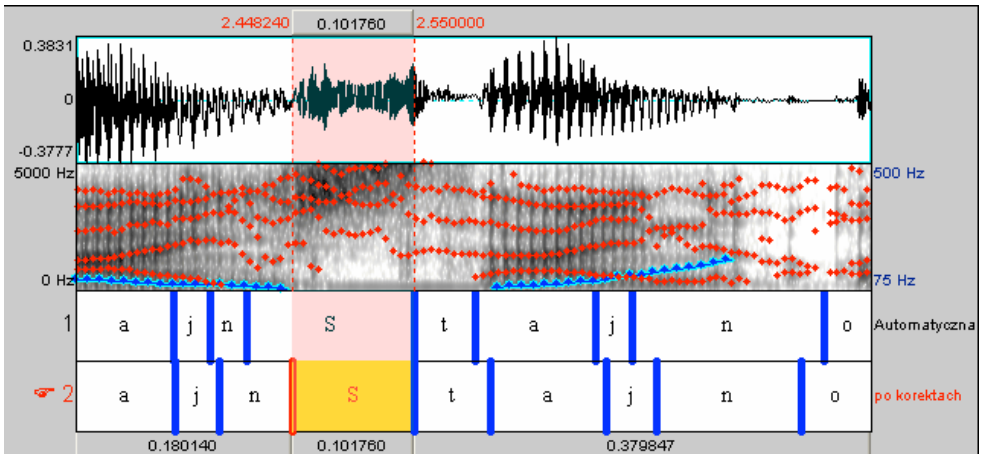
Fonemy których dotyczy	**Krótki opis typowych błędów**
/p/, /t/, /k/ (plozyjne bezdźwięczne)	- zaczynają się od części poprzedniego fonemu, a nie od początku ciszy

	- zbyt wczesne zakończenie - niekiedy wyznaczone granice obejmują koniec poprzedniego fonemu + cisza
/b/, /d/, /g/ (plozyjne dźwięczne)	- za krótki segment (bez dźwięcznej części)
plozyjne bezdźwięczne w połączeniu z trącymi (np. /pS/, /t S/, /ks'/, /ps/)	- niekiedy do spółgłoski plozyjnej dołączany jest fragment spółgłoski trącej
samogłoska w połączeniu z: /s/, /S/, /Z/, /z/, /s'/ (trące)	- część samogłoski nagminnie przydzielana jest do następującej po niej spółgłoski
/ts'/, /tS/, /ts/ (zwarto-trące), także w połączeniu z samogłoskami	- w wielu przypadkach granica początkowa segmentu wypada na końcówce poprzedniego fonemu - w połączeniu z samogłoskami koniec fonemu znajdował się w nagłosie samogłoski
/l/, /w/, /r/, oraz /v/	- często zbyt krótkie segmenty
dwa takie same fonemy jeden po drugim (głoski podwójne – geminaty) np. /rannI/	- jeśli nie były wyraźnie rozdzielone przez mówcę, prawie w całości oznaczane są jako jeden fonem, tym samym drugi zawiera jedynie niewielką pozostałą część
ostatni fonem w nagraniu	- na ogół do fonemu dołączany jest fragment ciszy
dłuższe przerwy międzywyrazowe	- cała cisza przydzielana jest do sąsiadujących fonemów

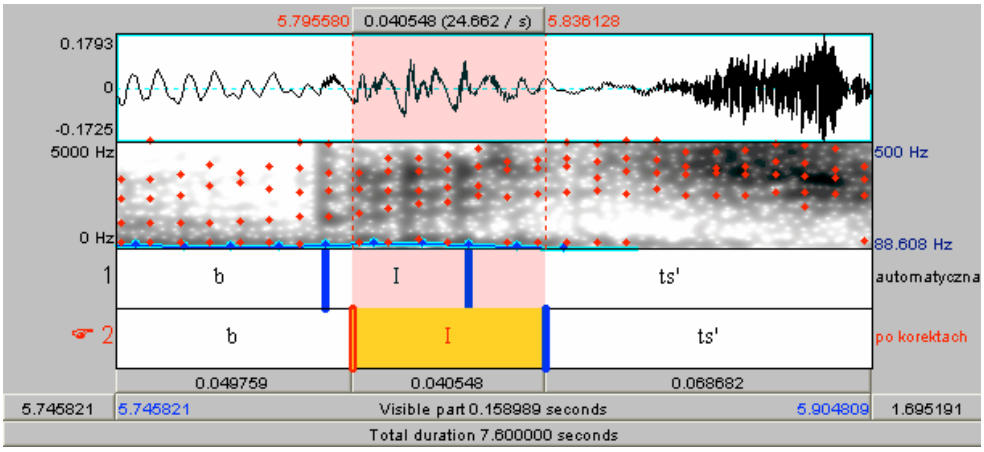
Tabela 4.4.1.1: Najczęstsze błędy automatycznej segmentacji.



Rysunek 4.4.1.2: Przykład korekty częstego błędu – samogłoska /e/ w połączeniu z trącą /s/.



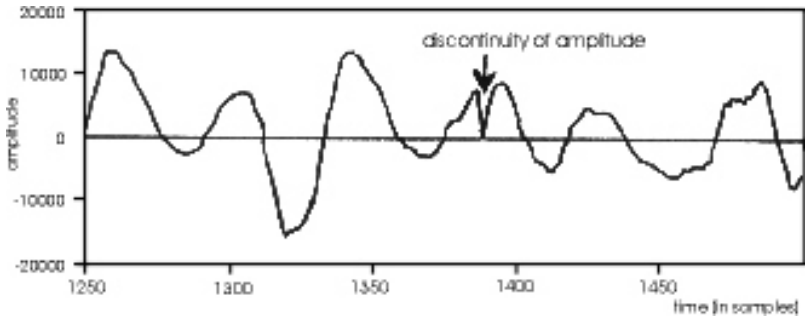
Rysunek 4.4.1.3: Przykład ręcznych korekt.



Rysunek 4.4.1.4: Inny przykład ręcznych korekt.

4.4.2. Opracowanie skryptu korygującego oraz weryfikacja jego działania

Decyzja o opracowaniu skryptu korygującego, przeznaczonego dla programu Praat, wynikła z potrzeby przesunięcia granic wszystkich fonemów do miejsc dodatniego przejścia sygnału przez zero. Potrzeba ta wynikała z wymogów syntezatora, podyktowanych brakiem modyfikacji łączonych fragmentów mowy. Gdyby granice fonemów nie znajdowały się w miejscach przejścia sygnału przez zero, podczas syntezy mowy mogłaby powstawać nieciągłość amplitudy i związane z nią trzaski. Przykład braku ciągłości amplitudy przedstawiono na rysunku 4.4.2.1. Dodatkowo, za powstaniem skryptu przemawiała potrzeba weryfikacji nagrań pominiętych podczas ręcznej korekty, pod kątem występowania silnych zakłóceń sieci elektrycznej oraz okazja do zbadania możliwości wprowadzenia automatycznych korekt wybranych błędów automatycznej segmentacji.

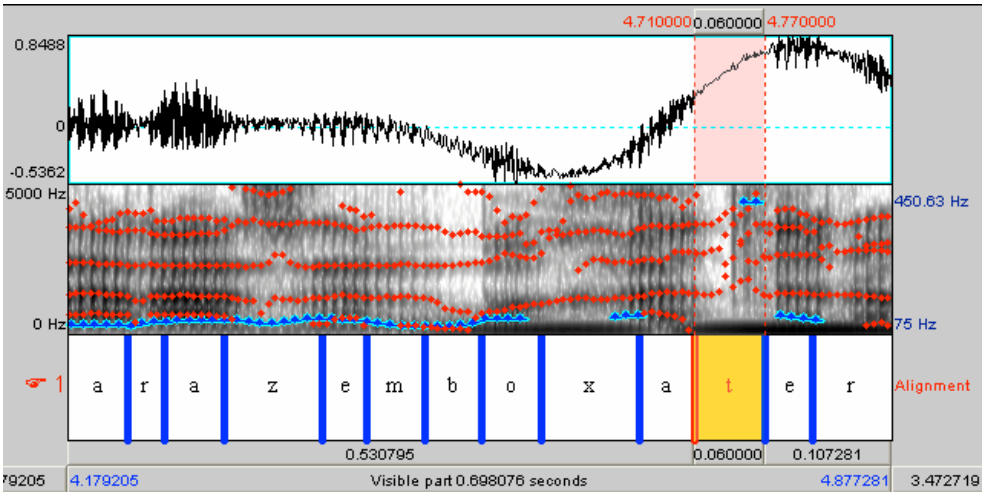


Rysunek 4.4.2.1: Przykład braku ciągłości amplitudy (w/g [2]).

Istotną przesłanką podczas tworzenia skryptu było wyeliminowanie potrzeby ręcznej weryfikacji wszystkich nagrań po automatycznie wprowadzonych korektach. Wymagało to zawarcia w skrypcie mechanizmów umożliwiających weryfikacje wprowadzanych korekt w sposób automatyczny i raportowanie niepewnych przypadków. Oznaczało to także poświęcenie dodatkowej ilości czasu na testowanie działania skryptu oraz ograniczenie korekt

do takich, które wnosiły niewielkie ryzyko wprowadzenia niezamierzonych błędów oraz które skrypt mógł poprawnie zweryfikować.

Proces powstawania skryptu podzielono na dwa etapy. Pierwszym było opracowanie algorytmu zapewniającego przesunięcie granic fonemów do dodatnich przejść sygnału przez zero oraz weryfikację wprowadzanych zmian. Etap ten obejmował także proces testowania. Za kryterium weryfikacji wprowadzanych przez skrypt korekt uznano odległość, o jaką skrypt zamierzał przesunąć granice danego fonemu względem ich poprzedniego położenia. Jeśli przesunięcie było większe niż wyznaczone ramy przypadek był raportowany, w skrajnych sytuacjach dodatkowo pozostawiana była dawna granica. Należy zauważyć, iż wymagana była weryfikacja nagrań, które nie podlegały ręcznej korekcie pod kątem występowania zakłóceń związanych z siecią elektryczną. Zakłócenia te objawiały się miejscowym brakiem przejścia sygnału przez zero, powodując na ogół próbę przesunięcia granic niektórych fonemów aż poza granice sąsiednich fonemów. Fakt ten czynił wspomniane zakłócenia łatwymi do zlokalizowania wobec przyjętych kryteriów weryfikacji. Przykład zakłóceń sieci elektrycznej przedstawiono na rysunku 4.4.2.2.



Rysunek 4.4.2.2: Przykład zakłóceń sieci elektrycznej (automatyczna segmentacja).

