AKADEMIA GÓRNICZO-HUTNICZA

IM. STANISŁAWA STASZICA W KRAKOWIE

Wydział Informatyki, Elektroniki i Telekomunikacji

Katedra Elektroniki

PRACA DYPLOMOWA

Inżynierska

Temat: Narzędzie programowe do optymalizacji zbioru modeli akustycznych języka polskiego

Software tool for the optimization of the statistical acoustic model set for polish language

Imię i nazwisko: Michał Dankiewicz

Kierunek studiów: Inżynieria Akustyczna

Opiekun pracy: dr inż. Jakub Gałka

Oświadczam, świadomy odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie i że nie korzystałem ze źródeł innych niż wymienione w pracy.

……

Składam serdeczne podziękowania
p. dr. inż. Jakubowi Gałce
za rozwianie wszelkich merytorycznych wątpliwości,
na które natknąłem się podczas pisania pracy.

Szczególne podziękowania składam Agnieszce
za wsparcie, którym darzyła mnie w licznych chwilach,
w których go potrzebowałem.

Streszczenie

Niniejszy dokument jest opisem działań wykonanych w ramach pracy dyplomowej o temacie *Narzędzie programowe do optymalizacji zbioru modeli akustycznych języka polskiego*. W pierwszej jego części znajduje się opis zagadnień teoretycznych związanych z komputerowym przetwarzaniem mowy. Opisane zostały niektóre pojęcia związane z tematem oraz główne algorytmy stosowane w systemach automatycznego rozpoznawania mowy do kodowania, modelowania i samego rozpoznawania mowy ludzkiej przez komputer. W drugiej części pracy przedstawione zostało stworzone narzędzie programowe. Poszukuje ono zbioru modeli akustycznych dającego większą skuteczność systemu rozpoznawania mowy niż wyjściowy zbiór 39 modeli (rozumianych jako fonemy języka polskiego) poprzez łączenie ze sobą niektórych modeli i traktowanie ich jako jeden. W pracy przedstawione są wyniki działania stworzonego narzędzia, to znaczy kombinacje modeli akustycznych dające poprawę skuteczności rozpoznania oraz dane liczbowe w postaci otrzymanych skuteczności. Następnie przedstawione zostały możliwości poszerzenia i ulepszenia funkcjonalności stworzonego narzędzia oraz wnioski z przeprowadzonej pracy.

Spis treści

Wykaz akronimów i oznaczeń

Lista akronimów

Acc	trafność rozpoznania, odsetek poprawnie rozpoznanych słów (ang. *Accuracy*)
ASR	automatyczne rozpoznawanie mowy (ang. *Automatic Speech Recognition*)
DCT	dyskretna transformacja kosinusowa (ang. *Discrete Cosine Transform*)
HMM	ukryty model Markowa (bądź liczba mnoga) (ang. *Hidden Markov Model*)
HTK	System HTK (Hidden Markov Models Toolkit)
MFCC	melowo-częstotliwościowe współczynniki cepstralne (ang. *Mel Frequency Cepstral Coefficients*)
SL	*Single Linkage*
WER	odsetek błędnie rozpoznanych słów (ang. *Word Error Rate*)

Lista wybranych oznaczeń

$\tilde{c}_n$	n-ty melowo-częstotliwościowy współczynnik cepstralny (MFCC)
$\tilde{S}_k$	energia sygnału wyjściowego z k-tego filtru banku filtrów melowych
λ	ukryty model Markowa
a_{ij}	prawdopodobieństwo przejścia ze stanu *i* do stanu *j* w modelu Markowa
$\boldsymbol{v}_k$	k-ty wektor obserwacji
$b_i(\boldsymbol{v}_k)$	prawdopodobieństwo emisji wektora obserwacji $\boldsymbol{v}_k$ przez stan *i*
$\boldsymbol{O}$	sekwencja obserwacji
$\boldsymbol{o}_t$	wektor obserwacji wyemitowany w chwili *t*
$\boldsymbol{q}$	sekwencja stanów
q_t	stan, w którym znajduje się system w chwili *t*
π_i	prawdopodobieństwo początkowe stanu *i*
A	macierz prawdopodobieństw przejść
B	macierz prawdopodobieństw emisji
π	macierz prawdopodobieństw początkowych
$\alpha_t(i)$	prawdopodobieństwo *forward*
$\beta_t(i)$	prawdopodobieństwo *backward*

1 Wstęp

Dynamiczny rozwój szeroko pojętej informatyki i elektroniki doprowadził do momentu, w którym przed komputerami codziennie stawiane są nowe wymagania. Interakcja człowieka z maszyną jest niezwykle ważna w nowoczesnych urządzeniach, ponieważ zależy od niej wygoda użytkowania, która bezpośrednio przekłada się na sprzedaż danego urządzenia. Nieodzownym elementem tej interakcji jest komunikacja z urządzeniem za pomocą głosu – służą do tego systemy automatycznego rozpoznawania mowy (ASR, ang. *Automatic Speech Recognition*). Wydawanie urządzeniu poleceń głosowych jest szczególnie istotne podczas prowadzenia samochodu bądź innych sytuacji, kiedy użycie rąk jest niemożliwe. Ten sposób komunikacji jest także – po prostu – wygodny. Systemy ASR mogą służyć również do automatycznej transkrypcji treści konferencji, wywiadów itp. Warunkiem sukcesu danego systemu na rynku jest jego niezawodność. Systemy ASR są wciąż udoskonalane przez wielu naukowców na całym świecie, tym bardziej, że są to narzędzia, których działanie w dużym stopniu zależy od języka. W niniejszej pracy autor przyczynia się do udoskonalenia automatycznego rozpoznawania mowy polskiej poprzez stworzenie narzędzia do optymalizacji zbioru modeli akustycznych – fonemów – języka polskiego, wykorzystywanych w systemach ASR.

1.1 Cele pracy

Celem niniejszej pracy jest stworzenie narzędzia programowego w środowisku MATLAB, które, przy pomocy narzędzia HTK do rozpoznawania mowy, będzie w stanie określić optymalną pod względem skuteczności rozpoznawania mowy kombinację fonemów, która to kombinacja zastąpi domyślny zbiór 39 fonemów dla mowy polskiej.

Koncepcja badania kombinacji fonemów oparta jest na badaniach przeprowadzonych przez Zespół Przetwarzania Sygnałów w Katedrze Elektroniki wydziału Informatyki, Elektroniki i Telekomunikacji AGH. Badania te wykazały, że zmniejszenie liczby modeli akustycznych do pewnego poziomu (ok. 16/17 fonemów) nie powoduje spadku rozróżnialności słów w słowniku zakodowanych na tej liczbie modeli. Przez słownik rozumie się tutaj listę słów i ich transkrypcji na poziomie fonemów. Zmniejszenie liczby fonemów w używanym alfabecie przekłada się na zmniejszenie liczby klas, które

w procesie rozpoznania musi rozróżnić system ASR, a im mniej klas do rozróżnienia, tym mniejsza szansa pomyłki systemu. W związku z powyższym zmniejszanie liczby modeli akustycznych jest zabiegiem opłacalnym i wartym zbadania.