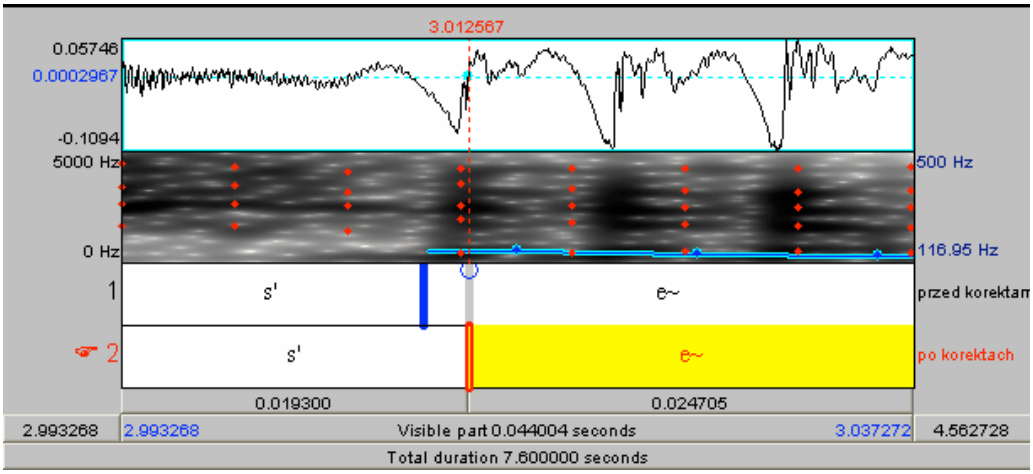
Drugi etap powstawania skryptu polegał na rozbudowaniu go o mechanizmy umożliwiające korektę niektórych wybranych błędów automatycznej segmentacji oraz o dodatkowe reguły weryfikacji. Etap ten poprzedzono badaniami możliwości automatycznego

wprowadzenia oraz weryfikacji różnych poprawek w segmentacji, dotyczących najczęstszych błędów, w oparciu o listę opracowaną podczas ręcznych korekt. Na podstawie przeprowadzonych badań i testów podjęto decyzję, iż automatyczna korekta dotyczyć będzie tylko fonemów /p/, /t/, /k/, które prawie zawsze zaczynały się od końcówki poprzedniego fonemu zamiast od ciszy, na ogół też za wcześnie się kończyły (czasem bez części wybuchowej). Są to jedne z najbardziej istotnych błędów na liście, gdyż zdarzające się przypadki wspomnianych fonemów, które zawierały tylko końcówkę poprzedniej głoski wraz z kawałkiem ciszy były nie do przyjęcia z punktu widzenia syntezy mowy. Istotna była też względna łatwość opisania, zaimplementowania oraz weryfikacji korekt wspomnianych fonemów. Należy zauważyć, że plozje bezdźwięczne poprzedzone są bardzo dużym spadkiem energii (cisza) tuż przed ich początkiem, po którym następuje duży skok energii (część wybuchowa - impuls) zakończony niekiedy krótkim szumem (przydech). Umożliwiło to przeprowadzenie korekty w dość prosty sposób, na podstawie śledzenia zmian poziomu energii. Pomimo iż omawiany etap powstawania skryptu dotyczył korekty zaledwie 3 fonemów, wymagał poświęcenia sporej ilości czasu na testowanie jego działania na automatycznie posegmentowanych nagraniach zawartych w bazie akustycznej. Konieczne było dobranie takich wartości parametrów, które zapewnią minimalizację ryzyka wprowadzania zbyt dużych modyfikacji, ale też zapewnią korektę istotnych przypadków. Problemem były tutaj duża ilość i zróżnicowanie nagrań, przekładające się na mnogość kontekstów, w jakich były użyte fonemy, ich różny czas trwania i różny poziom energii.

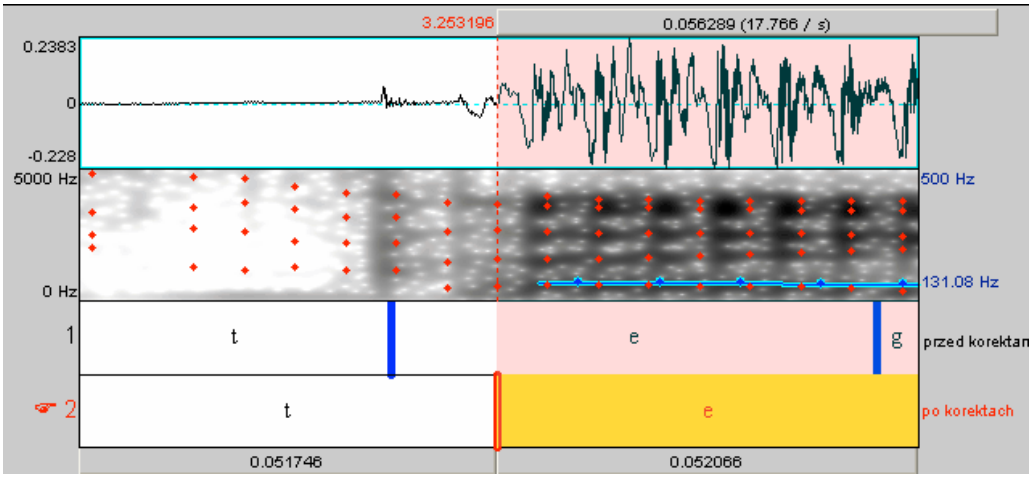
Podsumowując etap prac związany ze skryptem korygującym należy stwierdzić, iż osiągnięto wszystkie zamierzone cele automatycznej korekty. Granice wszystkich fonemów zostały przesunięte do miejsc pozytywnego przejścia sygnału przez zero, zlokalizowano pozostałe zakłócenia sieci elektrycznej, zapewniono korektę fonemów /p/, /t/, /k/, których dotyczyły istotne błędy oraz wyeliminowano potrzebę ręcznej weryfikacji wszystkich nagrań. W wyniku działania skryptu otrzymano listę ponad 700 fonemów wymagających ręcznej weryfikacji i ewentualnej korekty. Podczas weryfikacji wskazanych przez skrypt fonemów bardzo pobieżnie sprawdzane było także całe nagranie.

Na rysunku 4.4.2.3 przedstawiono przykład przesunięcia przez skrypt granicy fonemu do dodatniego przejścia sygnału przez zero. Natomiast przykład wprowadzonej przez skrypt korekty fonemu /t/ przedstawiono na rysunku 4.4.2.4.

Skrypt korygujący jest umieszczony na płycie dołączonej do niniejszej pracy, a jego kod źródłowy z komentarzami dołączono do opracowania w postaci załącznika.



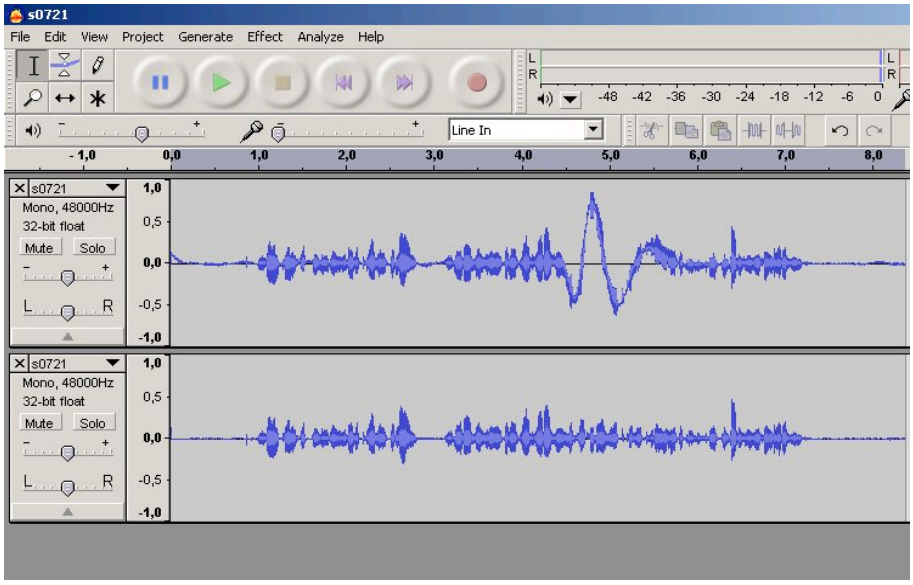
Rysunek 4.4.2.3: Przykład przesunięcia granicy do dodatniego przejścia przez zero.



Rysunek 4.4.2.4: Przykład korekty wprowadzonej przez skrypt.

4.4.3. Zakłócenia sieci elektrycznej (AC)

Podczas korekty automatycznej segmentacji, w niewielkiej części nagrań, zidentyfikowano dość silne zniekształcenia sygnału w zakresie częstotliwości 50 Hz, związane z działaniem sieci elektrycznej. W niektórych przypadkach mogły to też być zakłócenia pochodzące z otoczenia. Studio, w którym nagrano materiał, zlokalizowane jest na terenie uczelni, a nagrania przeprowadzono w trakcie roku akademickiego. Przykładem pochodzenia takich dźwięków otoczenia mogły być np. urządzenia o wirujących elementach, jak klimatyzacja czy suszarka do rąk. Zlokalizowane zakłócenia objawiały się zmiennym, oscylującym przesunięciem sygnału względem zera, powodując miejscami brak jego przejścia przez zero. Przykład omawianych zakłóceń ilustruje rysunek 4.4.2.2. Z pomocą skryptu korygującego sporządzono listę nagrań, w których one występowały. Przytoczone zakłócenia usunięto za pomocą filtracji górnoprzepustowej (*high-pass filter*) z częstotliwością graniczną 50Hz. Do tego zadania wykorzystano darmowy program Audacity[12]. Jest to edytor dźwięku umożliwiający między innymi wspomnianą filtrację. Na rysunku 4.4.4.1 przedstawiono okno programu Audacity zawierające przebieg sygnału z zakłóceniami oraz przebieg tego samego sygnału poddanego filtracji.



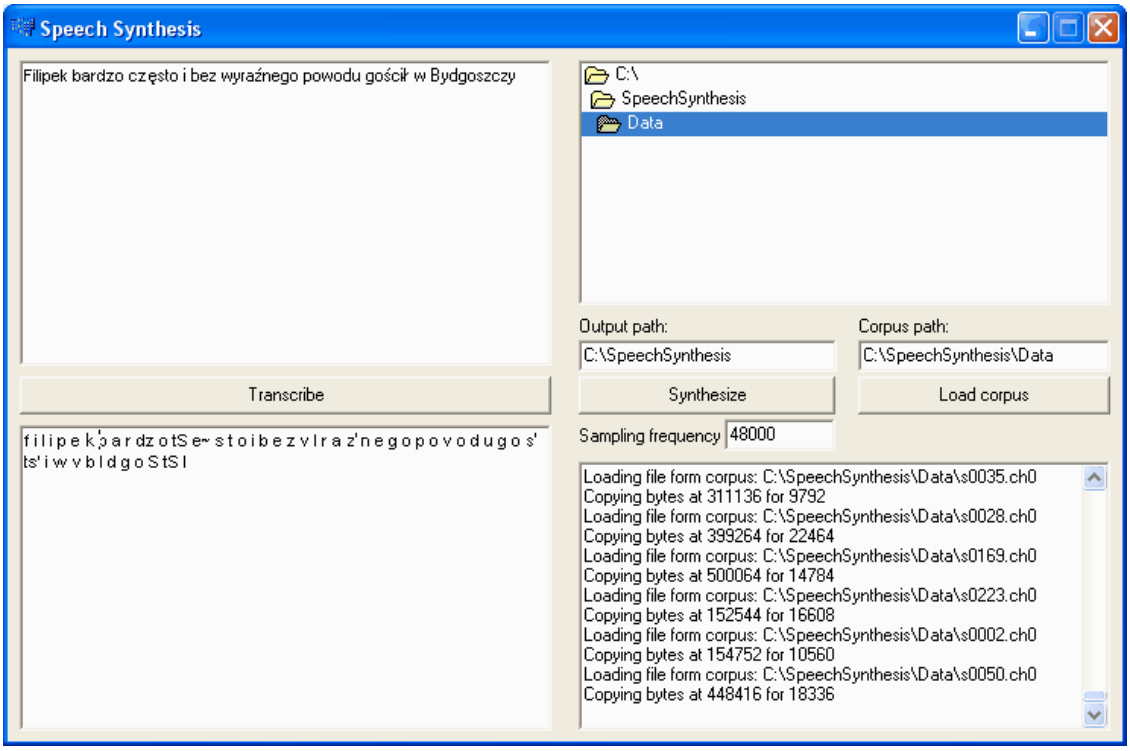
Rysunek 4.4.4.1: Filtracja zakłóceń sieci elektrycznej w programie Audacity.

[12] http://audacity.sourceforge.net/, 12-2007

4.4.4. Wstępna weryfikacja segmentacji w testowym syntezatorze

Przed rozpoczęciem procesu dostosowywania bazy akustycznej do potrzeb docelowego syntezatora postanowiono wstępnie zweryfikować poprawność segmentacji, a co za tym idzie ocenić jej przydatność. Warto zauważyć, że po dostosowaniu bazy dalsze wyszukiwanie błędów i wprowadzanie korekt w segmentacji wiązało się z utrudnieniami, choćby ze względu na inny format danych. Weryfikację przeprowadzono w syntezatorze, którego autorami są Danijel Korzinek oraz Łukasz Brocki, umożliwiającym wykorzystanie bazy akustycznej w jej ówczesnej postaci. Okno syntezatora przedstawiono na rysunku 4.4.5.1. Wyznacznikiem poprawności, stanowiącej warunek zakończenia etapu korekt, była ilość błędnych fonemów, zniekształceń, trzasków i innych zakłóceń w mowie syntetycznej, których powodem mogła być błędna segmentacja.

Procedura weryfikacji polegała na generowaniu oraz odsłuchu, czasem też obserwacji spektrogramu pojedynczych wypowiedzi, zaczerpniętych np. z różnych stron internetowych. Testy prowadzone były równolegle także przez autora powstającego korpusowego syntezatora mowy, a jego opinia miała decydujące znaczenie. Dodatkowo (głównie w celach porównawczych) wygenerowano część ze 100 zdań, które znalazły się w opracowanym korpusie testowym. Korpus ten powstał przede wszystkim na potrzeby końcowych testów, dlatego szerzej omówiony został w rozdziale 4.6.



Rysunek 4.4.5.1: Okno testowego syntezatora.

Należy zauważyć, że wykorzystany syntezator nie powstał z myślą o testowanej bazie akustycznej. By wygenerować zadaną wypowiedź, wybierał i łączył możliwie najdłuższe fragmenty zarejestrowanej mowy, nie brał jednak pod uwagę czasu trwania łączonych segmentów, ani też różnic w F0 (intonacja) pomiędzy łączonymi grupami jednostek akustycznych. Ze względu na dużą ilość nagrań w testowanej bazie akustycznej, a także zawarcie w niej nawet zdań pytających, przy wspomnianym kryterium doboru istniało duże ryzyko połączenia w procesie syntezy skrajnie różnych pod kątem czasu trwania czy intonacji fragmentów mowy, np. fragmentów zdania oznajmującego i pytającego. Oznaczało to, iż pewne zniekształcenia musiały się pojawiać niezależnie od poprawności segmentacji, utrudniając także jej weryfikację.

Początkowo brzmienie generowanej mowy wydawało się poniżej oczekiwań. W miejscach połączeń dość często występowały zniekształcenia, ponadto wyraźnie słyszalne były nagłe zmiany tempa wypowiedzi, a syntezowana mowa nie brzmiała wystarczająco naturalnie. Po dokładniejszym zbadaniu wielu przypadków okazało się jednak, że większość

zniekształceń nie wynikała z poprawności segmentacji, lecz z niedopasowanego do danych kryterium doboru łączonych fragmentów mowy. Jak już zostało wspomniane funkcja kosztu nie uwzględniała różnic w widmie i F0, brała pod uwagę jedynie długość łączonych fragmentów. Ostatecznie po wielu testach uznano, iż poprawność segmentacji jest wystarczająca do zakończenia etapu korekt, a czasem występujące przypadki zależnych od niej zniekształceń, były nie do uniknięcia, biorąc pod uwagę poziom automatyzacji prac.

4.5. Dostosowanie bazy akustycznej oraz ostatnie korekty

Istotnym celem niniejszej pracy było również dostosowanie posegmentowanej wcześniej bazy akustycznej do wymogów systemu Festival oraz powstającego korpusowego syntezatora mowy w taki sposób, aby można było ją wykorzystać w procesie syntezy mowy. Wymogi dotyczyły formatu danych, sposobu notacji transkrypcji fonetycznej, a także słownika transkrypcji wyrazów, które nie podlegały przyjętym regułom fonetycznym. Proces dostosowywania bazy akustycznej obejmował zakresem także dodatkową weryfikację segmentacji, a szczególnie transkrypcji fonetycznej, umożliwiając wprowadzenie ostatnich korekt.

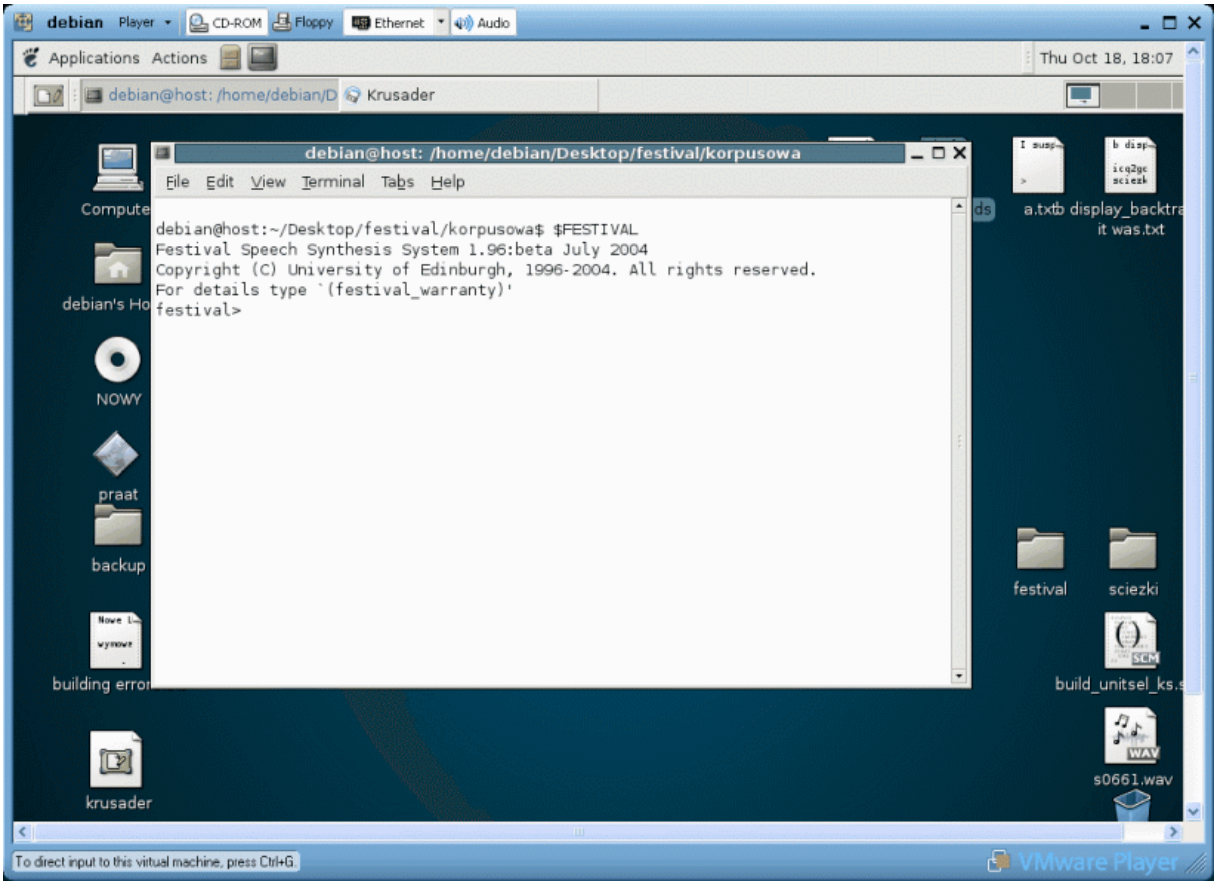
Na wstępie warto wyjaśnić, iż Festival jest to niekomercyjny, wielojęzyczny metasystem syntezy mowy, opracowany w Centrum Rozwoju Technologii Mowy (CSTR - *The Centre for Speech Technology Research*) na uniwersytecie w Edynburgu. Udostępniany jest on na licencji typu X11, dopuszczającej jego bezpłatne, nieograniczone wykorzystywanie, zarówno w celach niekomercyjnych, jak i komercyjnych. Festival wraz z narzędziami FestVox oraz biblioteką Speech Tools stanowi modularny szkielet, w którym można budować własne syntezatory mowy (konkatenacyjne, korpusowe, statystyczne - HTS, a także rozwiązania hybrydowe). Dodatkową jego zaletą jest możliwość współpracy z systemem syntezy mowy

MBROLA[13]. Należy zauważyć, iż w 1998 roku na uniwersytecie w Edynburgu opracowane zostały przez dr Dominikę Oliver moduły dla systemu Festival, umożliwiające opartą na difonach konkatenacyjną syntezę mowy polskiej [6]. Festival powstał w języku C++, z myślą o środowisku Unix (zawarty jest nawet w niektórych dystrybucjach systemów linuxowych), istnieje jednak także wersja źródłowa możliwa do skompilowania w środowisku Windows. Program działa w trybie tekstowym i przyjmuje komendy w języku skryptowym Scheme (SIOD). Festival jest cały czas rozwijany i udoskonalany, co potwierdza zawarcie w aktualnej obecnie wersji 2.0 (a ściślej 1.96 beta) mechanizmów umożliwiających najnowszą wersję statystycznej syntezy mowy HTS, opartą na ukrytych modelach Markowa. Twórcami systemu Festival w obecnym kształcie są: Alan W. Black (CMU – *Carnegie Mellon University*), Rob Clark (CSTR), Korin Richmond (CSTR), Volker Strom (CSTR), Simon King (CSTR), Heiga Zen (*Nagoya Institute of Technology*), Paul Taylor (CSTR) oraz Richard Caley (CSTR).