1.2 Zawartość pracy

Rozdział 1. zawiera wstęp do pracy i określenie jej celów. W rozdziale 2. objaśnione są wybrane pojęcia oraz zagadnienia z zakresu przetwarzania sygnałów, mające zastosowanie w przetwarzaniu sygnału mowy. Omówione są w nim między innymi współczynniki MFCC, jawne i ukryte modele Markowa i algorytmy: forward-backward, Viterbiego oraz Bauma-Welcha. W rozdziale 3. umieszczony jest opis stworzonego narzędzia programowego – jego uruchomienia i działania. Rozdział 4. zawiera analizę wyników badania – otrzymanych sprawności dla różnych zbiorów fonemów. W rozdziale 5. omówione są plany dotyczące dalszego rozwijania stworzonego narzędzia.

2 Pojęcia związane z przetwarzaniem mowy

Człowiek postrzega mowę jako ciąg zdarzeń akustycznych o zróżnicowanej charakterystyce częstotliwościowej [7]. Aby komputer mógł nas „zrozumieć", nie wystarczy nagranie fali dźwiękowej zarejestrowane mikrofonem, czyli reprezentacja czasowa sygnału mowy. Sygnał ten musi zostać podzielony na „zdarzenia" (fonemy) i poddany szeregowi przekształceń, które pozwolą wydobyć (wyekstrahować) z niego zbiór cech. Tak otrzymany zbiór cech jest porównywany ze zbiorem otrzymanym podczas procesu treningu, czyli „nauki" systemu i na tej podstawie podejmowane są decyzje co do zawartości nagrania.

2.1 HTK

System HTK, czyli *Hidden Markov Model Toolkit*, jest narzędziem programowym służącym do budowania i operowania na ukrytych modelach Markowa. Jego najczęstszym wykorzystaniem jest tworzenie systemów automatycznego rozpoznawania mowy. W niniejszej pracy zostało ono wykorzystane do sprawdzania skuteczności rozpoznawania po zmodyfikowaniu zbioru modeli akustycznych. HTK podczas procesu treningu systemu przyjmuje na swoje wejście nagrania mowy oraz ich transkrypcje na poziomie fonemów, tzn. opis, jaki fonem jest wypowiadany w danej chwili w nagraniu. Na tej podstawie tworzone są modele poszczególnych fonemów w postaci ukrytych modeli Markowa. Proces rozpoznania polega na przetworzeniu rozpoznawanego nagrania (otrzymane dane nazywamy obserwacją) i znalezieniu takiej sekwencji fonemów, której pojawienie się w rozpoznawanym nagraniu maksymalizuje prawdopodobieństwo otrzymania obserwacji. Cały proces zostanie szerzej opisany w kolejnych rozdziałach pracy.

2.2 Fonemy

Fonem to najmniejsza jednostka mowy rozróżnialna dla użytkowników danego języka, która pozwala na rozróżnianie znaczeń wypowiedzi, choć sama znaczenia nie posiada. Z punktu widzenia automatycznego rozpoznawania mowy najistotniejszą właściwością fonemu jest to, że stanowi on zbiór cech dystynktywnych pozwalający odróżnić go od innych fonemów. Zakładając, że system komputerowy potrafi odczytać te cechy z nagrania dźwiękowego, może on przyporządkować je poszczególnym fonemom, a z tego przyporządkowania odtworzyć wypowiedź. Powoduje to, że fonem jest niezwykle

ważną jednostką systemu językowego z puntu widzenia automatycznego rozpoznawania mowy.

W rozpoznawaniu mowy polskiej stosuje się podział na 39 fonemów:

a	*d*	*h*	*ł*	*o*	*t*	*zi*	*sz*
ą	*e*	*i*	*m*	*p*	*u*	*rz*	*dz*
b	*ę*	*j*	*n*	*r*	*w*	*cz*	*sp*
c	*f*	*k*	*ni*	*s*	*y*	*drz*	*sil*
ci	*g*	*l*	*N*	*si*	*z*	*dzi*	

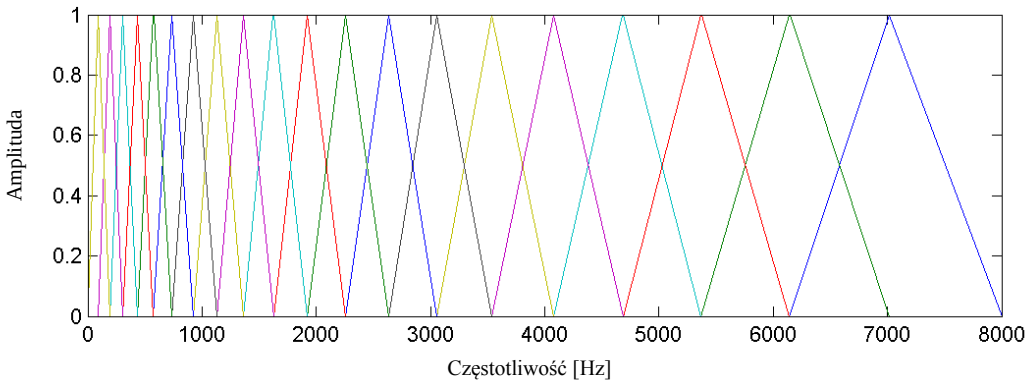
Jak widać, większość z nich bezpośrednio odpowiada literom alfabetu polskiego oraz polskim głoskom. Odstępstwami są fonemy *sp* i *sil*, które w systemach ASR oznaczają odpowiednio *krótką pauzę* i *ciszę*. Fonem *N* odpowiada dźwiękowi pojawiającemu się w mowie, kiedy głoska *n* poprzedza głoskę *k* lub *g*, na przykład w słowie *piosenka* lub *panga*.

2.3 Ekstrakcja współczynników MFCC

Powszechnie stosowaną metodą ekstrakcji cech sygnału mowy jest metoda MFCC (ang. *Mel Frequency Cepstral Coefficients*), czyli melowo-częstotliwościowych współczynników cepstralnych. Melowa skala częstotliwości wykorzystywana przez tę metodę to odpowiednio zmodyfikowana skala częstotliwości, która symuluje subiektywne, ludzkie odczuwanie wysokości dźwięku – niższe dźwięki są przez nas słyszane z większą rozdzielczością częstotliwościową niż dźwięki wyższe. Konwersja z częstotliwości w Hertzach f na częstotliwość melową m odbywa się najczęściej [2] według formuły:

$$m = 1127 \cdot \ln\left(1 + \frac{f}{700}\right) \tag{2.1}$$

Skala melowa jest realizowana przez bank filtrów. Filtry te posiadają trójkątne charakterystyki częstotliwościowe, a ich odległość i szerokość są ustalone przez stały interwał w skali melowej. Na Rys. 1 przedstawiony jest przykładowy bank filtrów.