Do przeprowadzenia omawianego procesu dostosowywania bazy akustycznej niezbędny był Festival z zainstalowanym prototypem korpusowego syntezatora mowy. W wyniku konsultacji i uzgodnień z autorem syntezatora - Krzysztofem Szklanny, na potrzeby prac przekazane zostały przez niego pliki maszyny wirtualnej[14] dla programu VMware Player, wyposażonej w system operacyjny Debian, w którym zainstalowany był Festival i prototyp syntezatora oraz inne niezbędne dane. Wspomniany VMware Player[15] jest darmowym programem stanowiącym „platformę wirtualizacji", odpowiedzialną za uruchamianie i obsługę maszyn wirtualnych (format OVF – *Open Virtual Machine Format*) oraz zarządzanie nimi (przydzielanie zasobów sprzętowych). VMware Player przewidziany jest zarówno dla środowiska Windows, jak i Linux. Bogaty wybór skonfigurowanych, wyposażonych w system operacyjny i gotowych do użycia maszyn wirtualnych (*virtual appliances*), udostępniony jest do pobrania na stronie producenta programu. Istotną zaletą przedstawionego oprogramowania jest możliwość używania dodatkowego systemu operacyjnego (lub kilku) równocześnie z podstawowym, na dodatek bez konieczności jego instalowania, czy też wydzielania dodatkowych partycji na dysku. Wynika to z faktu, że integralną częścią maszyny wirtualnej dla VMware Player są pliki, będące obrazem dysku z

[13] http://tcts.fpms.ac.be/synthesis/mbrola.html, 12-2007
[14] Maszyna wirtualna (*VM – Virtual Machine*) jest programem, który poprzez symulowanie działania rzeczywistego komputera, stanowi odizolowane środowisko uruchomieniowe dla innych programów. Może uruchamiać zarówno pojedynczą aplikację, jak i własny system operacyjny, a nawet inną maszynę wirtualną. Przez inne programy, czy komputery w sieci, nie jest odróżniana od prawdziwego komputera.
[15] http://www.vmware.com/products/player/, 12-2007

zainstalowanym już systemem operacyjnym. Oznacza to, iż przeniesienie takiego wirtualnego systemu na inny komputer w nienaruszonej postaci, wraz z zainstalowanym oprogramowaniem i wszystkimi danymi, wymaga jedynie przeniesienia plików maszyny wirtualnej i uruchomienia jej na innym komputerze z zainstalowanym programem VMware Player. Niestety, oznacza to także ograniczenie przestrzeni dyskowej dostępnej z poziomu wirtualnego systemu operacyjnego do rozmiaru plików z obrazem wirtualnego dysku. Wadą jest też stosunkowo wolne działanie takiego systemu operacyjnego, wynikające z faktu, iż jest on uruchomiony na maszynie wirtualnej, a ta z kolei jest programem w podstawowym systemie operacyjnym, co wymaga dużej mocy obliczeniowej. Przestrzeń dyskowa udostępnionej maszyny wirtualnej (8GB) okazała się jednak wystarczająca, a używany komputer (Pentium4 2,8GHz, 1GB RAM) zapewniał także wystarczającą wydajność wirtualnego systemu do przeprowadzenia w nim procesu dostosowywania bazy akustycznej. W związku z powyższym całość prac związanych z tą częścią projektu przeprowadzona została z wykorzystaniem programu VMware Player oraz udostępnionej maszyny wirtualnej, wyposażonej w system operacyjny Debian, metasystem Festival oraz prototyp polskiego korpusowego syntezatora mowy. Rysunek 4.5.1 przedstawia okno programu VMware Player z działającym systemem operacyjnym Debian oraz terminalem, w którym uruchomiono system Festival.



Rysunek 4.5.1: Okno programu VMware Player, uruchomiony system Debian oraz Festival.

Pierwszym etapem dostosowywania bazy akustycznej było przekonwertowanie plików w formacie programu Praat (rozszerzenie Textgrid) opisujących granice fonemów do formatu, który mógł być odczytany przez system Festival (format xwaves - rozszerzenie lab). Była to prosta konwersja, ponieważ oba formaty są plikami tekstowymi o ściśle określonej strukturze, przy czym docelowe pliki lab w kolejnych wierszach zawierają jedynie czas rozpoczęcia kolejnego fonemu w danym nagraniu, pewną stałą liczbę oraz symbol danego fonemu. Dodatkowo dostosowana została na tym etapie notacja transkrypcji fonetycznej do wymogów syntezatora, a mianowicie znak ‘ (np. fonem /s’/, czy /dz’/), zamieniany był na znak + (odpowiednio /s+/, /dz+/). Ponadto zamieniany był używany wcześniej symbol _sil_ oznaczający ciszę, na symbol #, który przyjęto jako jej nową reprezentację. Konwersja zrealizowana została w sposób automatyczny przez Krzysztofa Szklanny, przy pomocy skryptu dla programu Praat, a wynikowe pliki o rozszerzeniu lab umieszczone zostały na udostępnionej maszynie wirtualnej i przekazane wraz z nią. Przekazany został także

zmodyfikowany odpowiednio korpus językowy, zawierający zapis ortograficzny wszystkich nagranych wypowiedzi. Modyfikacje dotyczyły przyjętej konwencji zapisu kolejnych wypowiedzi oraz notacji polskich znaków diakrytycznych, które nie są obsługiwane w systemie Festival. Przyjętą notację polskich znaków przedstawia tabela 4.5.2, natomiast konwencję zapisu kolejnych wypowiedzi ilustruje kilka przedstawionych niżej przykładowych wierszy dostosowanego korpusu.

Polski znak	ą	ę	ó	ł	ń	ć	ś	ź	dź	dż	ż
Nowy symbol	o~	e~	u	l/	n~	c~	s~	z~	dz~	dz*	rz

Tabela 4.5.2: Notacja polskich znaków diakrytycznych przyjęta w syntezatorze i korpusie.

Przykładowe wiersze zmodyfikowanego korpusu:

(s0001 "jak pan jo~ ocenia jako celnik , jako osoba ktura nadzoruje sprawy celne i sprawy graniczne ")

(s0002 "czy dlatego rze dyrektorko~ jest pani glemp gazownicy robio~ co kilka miesie~cy jej prezent ")

(s0003 "pytal/em jeszcze dzis~ czy morze przyszl/a jakas~ odpowiedz~ ")

(s0004 "co be~dzie wtedy podstawo~ do dalszych prac czy projekt pierwotny czy to co wypracowal/a podkomisja ")

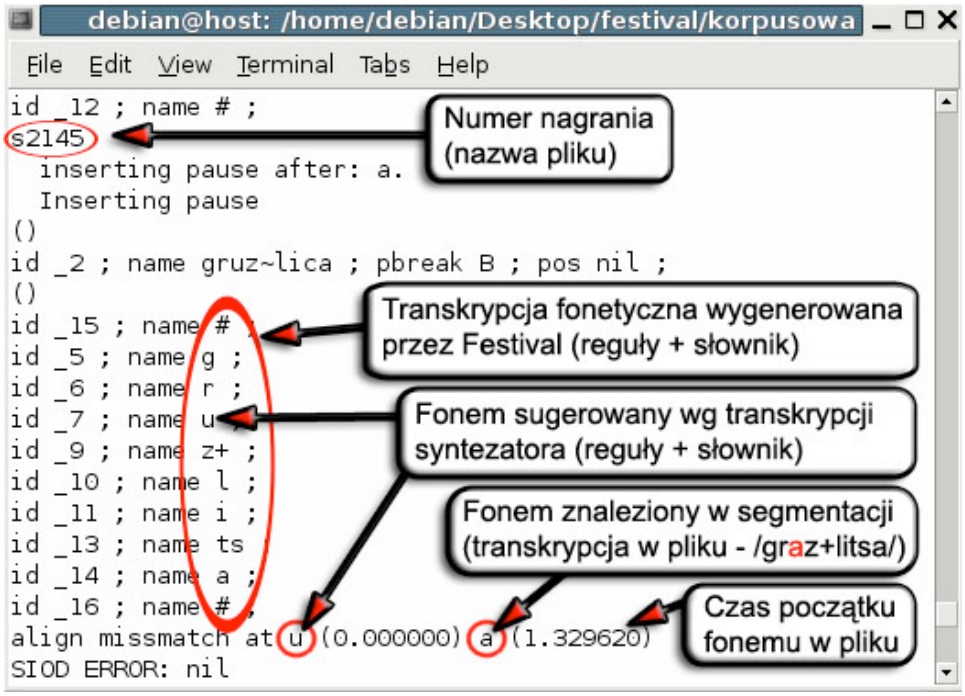
(s0005 "zapytajmy jak to jest w szwajcarii jak to jest w austrii ")

…

Otrzymane pliki lab oraz zmodyfikowany korpus stanowiły jedynie podstawę do dalszych prac, mających na celu wygenerowanie plików o rozszerzeniu utt, niezbędnych do wykorzystania bazy akustycznej w korpusowej syntezie mowy w systemie Festival (*multisyn voice*). Pomimo iż generowanie plików utt odbywało się w sposób automatyczny, z wykorzystaniem przekazanego wraz z innymi danymi skryptu dla systemu Festival, wiązało się jednak z szeregiem innych prac. Mianowicie wymagane było wprowadzenie drobnych poprawek w przygotowanym już wstępnie do wykorzystania korpusie językowym, wyeliminowanie ewentualnych pozostałych błędów w transkrypcji fonetycznej oraz zapewnienie jej pełnej zgodności z regułami syntezatora. Przy czym zapewnienie zgodności zapisu fonetycznego z regułami syntezatora wymagało dodania każdego wyrazu, który nie

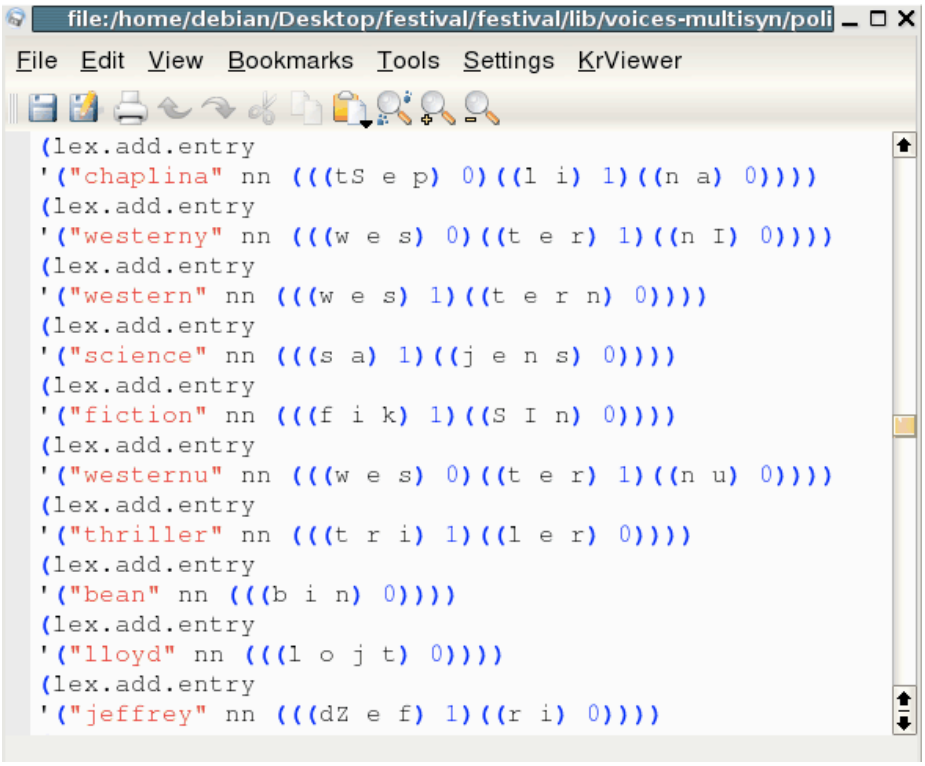
podlegał tym regułom, do słownika transkrypcji fonetycznej. Warto w tym miejscu wyjaśnić, iż skrypt generujący pliki utt posiadał mechanizm, powodujący zatrzymanie jego działania w przypadku napotkania niezgodności transkrypcji wygenerowanej przez syntezator (na podstawie korpusu, reguł i słownika) z transkrypcją w segmentacji (plikach lab). Zatrzymanie skryptu poprzedzone było krótkim raportem na temat napotkanej nieprawidłowości. Takie zachowanie zapewniało weryfikację poprawności zarówno korpusu, jak i jego zapisu fonetycznego, ułatwiając także lokalizację zwrotów, które należało dodać do słownika. Dodatkowo skrypt zatrzymywał się, gdy napotkał fonem, którego czas trwania był krótszy niż 10ms, stanowiąc także ostatni etap wyszukiwania błędów w segmentacji. W związku z powyższym omawiany etap prac polegał na wielokrotnym uruchamianiu skryptu oraz badaniu i korekcie lub też dodaniu do słownika każdego przypadku, który spowodował przerwanie jego działania, aż do przetworzenia wszystkich nagrań. Aby przyspieszyć przebieg prac i nie przetwarzać za każdym razem nagrań, dla których zostały już poprawnie wygenerowane pliki utt, korzystano z kopii korpusu, w której usuwano co jakiś czas przetworzone wypowiedzi, gdyż to one określały, które nagrania ma uwzględnić skrypt. Przykład raportu związanego z zatrzymaniem skryptu przedstawiono na rysunku 4.5.3. Przykładowy błąd został spreparowany na potrzeby prezentacji, tak aby dotyczył bardzo krótkiej wypowiedzi, co umożliwiło umieszczenie na ilustracji wszystkich istotnych informacji.

Pewnym problemem okazały się sporadyczne przypadki różnej wymowy tych samych słów, jak np. wyraz „mógłby” wymawiany w bazie akustycznej zarówno /mugwbI/ jak i /mugbI/. System Festival dopuszcza niestety tylko jedną wersję wymowy, w związku z czym wybierano najczęściej spotykaną wersję jako poprawną, natomiast w przypadku innych wersji modyfikowano zapis ortograficzny w korpusie, najczęściej tak, aby odpowiadał wymowie. Ewentualnie dodawano także wpis w słowniku definiujący transkrypcję zmodyfikowanego wyrazu.



Rysunek 4.5.3: Przykład raportu związanego z zatrzymaniem skryptu generującego pliki utt.

Zdecydowana większość przypadków, które spowodowały zatrzymanie skryptu dotyczyła wyrazów, które należało dodać do słownika określającego ich wymowę. Korpus językowy bazy akustycznej zawierał przede wszystkim wypowiedzi sejmowe oraz recenzje z gazet, dlatego znalazło się w nim wiele skrótów, imion, nazwisk, nazw własnych, wyrazów obcojęzycznych czy tytułów książek lub filmów o „nietypowej” wymowie, wymagających dodania do słownika jako wyjątki od przyjętych reguł transkrypcji fonetycznej. Na rysunku 4.5.4 przedstawiono zrzut ekranu z kilkoma przykładowymi wpisami dodanymi do słownika podczas generowania plików utt. Konwencja zapisu kolejnych wyrazów oraz ich transkrypcji wynika ze specyfiki języka Scheme oraz systemu Festival. Mianowicie każdy wpis w pliku jest komendą dodającą wyraz do słownika i ma następującą formę - (lex.add.entry ‘ (”zapis ortograficzny wyrazu” symbol części zdania (transkrypcja fonetyczna wyrazu z podziałem na sylaby i oznaczonym akcentem))). Przy czym symbol nn oznacza w przedstawionych przykładach rzeczownik (*noun)*, a zapis fonetyczny podzielony jest na sylaby, umieszczone w nawiasach wraz z oznaczeniem akcentu (w dodatkowych nawiasach) przyjmującym wartość 1 dla akcentowanej sylaby a 0 dla pozostałych. Wyrazy jednosylabowe zawsze zawierały oznaczenie akcentu o wartości 0.



Rysunek 4.5.4: Przykładowe wpisy dodane do słownika transkrypcji fonetycznej.

Należy zauważyć, iż moduły umożliwiające korpusową syntezę mowy polskiej w systemie Festival, wykorzystują reguły transkrypcji fonetycznej zaimplementowane i opisane przez dr Dominikę Oliver [6]. W trakcie dostosowywania bazy akustycznej do potrzeb syntezatora okazało się jednak, iż reguły te wymagały drobnych modyfikacji, gdyż musiały być w jak największym stopniu spójne z wymową mówcy. Zaobserwowane braki spójności zgłaszane były autorowi syntezatora, który nanosił stosowne poprawki w regułach. Przykładem zgłoszonej i naniesionej poprawki może być dodanie reguły określającej, że po fonemie /e/ lub /a/, litera „u” wymawiana jest jako /w/ (np. /ewro/, /awto/).

Wygenerowane pliki utt wraz ze słownikiem transkrypcji oraz poprawionym korpusem umieszczono w odpowiednich katalogach syntezatora w systemie Festival, by umożliwić wykorzystanie bazy akustycznej w procesie syntezy mowy. Następnie uruchomiono prototyp syntezatora w systemie Festival, a cała baza akustyczna została poprawnie wczytana, co było warunkiem zakończenia procesu dostosowywania bazy akustycznej i rozpoczęcia etapu testowania. W związku z powyższym wszystkie dane przekazane zostały także autorowi syntezatora w celu przeprowadzenia równoległych testów.

4.6. Testowanie (generowanie zdań testowych)

Ostatnim etapem niniejszej pracy było testowanie oraz ostateczna weryfikacja poprawności segmentacji, a co za tym idzie użyteczności gotowej bazy akustycznej w prototypie docelowego korpusowego syntezatora mowy polskiej w systemie Festival. Podstawą oceny był odsłuch wielu wygenerowanych zdań testowych. Natomiast wyznacznikiem poprawności była jakość generowanej przez syntezator mowy, a ściślej poprawne brzmienie poszczególnych łączonych fragmentów, szczególnie w miejscach ich połączeń. Podobnie jak podczas wstępnej weryfikacji poszukiwano błędnych segmentów oraz trzasków i innych zniekształceń, których powodem mogła być segmentacja.

Procedura testowania i weryfikacji polegała na wygenerowaniu oraz odsłuchu przede wszystkim 100 zdań, które znalazły się w omówionym poniżej korpusie testowym, ale także bardzo dużej ilości innych wybranych losowo pojedynczych zwrotów i zdań. Testy przeprowadzane były równolegle także przez autora powstającego syntezatora mowy. Opracowanie i wykorzystanie korpusu testowego miało na celu uzyskanie grupy zdań testowych o określonych parametrach, odpowiednio zróżnicowanych i odmiennych od tych, które znalazły się w bazie akustycznej. Posiadanie takiej stałej grupy zdań umożliwiło też porównanie ich z wersjami wygenerowanymi podczas wstępnej weryfikacji.

Podczas tworzenia korpusu istotne było zachowanie jego niewielkich rozmiarów, a zarazem pokrycie jak największej liczby różnych jednostek akustycznych oraz ich połączeń, a także dobór zdań o tematyce odmiennej od tych zawartych w bazie akustycznej. Ta odmienność miała zapewnić przetestowanie naturalności i zrozumiałości generowanych zwrotów występujących sporadycznie lub też wcale w korpusie przeznaczonym do syntezy. Rozmiar testowej bazy językowej postanowiono ograniczyć do 100 krótkich wypowiedzi (maksymalnie 60 fonemów). Za niewielkim rozmiarem przemawiał fakt, iż oprócz korpusu testowana była bardzo duża ilość przypadkowych zdań i zwrotów.

Korpus testowy opracowany został w programie CorpusCrt. Zdania postanowiono dobierać z trzech baz językowych, zawierających teksty z gazet o różnej tematyce, jednak bez wypowiedzi sejmowych oraz recenzji. Zanim przystąpiono do tworzenia korpusu testowego wymagane było połączenie i przygotowanie źródłowych baz językowych, czyli wygenerowanie transkrypcji fonetycznej każdego zdania w kodzie SAMPA dla fonemów, difonów oraz trifonów. Transkrypcja wygenerowana została z wykorzystaniem udostępnionych skryptów napisanych w języku Perl.

Kryteria doboru zdań podczas balansowania korpusu dotyczyły ich maksymalnej długości, ilości wystąpień różnych jednostek akustycznych, a co za tym idzie różnych konfiguracji głosek. Podczas balansowania korpusu określono, iż każdy fonem powinien w nim wystąpić przynajmniej 25 razy, a każdy difon i trifon powinien wystąpić co najmniej raz. Ze względu na niewielki rozmiar korpusu uzyskanie wszystkich difonów i trifonów było niemożliwe, jednak wspomniany warunek zapewnił ich możliwie dużą liczbę. Maksymalna długość dobieranych zdań określona została na 60 fonemów. Tabela 4.6.1 przedstawia, jak zmieniała się minimalna ilość wystąpień fonemów, a także ilość difonów i trifonów zależnie od maksymalnej długości zdań. Poniżej przedstawione zostało także kilka przykładowych zdań, które znalazły się w korpusie testowym.