Rys 1. Przykładowy bank filtrów skali melowej. Rysunek ze strony http://neural.cs.nthu.edu.tw

W pierwszym etapie ekstrakcji współczynników MFCC sygnał mowy (lub częściej jego fragment, tzw. ramka, która najczęściej ma długość 20 ms) jest poddawany filtracji bankiem filtrów. Niech $\tilde{S}_k$ oznacza energię sygnału wyjściowego z *k*-tego filtru (*k*=1,2,…,*K*, gdzie *K* – liczba filtrów). Aby otrzymać współczynniki MFCC (czyli tzw. cepstrum sygnału) należy zastosować dyskretną transformację kosinusową (DCT, ang. *Discrete Cosine Transform)* na zlogarytmowanych współczynnikach energii według następującego wzoru:

$$\tilde{c}_n = \sum_{k=1}^{K} \left(\log(\tilde{S}_k) cos\left[n\left(k - \frac{1}{2}\right)\frac{\pi}{K} \right] \right), \quad n = 1,2,\dots,L \tag{2.2}$$

gdzie *L* jest pożądaną długością cepstrum. *L* może być mniejsze od *K* (liczba współczynników cepstrum może być niższa od liczby filtrów w banku filtrów), co jest bardzo pożądaną właściwością dyskretnej transformacji kosinusowej – dzięki niej cepstrum można zapisać na mniejszej liczbie bitów, a przez to ograniczyć zajmowaną pamięć komputera. Zbiór $\tilde{c}_n$ nazywa się wektorem MFCC.

Narzędzie HTK posiada wbudowany moduł odpowiedzialny za ekstrakcję współczynników MFCC.

2.4 Modele Markowa

2.4.1 Procesy Markowa

Aby wytłumaczyć ideę modeli Markowa, rozważmy pewien system, który może być opisany w dowolnej chwili czasu jako znajdujący się w jednym z *N* odrębnych stanów oznaczonych {1,2,…,*N*}, jak na Rys. 1. Co pewien okres czasu (w każdej kolejnej, dyskretnej chwili) system zmienia swój stan (zmiana z danego stanu na ten sam stan jest dopuszczalna). Każda zmiana stanu może nastąpić z pewnym konkretnym prawdopodobieństwem, które zależy od stanu, w którym aktualnie znajduje się system, a nie zależy od stanów, w jakich znajdował się system w poprzednich chwilach. Kolejne chwile czasu powiązane ze zmianą stanu oznaczmy $t = 1,2,\ldots$, a stan, w którym znajduje się system w chwili t oznaczmy jako q_t. Probabilistyczny opis przejścia systemu pomiędzy stanami [5] bierze pod uwagę stan, z którego system przechodzi i stan, do którego przechodzi:

$$a_{ij} = P(q_t = j | q_{t-1} = i), \;\; 1 \leq i, j \leq N \tag{2.3}$$

gdzie:

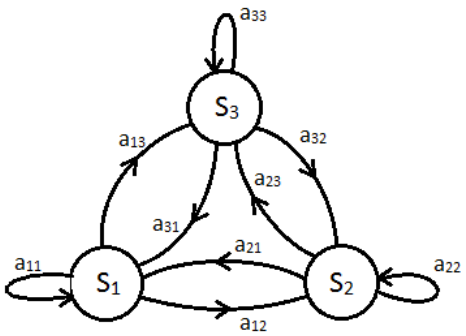
α_{ij} – prawdopodobieństwo przejścia ze stanu i do stanu j,

przy czym:

$$\alpha_{ij} \geq 0 \quad \forall i, j, \tag{2.4}$$

$$\sum_{j=1}^{N} \alpha_{ij} = 1 \quad \forall i, \tag{2.5}$$

ponieważ spełnione są podstawowe ograniczenia procesów stochastycznych.



Rys. 1 Prosty schemat modelu Markowa z 3 stanami i wszystkimi możliwymi przejściami.

Każdy jawny model Markowa (a taki do tej pory był rozważany) można opisać za pomocą 2 macierzy:

- macierzy przejść A
- macierzy prawdopodobieństw początkowych π

Macierz przejść, w ogólności, przy liczbie stanów równej N, jest następującej postaci:

$$A = \{a_{ij}\} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & & & \\ \vdots & & \ddots & \vdots \\ a_{N1} & & \dots & a_{NN} \end{bmatrix}$$

natomiast macierz prawdopodobieństw początkowych przyjmuje postać:

$$\pi = \{\pi_i\} = \begin{bmatrix} \pi_1 \\ \pi_2 \\ \vdots \\ \pi_N \end{bmatrix}$$

gdzie:

π_i – prawdopodobieństwo tego, że system w pierwszej chwili czasu będzie w stanie i.

2.4.2 Ukryte modele Markowa

Do tej pory rozważaliśmy model, w którym każdy stan odpowiadał obserwowalnemu zdarzeniu. Oznacza to, że w danej chwili można być pewnym, że system znajduje się w danym stanie. Istnieją jednak przypadki, kiedy stan systemu nie może być określony bezpośrednio, a jedynie na podstawie pewnych obserwacji, które emituje. Taki system może być zamodelowany za pomocą ukrytego (niejawnego) modelu Markowa (HMM).

HMM pozwalają oszacować prawdopodobieństwo znajdowania się systemu w danym stanie na podstawie obserwacji, która jest probabilistyczną funkcją stanu, więc nie określa stanu jednoznacznie. Słowo „ukryty" w nazwie modelu odnosi się do procesu stochastycznego zmiany stanów, który nie jest bezpośrednio obserwowalny.

Znając tylko obserwację możemy wysunąć przypuszczenie co do tego, który stan ją wyemitował. Chcemy więc obliczyć

$$P(q_t|\boldsymbol{o}_t), \tag{2.6}$$

gdzie:

$\boldsymbol{o}_t$ – wektor obserwacji wyemitowany w chwili *t*.