Maksymalna długość zdań (liczba fonemów)	Minimalna ilość wystąpień dowolnego fonemu	Liczba difonów	Liczba trifonów
100	25	925	3400
80	25	870	3200
60	**25**	**820**	**2850**
50	najmniej 15, średnio 20	775	2500
40	najmniej 7, średnio 15	730	2150

Tabela 4.6.1: Porównanie statystyk korpusu testowego zależnie od długości zdań.

Przykładowe zdania z korpusu testowego:

…

Ćwiczy się również ratunkową ewakuację poprzez wyrzutnie torpedowe.

Hydrotelefon ma kilka źródeł zasilania.

Okręt nabrał wody w ciągu dwóch minut i piętnastu sekund.

Są wybuchowi, w zdenerwowaniu krzyczą, a za chwilę już się śmieją.

Wszedł do niej jako stały ingredient.

Boksy z inkubatorami będące schronieniem nowonarodzonych.

…

Warto zauważyć, że do wykorzystania korpusu do syntezy wymagane było dostosowanie notacji polskich znaków diakrytycznych wg notacji przedstawionej w tabeli 4.5.2.

Na podstawie odsłuchu zdań testowych zawartych w korpusie oraz wielu innych przypadkowych zdań, na podstawie których dokonywano oceny poprawności segmentacji, stwierdzono spełnienie stawianych przed nią wymogów. Pomimo iż w czasie przeprowadzania testów funkcja kosztu, jak i model intonacyjny były jeszcze dopracowywane, jakość generowanej mowy okazała się zadowalająca, a ilość zniekształceń i błędów względnie niewielka. Przy czym występujące zniekształcenia w przeciwieństwie do tych, które napotkano podczas wstępnej weryfikacji, w większości wynikały z segmentacji. Nie utrudniały one jednak zrozumienia, a syntezowana mowa brzmiała dość naturalnie. Podczas testów przeprowadzono także porównanie części zdań z korpusu, wygenerowanych podczas testów oraz podczas wstępnej weryfikacji. Różnica jakości była bardzo duża i potwierdzała wagę funkcji kosztu. Należy zauważyć, że wraz z dopracowaniem funkcji kosztu oraz modelu intonacyjnego syntezowana mowa powinna brzmieć lepiej niż podczas testów, a ilość nieprawidłowości powinna się zmniejszyć dzięki jeszcze lepszemu doborowi łączonych fragmentów mowy. Stwierdzono, iż pojawianie się czasem zniekształceń syntezowanej mowy było nieuniknione ze względu na stopień automatyzacji prac związanych z segmentacją. Dlatego ostatecznie segmentacja została zaakceptowana, co oznaczało, iż wynikowa baza akustyczna zostanie wykorzystana w końcowej wersji korpusowego syntezatora mowy.

5. Zakończenie

Przedmiotem opracowania była analiza całokształtu problematyki procesu segmentacji nagrań z wykorzystaniem mechanizmów automatycznego rozpoznawania mowy, w kontekście powstawania językowej bazy akustycznej.

Nadrzędnym celem pracy była segmentacja bazy akustycznej, przeznaczonej na potrzeby realizacji korpusowej syntezy mowy oraz dodatkowo - dostosowanie jej do wymogów systemu Festival i powstającego syntezatora. Celem była także częściowa automatyzacja prac z wykorzystaniem mechanizmów rozpoznawania mowy. Jako aparat matematyczny do rozpoznawania mowy, niezbędny do automatycznego wygenerowania granic segmentów, wykorzystano metody statystyczne oparte o ukryte modele Markowa (HMM). Założono, że automatycznie wygenerowana segmentacja, pomimo niewystarczającej z punktu widzenia możliwych zastosowań dokładności i zdarzających się błędów, stanowi optymalne rozwiązanie jako wstęp do dalszych prac nad dużą bazą nagrań. Ponadto przyjęto, iż korekta wyznaczonych w ten sposób segmentów może w pewnym stopniu zostać zautomatyzowana, umożliwiając przetworzenie dużej ilości materiału w rozsądnym czasie. Przez automatyzację korekty rozumie się tutaj zarówno automatyzację procesu wyszukiwania błędów, jak i poprawy niektórych z nich.

Wymogi stawiane przed projektem dotyczyły poprawności wyznaczonych granic segmentów oraz ich transkrypcji fonetycznej, a także formatu danych. Istotnym warunkiem było opracowanie słownika transkrypcji fonetycznej wyrazów, które stanowiły wyjątki od zaimplementowanych w syntezatorze reguł. Podstawą weryfikacji był odsłuch zdań testowych, wygenerowanych we wczesnej wersji docelowego syntezatora. Wyznacznikiem spełnienia przedstawionych wymogów była jakość generowanej przez syntezator mowy, a ściślej poprawne brzmienie jej poszczególnych łączonych fragmentów, szczególnie w miejscach łączeń. Poszukiwano błędnych segmentów oraz trzasków i innych zakłóceń, których źródłem mogła być segmentacja.

W ramach pracy:

- Przeprowadzono analizę oraz porównanie automatycznej segmentacji fragmentów bazy akustycznej, uzyskanej w oparciu o różne zestawy modeli HMM, celem wyboru najodpowiedniejszego.
- Wygenerowano automatyczną segmentację z wykorzystaniem udostępnionego programu Aligner oraz wybranych modeli HMM.
- Wygenerowano listę fonemów, których czas trwania odbiegał od wyliczonych średnich przynajmniej o podwojone odchylenie standardowe. Zadanie to zrealizowano przy pomocy udostępnionych skryptów napisanych w języku Perl oraz języku skryptowym programu Praat.
- Opracowano listę najczęstszych błędów w segmentacji oraz zakres ręcznych korekt, pozwalający utrzymać zgodność z automatycznie wygenerowaną segmentacją.
- Przeprowadzono ręczne korekty segmentacji oraz transkrypcji fonetycznej nagrań wskazanych przez skrypty w określonym wcześniej zakresie.
- Opracowano skrypt korygujący przeznaczony dla programu Praat. Korekty wprowadzane przez skrypt dotyczyły przesunięć granic fonemów do miejsc dodatnich przejść sygnału przez zero oraz kilku wybranych błędów. Opracowano także metodę weryfikacji działania skryptu, polegającą na wylistowaniu fonemów, w przypadku których granice zostały przesunięte o odległości większe niż zadana wartość progowa. Wspomniana metoda weryfikacji stanowiła zarazem mechanizm wyszukiwania istotnych zakłóceń sieci elektrycznej, powodującej miejscami brak poszukiwanych przejść sygnału przez zero.
- Przeprowadzono ręczną korektę granic fonemów, które zostały wskazane przez skrypt korygujący. Na tym etapie dokonano także pobieżnej weryfikacji całych nagrań, w których znalazły się fonemy wypisane przez skrypt.
- Usunięto zlokalizowane w niewielkiej części nagrań zakłócenia związane z częstotliwością pracy sieci elektrycznej (50 Hz) w procesie filtracji górnoprzepustowej, w programie Audacity.
- Opracowano korpus 100 zdań testowych przy pomocy programu Corpus CRT.

- Wstępnie zweryfikowano poprawność segmentacji w udostępnionym na potrzeby projektu testowym syntezatorze. Podstawą do weryfikacji było wygenerowanie losowych zdań testowych oraz zdań zawartych w opracowanym korpusie.
- Wygenerowano pliki „utt" w formacie systemu Festival, umożliwiające wykorzystanie bazy akustycznej do korpusowej syntezy mowy we wspomnianym systemie. Proces ten obejmował także dodatkową automatyczną weryfikację poprawności transkrypcji i w pewnym stopniu segmentacji oraz wprowadzenie ostatnich korekt.
- Przygotowano słownik transkrypcji fonetycznej w formacie systemu Festival (festvox) dla wyrazów, które nie podlegały przyjętym w syntezatorze regułom transkrypcji.
- Wygenerowano w prototypie polskiego korpusowego syntezatora mowy zdania zawarte w opracowanym korpusie testowym, będące przedmiotem oceny poprawności segmentacji.

Na podstawie odsłuchu przygotowanych zdań testowych oraz wielu innych losowo wybranych zdań stwierdzono spełnienie stawianych przed projektem wymogów. Weryfikację przeprowadził zarówno autor niniejszego opracowania, jak i Krzysztof Szklanny, autor powstającego korpusowego syntezatora mowy, będącego przeznaczeniem bazy akustycznej. Pomimo, iż w czasie przeprowadzania testów funkcja kosztu była jeszcze dopracowywana, jakość generowanej mowy okazała się zadowalająca, na poziomie zbliżonym do systemów komercyjnych. Należy zaznaczyć, iż mówca, który użyczył głosu podczas rejestrowania bazy akustycznej nie używał na co dzień głosu w sposób profesjonalny (zawodowy). W przypadku komercyjnych systemów nagrania wykonują zawodowi mówcy (aktorzy, speakerzy radiowi itp.), co wpływa w istotny sposób na jakość generowanej przez nie mowy syntetycznej. Natomiast pojawiające się zniekształcenia syntezowanej mowy były nieuniknione, biorąc pod uwagę stopień automatyzacji prac związanych z segmentacją. Potwierdzeniem sukcesu jest akceptacja i wykorzystanie zrealizowanej bazy akustycznej we wspomnianym syntezatorze.

Przeprowadzone badania, przebieg prac oraz testów wynikowej bazy akustycznej uzasadniają poprawność postawionych tez badawczych. Założenia te znalazły potwierdzenie w przebiegu prac, stopniu ich automatyzacji, a także spełnieniu wymogów dotyczących jakości segmentacji oraz akceptacji wynikowej bazy akustycznej i wykorzystaniu jej do korpusowej syntezy mowy. Automatyczna segmentacja okazała się wręcz niezbędna przy tak

dużej ilości nagrań, a dalsza automatyzacja prac związanych z korektą okazała się możliwa, a nawet wskazana, umożliwiając wyeliminowanie największych nieprawidłowości i błędów w stosunkowo krótkim czasie.

Podsumowując należy stwierdzić, iż proces segmentacji nagrań bazy akustycznej z wykorzystaniem mechanizmów rozpoznawania mowy jest złożony, obejmuje wiele zagadnień i aby osiągnąć zadawalający efekt, wymaga dużego nakładu pracy. Nakład ten jest jednak niewspółmiernie mniejszy niż w przypadku ręcznej segmentacji tak dużej ilości nagrań. Pomimo większej dokładności i mniejszej ilości „przeoczonych" błędów, ręczna segmentacja tak dużej ilości materiału jest nierealizowalna w rozsądnym czasie. Ponadto istotną zaletą automatycznie generowanej segmentacji okazała się duża konsekwencja w wyznaczaniu granic, niemożliwa do uzyskania przez człowieka. Należy spodziewać się, że wraz z rozwojem metod rozpoznawania mowy, jakość automatycznej segmentacji sprawi, iż ręczna obróbka sygnału stanie się całkiem zbędna.