Wyrażenie (2.6) możemy zapisać przy pomocy reguły Bayesa jako:

$$P(q_t|\boldsymbol{o}_t) = \frac{P(\boldsymbol{o}_t|q_t)P(q_t)}{P(\boldsymbol{o}_t)}. \tag{2.7}$$

Czynnik $P(q_t)$ może zostać obliczony rekurencyjnie według wzorów:

$$P(q_t) = P(q_{t-1})a_{q_t\, q_{t-1}} \tag{2.8}$$

$$P(q_1) = \pi_{q_1} \tag{2.9}$$

Aby obliczyć czynniki $P(\boldsymbol{o}_t|q_t)$ i $P(\boldsymbol{o}_t)$ musimy znać prawdopodobieństwa emisji danych wektorów obserwacji przez dany stan. Jawny model Markowa określany jest przez 2 macierze: przejść i stanów początkowych. Do opisu ukrytego modelu Markowa potrzebna jest jeszcze jedna macierz – macierz prawdopodobieństw emisji obserwacji. Przybiera ona postać:

$$B = \{b_i(\boldsymbol{v}_k)\} = \begin{bmatrix} b_1(\boldsymbol{v}_1) & b_1(\boldsymbol{v}_2) & \cdots & b_1(\boldsymbol{v}_M) \\ b_2(\boldsymbol{v}_1) & & & \\ \vdots & & \ddots & \vdots \\ b_N(\boldsymbol{v}_1) & & \cdots & b_N(\boldsymbol{v}_M) \end{bmatrix}$$

gdzie:

$\boldsymbol{v}_k$ – *k*-ty wektor obserwacji,

$b_i(\boldsymbol{v}_k)$ – prawdopodobieństwo wyemitowania przez stan *i* wektora obserwacji $\boldsymbol{v}_k$.

Znając prawdopodobieństwa emisji obserwacji możemy obliczyć pozostałe czynniki równania (2.7) za pomocą wzorów:

$$P(\boldsymbol{o}_t|q_t) = b_{q_t}(\boldsymbol{o}_t), \tag{2.10}$$

$$P(\boldsymbol{o}_t) = \sum_{i=1}^{N} P(\boldsymbol{o}_t|i)P(i), \tag{2.11}$$

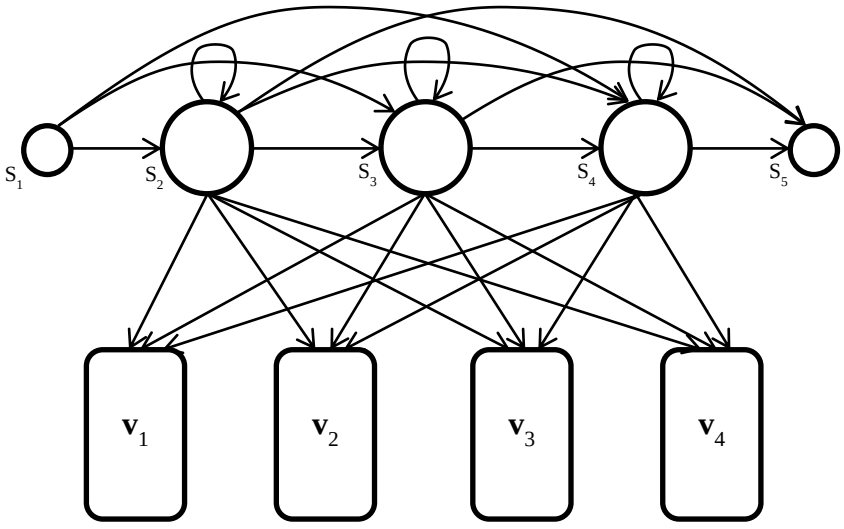
gdzie:

N – liczba stanów modelu.

Cały model Markowa oznaczany jest symbolem λ. Dla ukrytego modelu Markowa:

$$\lambda = \{A, B, \pi\}.$$

Nie wszystkie stany modelu muszą emitować obserwacje. Jeśli dany stan ma zerowe prawdopodobieństwo emisji wszystkich wektorów obserwacji, nazywamy go stanem nieemitującym. Stan, w którym zachodzi emisja wektorów obserwacji, nazywamy stanem emitującym. Schemat ukrytego modelu Markowa może przestawiać się jak na Rys. 3.:



Rys. 2 Przykładowy schemat ukrytego modelu Markowa typu *left-to-right* ze stanami emitującymi i nieemitującymi

W powyższym modelu S_1 i S_5 są stanami nieemitującymi, S_2, S_3 i S_4 są stanami emitującymi, a $\mathbf{v}_1$-$\mathbf{v}_4$ są wektorami obserwacji. Określenie *left-to-right* oznacza, że nie jest

możliwe przejście ze stanu S_j do S_i, jeśli $j>i$. Na rysunku nie naniesiono prawdopodobieństw przejść oraz emisji w celu zwiększenia czytelności.

HMM pozwalają również określić najbardziej prawdopodobną sekwencję stanów na podstawie sekwencji obserwacji, co jest wykorzystywane bezpośrednio w procesie rozpoznawania mowy – własność ta będzie omówiona w sekcji 2.6.

2.4.3 Ukryte modele Markowa w HTK

System HTK wykorzystuje HMM do modelowania fonemów. Każdemu fonemowi, który wyszczególniony jest w danych wejściowych systemu, odpowiada jeden HMM. Liczba stanów modelu może być dowolnie ustalona - w niniejszej pracy używane są modele z 5 stanami, w tym 3 emitującymi (pierwszy i ostatni stan to stany nieemitujące). W procesie **kodowania**, który poprzedza proces treningu, każde nagranie dzielone jest na mniejsze fragmenty, z których ekstrahowane są później współczynniki MFCC, które stanowią obserwację. W ten sposób otrzymuje się zbiór wektorów MFCC, które w procesie **treningu** systemu przyporządkowywane są do poszczególnych stanów HMM.

2.5 Algorytm forward-backward

Dzięki algorytmowi forward-backward możemy odpowiedzieć na pytanie: jakie jest prawdopodobieństwo otrzymania danej sekwencji obserwacji $\boldsymbol{O} = (\boldsymbol{o}_1, \boldsymbol{o}_2, \dots, \boldsymbol{o}_T)$ pod warunkiem danego modelu $\lambda = \{A, B, \pi\}$, to znaczy, możemy obliczyć $P(\boldsymbol{O}|\lambda)$. Bezpośrednie obliczenie tej wartości odbywałoby się za pomocą sumowania po wszystkich możliwych sekwencjach stanów $\boldsymbol{q}$ prawdopodobieństwa wygenerowania obserwacji pod warunkiem tej sekwencji stanów i danego modelu pomnożonego przez prawdopodobieństwo tej sekwencji stanów pod warunkiem danego modelu, to jest:

$$P(\boldsymbol{O}|\lambda) = \sum_{wszystkie\ \boldsymbol{q}} P(\boldsymbol{O}|\boldsymbol{q}, \lambda) P(\boldsymbol{q}|\lambda). \tag{2.12}$$

Złożoność obliczeniowa tego równania jest jednak bardzo duża ($2T \cdot N^T$, gdzie T – długość sekwencji obserwacji, N-liczba stanów emitujących modelu), dlatego obliczanie $P(\boldsymbol{O}|\lambda)$ musi odbywać się w inny, bardziej efektywny sposób – służy do tego procedura *forward*.

2.5.1 Procedura forward

Rozważmy zmienną *forward* $\alpha_t(i)$ zdefiniowaną jako

$$\alpha_t(i) = P(\boldsymbol{o}_1 \boldsymbol{o}_2 \dots \boldsymbol{o}_t, q_t = i|\lambda), \tag{2.13}$$

czyli prawdopodobieństwo częściowej sekwencji obserwacji, $\boldsymbol{o}_1 \boldsymbol{o}_2 \dots \boldsymbol{o}_t$, (do chwili *t*) i bycia w stanie *i* w chwili *t*, pod warunkiem modelu λ. Wartość tę możemy wyznaczyć iteracyjnie według następujących kroków:

1. Inicjalizacja

$$\alpha_1(i) = \pi_i b_i(\boldsymbol{o}_1), \quad 1 \le i \le N \tag{2.14}$$

2. Indukcja

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(\boldsymbol{o}_{t+1}), \quad \begin{matrix} 1 \le t \le T-1 \\ 1 \le j \le N \end{matrix} \tag{2.15}$$

3. Zakończenie

$$P(\boldsymbol{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{2.16}$$

W kroku pierwszym obliczamy prawdopodobieństwo tego, że system znajduje się w pierwszej chwili czasu w stanie *i* i emituje obserwację $\boldsymbol{o}_1$. Otrzymujemy zmienną *forward* dla każdego stanu w modelu dla pierwszej chwili czasu. W drugim kroku rozważamy stan *j* w chwili *t*+1. System może znaleźć się w tym stanie po przejściu z każdego z *N* stanów, w których mógł znajdować się w chwili *t* – stąd sumowanie iloczynów po stanach *i*. Z tego sumowania otrzymujemy prawdopodobieństwo bycia systemu w stanie *j* w chwili *t*+1 i otrzymania sekwencji wcześniejszych obserwacji, tj. $\boldsymbol{o}_1 \dots \boldsymbol{o}_t$. Aby „dołożyć" do tej sekwencji obserwację $\boldsymbol{o}_{t+1}$ należy otrzymaną wartość pomnożyć przez prawdopodobieństwo emisji tej obserwacji przez stan *j*, tj. $b_j(\boldsymbol{o}_{t+1})$. Procedura ta powtarzana jest dla każdego stanu modelu i każdej chwili czasu. Kiedy obliczone zostaną wszystkie zmienne *forward*, należy zsumować ich wartości dla każdego stanu i chwili *T* – otrzymujemy w ten sposób pożądane prawdopodobieństwo $P(\boldsymbol{O}|\lambda)$,

ponieważ sekwencja obserwacji $\boldsymbol{O}$ mogła zostać wygenerowana przez sekwencję stanów, na której końcu mógł znaleźć się którykolwiek z *N* stanów.

2.5.2 Procedura backward

To samo prawdopodobieństwo możemy znaleźć stosując nieco inne podejście. Zdefiniujmy zmienną *backward* $\beta_t(i)$ oznaczającą prawdopodobieństwo częściowej sekwencji obserwacji od chwili *t*+1 do końca pod warunkiem bycia w chwili *t* w stanie *i* oraz modelu λ:

$$\beta_t(i) = P(\boldsymbol{o}_{t+1}\, \boldsymbol{o}_{t+2} \dots \boldsymbol{o}_T | q_t = i, \lambda). \tag{2.17}$$

Podobnie jak w przypadku zmiennej *forward*, możemy znaleźć $\beta_t(i)$ iteracyjnie:

1. Inicjalizacja

$$\beta_T(i) = 1, \quad 1 \le i \le N \tag{2.18}$$

2. Indukcja

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(\boldsymbol{o}_{t+1}) \beta_{t+1}(j), \quad \begin{matrix} t = T-1, T-2, \dots, 1 \\ 1 \le i \le N \end{matrix} \tag{2.19}$$

W pierwszym kroku $\beta_T(i)$ jest dla każdego stanu i chwili *T* ustalana jako 1. Następnie, aby obliczyć $\beta_T(i)$ dla danego stanu w którejś z poprzednich chwil *t*, należy wziąć pod uwagę wszystkie stany *j*, do których może przejść system w chwili *t*+1 (a_{ij}), prawdopodobieństwo wyemitowania przez nie następnej z sekwencji obserwacji ($b_j(\boldsymbol{o}_{t+1})$) oraz emisję reszty sekwencji obserwacji z tego stanu *j* ($\beta_{t+1}(j)$).

W procesie rozpoznawania mowy algorytm forward-backward odgrywa istotną rolę. Po pierwsze – wykorzystywany jest w procesie treningu systemu przez algorytm Bauma-Welcha (patrz: sekcja 2.7). Po drugie – na procedurze *forward* oparty jest algorytm Viterbiego (patrz: sekcja 2.6) służący, po pewnych modyfikacjach, do rozpoznawania mowy ciągłej w HTK.

2.6 Algorytm Viterbiego

Algorytm Viterbiego daje nam odpowiedź na pytanie: jaka sekwencja stanów danego modelu z największym prawdopodobieństwem mogła wyemitować daną sekwencję obserwacji?

Aby znaleźć najlepszą (optymalną) sekwencję stanów $\boldsymbol{q}^* = (q_1\, q_2 \ldots q_T)$ dla danej sekwencji obserwacji $\boldsymbol{O} = (\boldsymbol{o}_1\, \boldsymbol{o}_2 \ldots \boldsymbol{o}_T)$, zdefiniujmy wielkość

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P[q_1\, q_2 \ldots q_{t-1},\ \ q_t = i,\ \ \boldsymbol{o}_1\, \boldsymbol{o}_2 \ldots \boldsymbol{o}_t\, |\lambda]. \tag{2.20}$$

Oznacza to, że $\delta_t(i)$ jest najbardziej prawdopodobną spośród wszystkich ścieżek (sekwencji stanów) w chwili *t*, która odpowiada pierwszym *t* obserwacjom i kończy się w stanie *i*. Przez indukcję:

$$\delta_{t+1}(j) = \left[\max_i \delta_t(i)\, a_{ij}\right] b_j(\boldsymbol{o}_{t+1}). \tag{2.21}$$

Aby uzyskać całą sekwencję stanów, musimy zapamiętywać argument *i*, który maksymalizował powyższe równanie dla każdego *j* i *t*, czyli to, „z którego stanu najlepiej przyjść do stanu *j* w chwili *t*". Robimy to za pomocą macierzy $\psi_t(j)$. Całość procesu znajdowania najlepszej sekwencji przedstawia następujący algorytm:

1. Inicjalizacja

$$\delta_1(i) = \pi_i b_i(\boldsymbol{o}_1), \quad 1 \le i \le N \tag{2.22}$$

$$\psi_1(i) = 0 \tag{2.23}$$

2. Rekurencja

$$\delta_t(j) = \max_{1 \le i \le N} \left[\delta_{t-1}(i) a_{ij}\right] b_j(\boldsymbol{o}_t), \quad \begin{matrix} 2 \le t \le T \\ 1 \le j \le N \end{matrix} \tag{2.24}$$

$$\psi_t(j) = \arg\max_{1 \le i \le N} \left[\delta_{t-1}(i) a_{ij}\right], \quad \begin{matrix} 2 \le t \le T \\ 1 \le j \le N \end{matrix} \tag{2.25}$$

3. Zakończenie

$$q_T^* = \arg\max_{1 \le i \le N} \delta_T(i) \tag{2.26}$$

4. Wyznaczanie wstecz ścieżki

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \dots, 1, \tag{2.27}$$

gdzie q_t^* oznacza stan, w którym znajduje się system w chwili t w przypadku optymalnej sekwencji stanów.

Przykładowo, dla HMM danego macierzami

$$A = \begin{bmatrix} 0{,}5 & 0{,}4 & 0{,}1 \\ 0{,}5 & 0{,}2 & 0{,}3 \\ 0{,}3 & 0{,}1 & 0{,}6 \end{bmatrix}, \quad B = \begin{bmatrix} 0{,}2 & 0{,}8 \\ 0{,}7 & 0{,}3 \\ 0{,}4 & 0{,}6 \end{bmatrix}, \quad \pi = [0 \quad 1 \quad 0]$$

macierze δ i ψ są następujące (kolejne kolumny oznaczają chwile czasu, a wiersze - stany):

$$\delta = \begin{bmatrix} 0 & 0{,}12 & 0{,}012 & 0{,}01344 \\ 0{,}3 & 0{,}018 & 0{,}0336 & 0{,}002016 \\ 0 & 0{,}054 & 0{,}01296 & 0{,}006048 \end{bmatrix}, \quad \psi = \begin{bmatrix} 0 & 2 & 1 & 2 \\ 0 & 2 & 1 & 2 \\ 0 & 2 & 3 & 2 \end{bmatrix} \quad q_T^* = 1$$

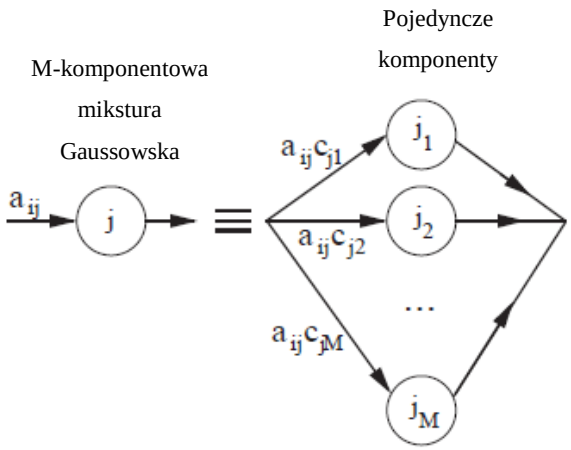
Wyznaczanie wstecz optymalnej sekwencji stanów oznaczone jest strzałkami. Stan q_T^* to stan 1, ponieważ w ostatniej kolumnie macierzy δ największa wartość pojawia się dla stanu 1. Kolejne kroki wyznaczania ścieżki opisane są już bezpośrednio w macierzy ψ. Stąd

$$\boldsymbol{q}^* = (S_2\ S_1\ S_2\ S_1).$$

Algorytm Viterbiego w systemie HTK jest zmodyfikowany w taki sposób, aby radzić sobie nie tylko z wyszukaniem optymalnej sekwencji stanów w obrębie jednego HMM, ale także w obrębie wielu połączonych HMM w celu rozpoznawania mowy ciągłej. HTK łączy modele Markowa przy pomocy nieemitujących stanów początkowych i końcowych – końcowy stan jednego modelu staje się początkowym następnego. Dzięki temu możliwe jest połączenie wielu modeli różnych fonemów na przykład w celu stworzenia jednego modelu danego słowa.

2.7 Algorytm Bauma-Welcha

Algorytm Bauma-Welcha to metoda służąca estymacji parametrów HMM (macierzy *A*, *B* i π) na podstawie danej sekwencji obserwacji. Zanim zostanie przedstawiona jego istota [5] należy zauważyć, że w HTK macierz *B* nie zawiera pojedynczych prawdopodobieństw. Prawdopodobieństwo emisji obserwacji modelowane jest przez kombinacje liniowe M rozkładów Gaussa (mikstury gaussowskie, GMM, ang. *Gaussian Mixture Models*) na *N*-wymiarowej przestrzeni, gdzie *N* - liczba współczynników MFCC ekstrahowanych w procesie kodowania. HTK, obliczając parametry poszczególnych komponentów mikstury, traktuje je tak, jakby były osobnymi stanami modelu. Prawdopodobieństwa przejść do tych stanów reprezentują wagi poszczególnych komponentów [4]. Ilustruje to Rys. 3:



Rys. 3 Reprezentacja mikstury gaussowskiej (rys.:[3])

Stąd

$$P(\boldsymbol{v}_k|j) = \sum_{m=1}^{M} c_{jm} p_{jm}(\boldsymbol{v}_k), \tag{2.28}$$

gdzie:

M – liczba komponentów mikstury,

c_{jm} – waga *m*-tego komponentu mikstury stanu *j*,

$p_{jm}(\boldsymbol{v}_k)$ – prawdopodobieństwo emisji wektora obserwacji $\boldsymbol{v}_k$ dane *m*-tym komponentem mikstury.

Zadaniem algorytmu Bauma-Welcha jest znalezienie takiego modelu $\bar{\lambda}$, aby wiarygodność $P(\boldsymbol{O}|\bar{\lambda})$ była większa niż $P(\boldsymbol{O}|\lambda)$, gdzie λ – znany model, $\boldsymbol{O}$ – sekwencja

obserwacji. Odbywa się to za pomocą iteracyjnej procedury zwanej reestymacją. Aby ją opisać, zdefiniujmy najpierw funkcję $\xi_t(i,j)$ oznaczającą prawdopodobieństwo bycia w stanie i w chwili t i bycia w stanie j w chwili t+1 pod warunkiem danego modelu i sekwencji obserwacji, tzn.:

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j|\boldsymbol{O}, \lambda). \tag{2.29}$$