Metody przedstawione w niniejszej pracy odpowiadają zapotrzebowaniu na duże poetykietyzowane bazy akustyczne z dokładnie wyznaczonymi granicami segmentów, przeznaczone przede wszystkim dla korpusowej syntezy mowy oraz systemów ASR.

Podziękowania

Serdeczne podziękowania chciałbym przekazać Krzysztofowi Szklanny. Przede wszystkim za pomoc w korekcie automatycznej segmentacji, przekazane materiały, poświęcony mi czas, niezliczoną ilość porad i wskazówek, a także słowa otuchy w chwilach zwątpienia. Równie gorąco chciałbym podziękować promotorowi prof. Krzysztofowi Maraskowi za cierpliwość, udostępnione materiały i wsparcie w moich dążeniach.

Bibliografia

[1] R. Gubrynowicz, Wykład Podstawy fonetyki akustycznej (PFA), 2004

[2] K. Marasek, Wykład Werbalna komunikacja z komputerem (WKK), 2005

[3] K. Marasek, Electroglottographic description of voice quality, Habilitationsschrift, Stuttgart: AIMS, 1997

[4] K. Szklanny, Przygotowanie bazy difonów języka polskiego dla realizacji syntezy mowy w systemie MBROLA, Praca magisterska, Warszawa: PJWSTK, 2002

[5] K. Szklanny, D. Oliver, Creation and analysis of a Polish speech database for use in unit selection speech synthesis, Genova: LREC Conference, 2006

[6] D. Olivier, Polish Text To Speech Synthesis, M.Sc. Speech and Language Processing, Edinburgh: University of Edinburgh, 1998

[7] R. Tadeusiewicz, Sygnał mowy, Warszawa: Wydawnictwo Komunikacji i Łączności, 1988

[8] R. Tadeusiewicz, Sieci neuronowe, Warszawa: Akademicka Oficyna Wydawnicza, 1993

[9] B. Wierzchowska, Fonetyka i fonologia języka polskiego, Wrocław: Ossolineum, 1980

[10] B. Wierzchowska, Opis fonetyczny języka polskiego. Warszawa: PWN, 1967

[11] L. Roudet, przekład T. Benni, Zasady fonetyki ogólnej, Warszawa, 1947

[12] D. Korzinek, Hybrydowy System Automatycznego Rozpoznawania Mowy w Języku Polskim, Praca magisterska, Warszawa: PJWSTK, 2007

[13] J. Tebelskis, Speech Recognition using Neural Networks, Pittsburgh: Carnegie Mellon University, 1995

[14] J. Moczko L. Kramer, Cyfrowe metody przetwarzania sygnałów biomedycznych, Poznań: Wydawnictwo naukowe, 2001

[15] J. Laver Principles of phonetics, Oxford: Oxford University Press, 1994

[16] D. Klatt, Review of text-to-speech conversion for English, J. Acoust. Soc. Am. 82, 1987

[17] X. Huang, A. Acero, H-W. Hon, Spoken Language Processing: A Guide to Theory, Algorithm and System Development, New Jersey: Prentice Hall PTR, 2001

[18] K. Tokuda, H. Zen, A. W. Black, An HMM-based speech synthesis system applied to English, IEEE SSW, 2002 - http://hts.sp.nitech.ac.jp/?Publications, 12-2007

[19] G. Demenko, Analiza cech suprasegmentalnych języka polskiego na potrzeby technologii mowy, Praca habilitacyjna, Poznań: Wydawnictwo Naukowe UAM, 1999

Załącznik A. Kod źródłowy skryptu korygującego

Skrypt w Praat

```
clearinfo
# Michal Wojtowski
# maindir - glowny katalog (w nim skrypt szuka pozostalych katalogow)
# sourcedir - katalog zrodlowy textgridow
# destdir - katalog docelowy textgridow
# wavdir - podkatalog z nagraniami (takie same nazwy jak textgridy)
# wavtype - rozszezenie nagran - raw, ch0

maindir$ = "C:\mgr\korekta\"
sourcedir$ = "textgrid"
destdir$ = "zero crossing"
wavdir$ = "wav"
wavtype$ = "raw"

# utworzenie listy nagran
Create Strings as file list...  list 'maindir$''wavdir$'\*.'wavtype$'
number_files = Get number of strings

errors   = 0
warnings = 0

# petla iterujaca po wszystkich nagraniach
for j from 1 to number_files

  # wczytanie nagrania i "textgridu" oraz ustawienie czestotliwosci probkowania na 48kHz
  select Strings list
  current_token$ = Get string... 'j'
  Read Sound from raw 16-bit Little Endian file... 'maindir$''wavdir$'\'current_token$'
  object_name$ = selected$ ("Sound")
  Override sampling frequency... 48000
  Read from file... 'maindir$''sourcedir$'\'object_name$'.TextGrid

    # pobranie liczby granic fonemow w danym "textgridzie"
    select TextGrid 'object_name$'
    fonemow = Get number of intervals... 1

    # petla iterujaca po kolejnych granicach fonemow w danym "textgridzie"
    for i from 1 to fonemow-1

      # pobranie symboli fonemow: aktualnego i nastepnego
      label0$ = Get label of interval... 1 i
      label1$ = Get label of interval... 1 i+1

      # pobranie pozycji granic fonemów: aktualnego i nastepnego
      # oraz poprzedniego (takze jego symbolu), jesli aktualny nie jest pierwszym
      test = Get end point... 1 i
      test2 = Get end point... 1 i+1
      test3 = 0

      if i > 1
        test3 = Get end point... 1 i-1
        label2$ = Get label of interval... 1 i-1
      endif
```

```
    select Sound 'object_name$'
    res = 0.00001
    skok = 0
    marker = 0
    counter =0
    offset = 0
    # tolerancja - okresla o ile moze ewentualnie zostac cofnieta granica fonemu
    tolerancja = 0.0025

    # ustawienie poczatkowego przesuniecia(offset) i tolerancji dla wybranych fonemow.
    if label0$ = "p" or label0$ = "t" or label0$ = "k" or label0$ = "r" or label0$ = "l" or label0$ =
"w" or label0$ = "v"

      offset = 0.001
      tolerancja = 0.001
    endif

    # petla while poszukujaca miejsca przeciecia zera spelniajacego zadane kryteria.
    # Warunkiem jest zmienna marker osiagajaca wartosc 1 gdy spelnione zostana kryteria
    # lub wyczerpany zostanie licznik (500 iteracji [counter])
    while (marker = 0)

      # pobranie miejsca przejscia przez zero najblizszego zadanemu czasowi
      zero = Get nearest zero crossing... 'test'+ offset + skok

      # pobranie wartosci sygnalu przed oraz za przejsciem przez zero
      #(by okreslic czy przejscie pozytywne)
      select Sound 'object_name$'
      pomiar1 = Get value at time... 'zero'-'res' Nearest
      pomiar2 = Get value at time... 'zero'+'res' Nearest

      # pomiary energii sygnalu
      energia = Get power... 'zero'-0.0001 'zero'+0.0001
      energia2 = Get power... 'zero'-0.0015 'zero'
      energia3 = Get power... 'zero' 'zero'+0.001

      # jesli przejscie przez zero pozytywne i nie cofniete wiecej niz tolerancja
      if pomiar2 - pomiar1 >0 and zero > test - 'tolerancja'

        # w przypadku fonemow /p/ /t/ /k/ (koncowa granica fonemu)
        if label0$ = "p" or label0$ = "t" or label0$ = "k"

          # jesli energia w miejscu przeciecia zera lub za nim osiagnie zadana wartosc ustawienie
markera
           if energia>0.002 or energia3>0.006
             marker =1
           else

           # w przypadku kiedy nastepnym fonem jest ktoryms z wymienionych nizej
           # i energia spadla "dwukrotnie" ustawienie markera
           if (label1$ = "v" or label1$ = "f" or label1$ = "x" or label1$ = "w" or label1$ = "ts'" or
label1$ = "j" or label1$ = "l") and energia2 > energia*2
             marker = 1
           endif

           # jesli energia znacznie zmalala lub spadla ponizej zadanej wartosci ustawienie markera
           if energia2 > energia*11 or energia<7.00000000000000e-07
             marker =1

           else
           # granica nie spelnia kryteriow
```

```
            # zwiekszenie licznika i przesuniecia przy kolejnym wyszukiwaniu granicy (skok)
              skok = skok + 0.0005
              counter = counter +1
            endif
          endif

          # w przypadku gdy po aktualnym fonemie (dowolnym) nastepuje /p/ /t/ lub /k/
          elsif label1$ = "p" or label1$ = "t" or label1$ = "k"

            # jesli dodatkowo aktualnym fonem jest ktoryms z wymienionych nizej
            # i energia spadla "dwukrotnie" ustawienie markera
            if (label0$ = "s" or label0$ = "v" or label0$ = "f" or label0$ = "x" or label0$ = "w") and
energia2 > energia*2
              marker = 1

            #jesli aktualny fonem (poprzedzający /p/, /t/, /k/) to cisza ustawienie markera
            elsif label0$ = "_sil_"
              marker =1

            # jesli energia bardzo spadla lub jest nizsza niz zadana wartosc ustawienie markera
            elsif energia2>energia*15 or energia<0.0001
              marker = 1

            else
            # granica nie spelnia kryteriow
            # zwiekszenie licznika i przesuniecia przy kolejnym wyszukiwaniu granicy (skok)
              skok = skok + 0.0005
              counter = counter +1
            endif

          # w przypadku pozostalych fonemow ustawienie markera
          else
            marker = 1
          endif

        else
        # granica w negatywnym przejsciu przez zero
        # zwiekszenie licznika i przesuniecia przy kolejnym wyszukiwaniu granicy (skok)
          skok = skok + 0.0005
          counter = counter +1
        endif

        #licznik wyczerpany - ustawienie markera + komunikat (pozostaje dawna granica)
        if counter = 500
          marker = 1
          printline licznik wyczerpany plik 'object_name$' - fonem 'label0$' nr 'i' do poprawy
          errors = errors+1
          zero = 'test'
        endif

      endwhile

      # weryfikacja nowej granicy fonemu

      # gdy granica wypada przed lub na poprzedniej granicy
      if 'zero' <= 'test3' and i > 1

        # przesuniecie do przodu wzgledem dawnej pozycji i szukanie nowego przejscia przez zero
        zero2 = Get nearest zero crossing... test+0.001

        if 'zero2'> 'test3' and 'zero2' < 'test2'
```

```
          zero = zero2
          printline ciecie poprawione awaryjnie (lewe) 'object_name$' fonem 'label0$' nr 'i'
          warnings = warnings+1

        else
        # jesli wciaz ciecie wypada przed poprzednim pozostaje bez zmian + komunikat
          zero = 'test'
          printline 'object_name$' - do poprawek recznych (DC lewa) - fonem 'label0$' nr 'i' -
pozostawiony bez zmian
          errors = errors+1
        endif
      endif

      # gdy granica wypada za lub na nastepnej granicy
      if 'zero' >= 'test2'

        # przesuniecie do tylu wzgledem poczatkowej pozycji i szukanie nowego ciecia
        zero2 = Get nearest zero crossing... 'test'-0.001
          if 'zero2' < 'test3' && 'zero2' > 'test2'
          zero = zero2
          printline ciecie poprawione awaryjnie (prawe) 'object_name$' fonem 'label0$' nr 'i'
          warnings = warnings+1
        else
        # jesli wciaz ciecie wypada za nastepnym pozostaje bez zmian + komunikat
          zero = 'test'
          printline 'object_name$' - do poprawek recznych (DC prawa) - fonem 'label0$' nr 'i' -
pozostawiony bez zmian 'counter'
          errors = errors+1
        endif
      endif
      select TextGrid 'object_name$'

      # komunikat w przypadku wymienionych nizej fonemow jesli nowa granica w stosunku do
      # poprzedniej cofnela sie o wiecej niz 4ms lub przesuela do przodu o wiecej niz 30ms
      if label0$ = "p" or label0$ = "t" or label0$ = "k" or label0$ = "r" or label0$ = "l" or label0$ =
"w" or label0$ = "v"

        if 'test' > 'zero'+ 0.004
           przesuniecie = zero - test
           printline 'object_name$' - do sprawdzenia kluczowy fonem 'label0$' nr 'i' - duze
przesuniecie w lewo 'przesuniecie' 'counter'
           warnings = warnings+1
        endif

        if 'test' < 'zero'- 0.03
          przesuniecie = zero - test
          printline 'object_name$' - do sprawdzenia kluczowy fonem 'label0$' nr 'i' - duze
przesuniecie w prawo 'przesuniecie' 'counter'
          warnings = warnings+1
        endif

      # w przypadku pozostalch fonemow komunikat jesli nowa granica w stosunku do poprzedniej
      # cofnela sie o wiecej niz 10ms lub przesuela do przodu o wiecej niz 40ms
      else

        if 'test' > 'zero'+ 0.01 or 'test' < 'zero'- 0.04
           przesuniecie = zero - test
           printline 'object_name$' Warning: fonem 'label0$' nr 'i' - duze przesuniecie 'przesuniecie'
           warnings = warnings+1
        endif
      endif
```

```
      # komunikat jesli granica znalazla sie w negatywnym przejsciu przez zero
      if i<fonemow-1
        select Sound 'object_name$'
        pomiar1 = Get value at time... 'zero'-'res' Nearest
        pomiar2 = Get value at time... 'zero'+'res' Nearest
        if pomiar1>pomiar2
          printline 'object_name$' blad przy cieciu z minusa na plus fonem 'label0$' nr 'i'
          errors = errors+1
        endif
        select TextGrid 'object_name$'
      endif

      # usuniecie starej granicy
      Remove boundary at time... 1 'test'

      # wstawienie nowej granicy
      Insert boundary... 1 'zero'

      # wstawienie poprawnych nazw fonemow
      # w wyniku poprzednich operacji symbol fonemu 1 zostal doklejony do fonemu nr 2
      Set interval text... 1 i 'label0$'
      Set interval text... 1 i+1 'label1$'
    endfor

    Write to text file... 'maindir$''destdir$'\'object_name$'.TextGrid
    select all
    minus Strings list
    Remove
endfor

select all
Remove
printline all done --- warnings: 'warnings' ---- errors: 'errors'
```

Załącznik B. Spis rysunków i tabel

Rysunek 1.1: Dziedziny wiedzy związane z mową (w/g [1]). 7
Rysunek 1.2: Poziomy komunikacji werbalnej (częściowo w/g [19]). 8
Tabela 1.2.1: Transkrypcja fonetyczna samogłosek (w/g [1]). 13
Tabela 1.2.2: Transkrypcja fonetyczna spółgłosek trących (w/g [1]). 13
Tabela 1.2.3: Transkrypcja fonetyczna spółgłosek zwartych, czyli plozyjnych (w/g [1]). 14
Tabela 1.2.4: Transkrypcja spółgłosek zwanych sonorantami lub rezonantami (w/g [1]). 14
Tabela 1.2.5: Transkrypcja fonetyczna spółgłosek zwarto-trących (w/g [1]). 15
Rysunek 1.4.1: Wypowiedź „serce", z zaznaczonymi granicami fonemów (w/g [7]). 18
Rysunek 1.4.2: Przebieg czasowy, spektrogram i okno segmentacji w programie Praat. 19
Rysunek 1.4.3: Czworobok samogłosek (w/g [1]). 20
Tabela 1.4.4: Charakterystyka poszczególnych klas głosek (w/g [4]). 21
Rysunek 2.1.1: Sterowanie syntezatorem formantowym (w/g [1]). 25
Rysunek 2.1.2: Schemat formantowego syntezatora mowy Dennisa Klatta (w/g [1]). 26
Rysunek 2.2.1: Przykładowy model toru głosowego zbudowany (na podstawie przekrojów) w oparciu o odcinki rur cylindrycznych. 27
Rysunek 2.2.2: Uproszczone modelowanie ruchów artykulacyjnych (w/g [1]). 27
Rysunek 2.3.1: Schemat konkatenacyjnej syntezy mowy (w/g [1]). 28
Rysunek 2.5.1: Schemat syntezy mowy HTS (w/g [18]) 31
Rysunek 3.1: Etapy prowadzące do automatycznego rozpoznawania mowy (w/g [2]). 33
Rysunek 3.1.1: Położenie obszarów skupień samogłosek na płaszczyźnie 1 i 2 formantu (w/g [7]). ... 35
Rysunek 3.1.1.1: Przykład chwilowego widma sygnału mowy, z oznaczonym pierwszym i drugim formantem. 37
Rysunek 3.1.1.2: Przykładowy sygnał o niecałkowitej liczbie cykli i jego widmo, bez okienkowania i z zastosowaniem okna Hanninga (w/g [14]). 38
Rysunek 3.1.1.3: Widmo chwilowe stacjonarnej części samogłoski "i" - amplituda w skali logarytmicznej (w/g [7]). 38
Rysunek 3.1.1.4: Seria widm chwilowych w rzucie trójwymiarowym (Spektrogram 3D). 39
Rysunek 3.1.1.5: Seria widm chwilowych, widok 3D. Wypowiedź "serce" (w/g [7]). 39
Rysunek 3.1.1.6: Przykładowy spektrogram sygnału mowy razem z przebiegiem czasowym. 40
Rysunek 3.1.1.7: Filtry melowe (24) przedstawione w skali częstotliwościowej (w/g [2]). 40
Rysunek 3.1.1.8: Zestaw 25 trójkątnych filtrów w skali melowej (wysokości). 40
Rysunek 3.1.1.9: Etapy ekstrakcji parametrów spektralnych mowy (w/g [2]). 41
Rysunek 3.1.2.1: Schemat blokowy analizy cepstralnej (homomorficznej). 42
Rysunek 3.2.1.1: Prosty ukryty model Markowa (HMM), posiadający dwa stany i dwie możliwe obserwacje: X i Y. 44
Rysunek 3.2.1.2: Statystyczny model HMM głoski – fonemu (w/g [2]). 46
Rysunek 3.2.1.3: Przykład obrazujący działanie algorytmu Viterbiego (w/g [13]). 48
Rysunek 3.2.2.1: Przykładowy schemat sztucznego neuronu. 51
Rysunek 4.1.1: Częstotliwość występowania poszczególnych fonemów w bazie językowej (w/g [5]). 58
Rysunek 4.3.1: Okno programu Aligner. 61
Tabela 4.3.1.1: Porównanie poziomu rozpoznawalności różnych modeli HMM. 66
Rysunek 4.3.1.2: Porównanie segmentacji opartej na modelach (HMM) fonemów. 68
Rysunek 4.3.1.3: Porównanie modeli HMM fonemów inny przykład. 69
Rysunek 4.3.1.4: Porównanie modeli fonemów z modelami difonów. 70
Rysunek 4.3.1.5: Porównanie modeli fonemów z modelami difonów inny przykład. 70
Rysunek 4.4.1: Porównanie automatycznej segmentacji oraz wersji po korektach. 74
Tabela 4.4.1.1: Najczęstsze błędy automatycznej segmentacji. 78
Rysunek 4.4.1.2: Przykład korekty częstego błędu – samogłoska /e/ w połączeniu z trącą /s/. 79
Rysunek 4.4.1.3: Przykład ręcznych korekt. 79
Rysunek 4.4.1.4: Inny przykład ręcznych korekt. 79
Rysunek 4.4.2.1: Przykład braku ciągłości amplitudy (w/g [2]). 80

RYSUNEK 4.4.2.2: PRZYKŁAD ZAKŁÓCEŃ SIECI ELEKTRYCZNEJ (AUTOMATYCZNA SEGMENTACJA). 81
RYSUNEK 4.4.2.3: PRZYKŁAD PRZESUNIĘCIA GRANICY DO DODATNIEGO PRZEJŚCIA PRZEZ ZERO. 83
RYSUNEK 4.4.2.4: PRZYKŁAD KOREKTY WPROWADZONEJ PRZEZ SKRYPT. 83
RYSUNEK 4.4.4.1: FILTRACJA ZAKŁÓCEŃ SIECI ELEKTRYCZNEJ W PROGRAMIE AUDACITY. 84
RYSUNEK 4.4.5.1: OKNO TESTOWEGO SYNTEZATORA. 86
RYSUNEK 4.5.1: OKNO PROGRAMU VMWARE PLAYER, URUCHOMIONY SYSTEM DEBIAN ORAZ FESTIVAL. 90
TABELA 4.5.2: NOTACJA POLSKICH ZNAKÓW DIAKRYTYCZNYCH PRZYJĘTA W SYNTEZATORZE I KORPUSIE. 91
RYSUNEK 4.5.3: PRZYKŁAD RAPORTU ZWIĄZANEGO Z ZATRZYMANIEM SKRYPTU GENERUJĄCEGO PLIKI UTT. 93
RYSUNEK 4.5.4: PRZYKŁADOWE WPISY DODANE DO SŁOWNIKA TRANSKRYPCJI FONETYCZNEJ. 94
TABELA 4.6.1: PORÓWNANIE STATYSTYK KORPUSU TESTOWEGO ZALEŻNIE OD DŁUGOŚCI ZDAŃ. 96

Załącznik C. Zawartość płyty

- Niniejszy dokument
- Automatyczna segmentacja (pliki Textgrid)
- Segmentacja po korektach (poprawione pliki Textgrid)
- Ostateczna wersja segmentacji w formacie systemu Festival (pliki lab oraz wygenerowane na ich podstawie pliki utt)
- Korpus językowy bazy akustycznej
- Nagrania w postaci plików wav (16kHz)
- Skrypt korygujący (język skryptowy programu Praat)