Korzystając z definicji prawdopodobieństw forward i backward możemy zapisać funkcję $\xi_t(i,j)$ jako:

$$\begin{aligned}\xi_t(i,j) &= \frac{P(q_t = i, q_{t+1} = j, \boldsymbol{O}|\lambda)}{P(\boldsymbol{O}|\lambda)} \\ &= \frac{\alpha_t(i)a_{ij}b_j(\boldsymbol{o}_{t+1})\beta_{t+1}(j)}{P(\boldsymbol{O}|\lambda)} \\ &= \frac{\alpha_t(i)a_{ij}b_j(\boldsymbol{o}_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(\boldsymbol{o}_{t+1})\beta_{t+1}(j)}.\end{aligned} \tag{2.30}$$

Funkcję $\gamma_t(i)$ zdefiniujmy jako prawdopodobieństwo bycia w stanie i w chwili t pod warunkiem danego modelu i sekwencji obserwacji, to jest:

$$\gamma_t(i) = P(q_t = i|\boldsymbol{O}, \lambda). \tag{2.31}$$

Można zatem odnieść $\gamma_t(i)$ do $\xi_t(i,j)$ poprzez sumowanie po j:

$$\gamma_t(i) = \sum_{j=1}^{N}\xi_t(i,j). \tag{2.32}$$

Jeśli $\gamma_t(i)$ zsumujemy po t, otrzymamy liczbę, która może być interpretowana jako przewidywana liczba odwiedzin stanu i (bycia systemu w stanie i) lub, równoważnie, liczba przejść ze stanu i (o ile wykluczymy z sumowania chwilę t=T). Podobnie, zsumowanie $\xi_t(i,j)$ po t da nam estymatę liczby przejść ze stanu i do j. Pamiętając, że $\xi_t(i,j)$ zależy od sekwencji obserwacji $\boldsymbol{O}$, powyższe zależność można zapisać jako:

$\sum_{t=1}^{T-1} \gamma_t(i)$ = przewidywana liczba przejść ze stanu i w $\boldsymbol{O}$ (2.33)

$\sum_{t=1}^{T-1} \xi_t(i,j)$ = przewidywana liczba przejść ze stanu i do stanu j w $\boldsymbol{O}$ (2.34)

Używając powyższych wzorów można podać metodę reestymacji parametrów HMM. Dla macierzy π, A, B otrzymujemy:

$$\bar{\pi}_i = \text{oczekiwana liczba razy bycia systemu w stanie } i \text{ w czasie } (t=1) = \gamma_1(i) \tag{2.35}$$

$$\bar{a}_{ij} = \frac{\text{oczekiwana liczba przejść ze stanu } i \text{ do } j}{\text{oczekiwana liczba przejść ze stanu } i} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{2.36}$$

$$\bar{b}_i(k) = \frac{\text{oczekiwana liczba razy w stanie } i \text{ i obserwowania wektora } \mathbf{v}_k}{\text{oczekiwana liczba razy w stanie } i} = \frac{\sum_{t=1,\ \boldsymbol{o}_t=\boldsymbol{v}_k}^{T} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)} \tag{2.37}$$

Jeśli do obliczenia prawych stron równań (2.35)-(2.37) użyjemy znanego modelu $\lambda = \{A, B, \pi\}$, a lewych stron tych równań użyjemy do stworzenia modelu $\bar{\lambda} = \{\bar{A}, \bar{B}, \bar{\pi}\}$, to:
(1) albo otrzymany model $\bar{\lambda}$ będzie bardziej wiarygodny, tzn. $P(\boldsymbol{O}|\bar{\lambda}) > P(\boldsymbol{O}|\lambda)$,
(2) albo model wyjściowy λ jest punktem krytycznym funkcji wiarygodności i $\bar{\lambda} = \lambda$.

Według powyższej procedury iteracyjnie używamy $\bar{\lambda}$ zamiast λ i powtarzamy obliczenia reestymacji. Można w ten sposób zwiększać prawdopodobieństwo wyemitowania sekwencji $\boldsymbol{O}$ przez model dopóty, dopóki nie osiągnie się pewnego punktu końcowego. Ostatecznym wynikiem algorytmu jest najlepsza w sensie ML (ang. *maximum likelihood*, maksymalna wiarygodność) estymata HMM.

Należy zauważyć, że algorytm forward-backward używany w algorytmie Bauma-Welcha nie jest odporny na maksima lokalne, podczas gdy funkcja wiarygodności posiada ich wiele.

3 Stworzone narzędzie programowe

W celu optymalizacji zbioru modeli akustycznych języka polskiego stworzone zostało narzędzie programowe (dalej: Narzędzie). Jest to skrypt środowiska MATLAB automatyzujący procesy treningu i rozpoznania przeprowadzane przez system HTK oraz dokonujący zmian w transkrypcji plików dźwiękowych i zbiorze fonemów. Szczegóły narzędzia oraz bazy danych użytej w tworzeniu Narzędzia zostaną opisane w następnych sekcjach.

3.1 Koncepcja sposobu szukania optymalnych zbiorów

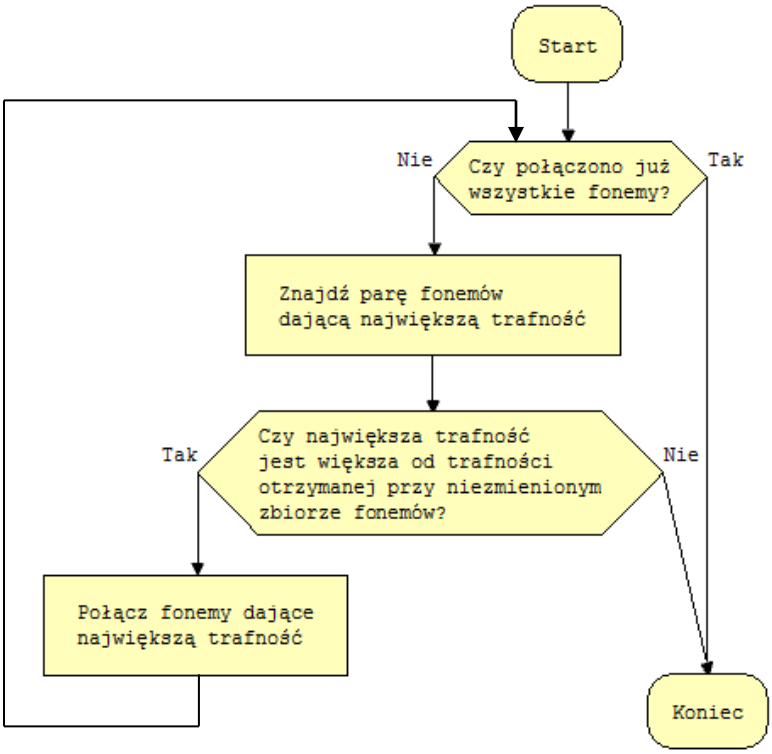
Za zbiór modeli akustycznych języka uznaje się zbiór ukrytych modeli Markowa używanych przez HTK do modelowania fonemów. Optymalizacja tego zbioru polega na zmniejszeniu liczby jego elementów poprzez traktowanie dwóch lub więcej fonemów, jako jednego (dalej: połączenie fonemów). Zmniejszenie liczby elementów zbioru pozwoli przyspieszyć proces rozpoznawania i wpłynie na jego skuteczność. Zadaniem stworzonego Narzędzia jest znalezienie takich połączeń fonemów, dla których skuteczność systemu ASR będzie wyższa niż w przypadku traktowania każdego fonemu osobno.

Aby otrzymać swego rodzaju referencyjną skuteczność systemu ASR, to znaczy skuteczność w przypadku nie wprowadzania żadnych zmian do zbioru fonemów, należy skorzystać ze skryptu `test_bez_zmian.m`, a następnie wprowadzić otrzymaną skuteczność jako zmienną w odpowiednim skrypcie wywołującym sprawdzanie różnych kombinacji fonemów (vide sekcja 3.4.4). Przed uruchomieniem skryptu `test_bez_zmian.m` należy jednak wykonać czynności przygotowawcze opisane w sekcji 3.4.

3.1.1 Algorytm single linkage

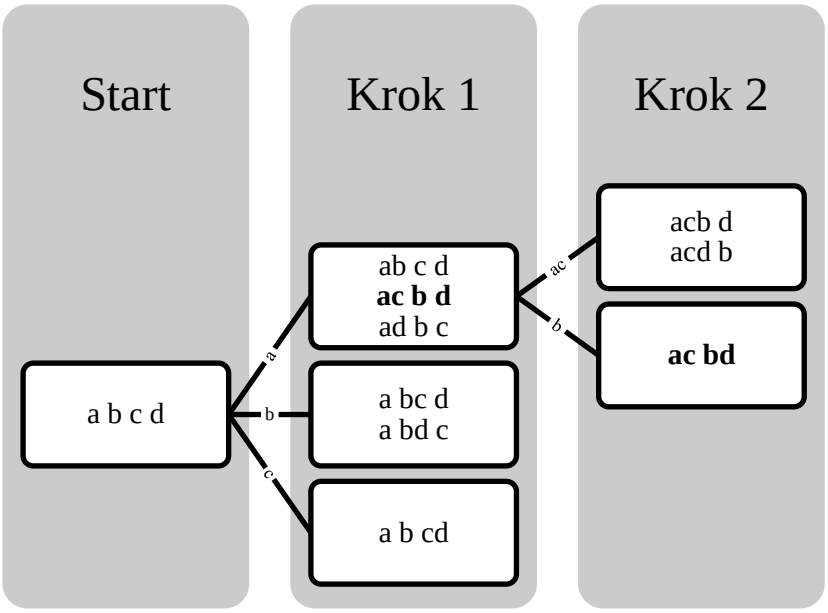
Single linkage (SL) to metoda klasteryzacji zbioru danych. Jest to metoda iteracyjna polegająca na wykonywaniu jednego łączenia dwóch elementów zbioru w jednym kroku. Za każdym razem wybierane są te dwa elementy, których odległość w pewnej metryce jest najmniejsza. W przypadku zbioru fonemów odległość określona jest przez parametr *Word Error Rate* (WER), czyli procent niepoprawnie rozpoznanych przez system słów w stosunku do liczby wszystkich słów w zbiorze testowym. Minimalizacja tej funkcji równoważna jest maksymalizacji parametru zwracanego bezpośrednio przez narzędzie HResults wchodzące w skład systemu HTK, którym to parametrem jest

trafność rozpoznania (ang. *accuracy*, Acc), będąca procentowym udziałem poprawnie rozpoznanych słów w stosunku do liczby wszystkich słów w zbiorze testowym. Z powyższego wynika, że $Acc = 1 - WER$. W stworzonym Narzędziu, w metodzie single linkage, w każdym kroku sprawdzana jest trafność rozpoznania po połączeniu każdej możliwej pary fonemów, a następnie trwale łączone są te dwa fonemy, których połączenie daje największą trafność. W kolejnym kroku traktowane są one jako jeden fonem. Jeśli kilka kombinacji dało taki sam, najwyższy, wynik, trwale łączona jest kombinacja uzyskana jako pierwsza. Powyższy algorytm ilustruje schemat:



Rys. 4 Schemat zastosowania metody single linkage

Na przykładowym, 4-literowym alfabecie, przebieg łączenia fonemów tą metodą mógłby wyglądać następująco:



Rys. 5 Schemat działania algorytmu Viterbiego zastosowanego do znajdowania optymalnej kombinacji fonemów. Przykład przedstawiono na 4-literowym alfabecie. Pogrubioną czcionką oznaczono kombinację, która spośród kombinacji z danego kroku dała najlepszą trafność rozpoznania. Kombinacja ta jest rozwijane w następnym kroku, co przedstawione jest również poprzez podpisy linii.

W stworzonym narzędziu programowym szukanie lepszego (mniejszego niż wyjściowy i dającego większą trafność rozpoznania) zbioru fonemów za pomocą metody single linkage realizuje skrypt `test_single_linkage.m`. W rozdziale 4.1 umieszczono opracowanie otrzymanych właśnie tą metodą wyników.

3.2 Baza danych

Do treningu i ewaluacji systemu ASR tworzonego za pomocą HTK przy użyciu różnych zbiorów modeli akustycznych użyto nagrań głosu męskiego z bazy nagrań mowy Corpora. Dane techniczne:

- Liczba plików nagrań treningowych: 10925 (użyto 9830)
- Liczba plików nagrań testowych: 4435 (użyto 165)
- Liczba mówców: 28
- Liczba nagrań na mówcę: ok. 390
- Liczba kanałów: 1
- Częstotliwość próbkowania: 16000 Hz

Wyniki przedstawione w pracy zostały otrzymane przy przeprowadzaniu treningu na nagraniach wszystkich mówców oprócz jednego, a testów na nagraniach jednego, pozostałego mówcy.

3.3 Wymagania do uruchomienia skryptu

Aby uruchomić Narzędzie należy umieścić wszystkie pliki wchodzące w jego skład w głównym folderze bazy nagrań. W tym samym folderze powinny być umieszczone pliki wykonywalne poniższych programów wchodzących w skład systemu HTK[1]:

- HCopy
- HInit
- HRest
- HERest
- HVite
- HResults

Oprócz tego w głównym folderze bazy nagrań powinny znajdować się następujące pliki tekstowe (nazwy zaznaczono `inną czcionką`, mogą jednak być zmienione, pod warunkiem, że zmieni się je także w plikach skryptów):

- Plik konfiguracyjny 1 (dołączony do Narzędzia, `config_c`)
- Plik konfiguracyjny 2 (dołączony do Narzędzia, `config_f`)
- Lista fonemów (`monophones`)

Poniższe pliki nie muszą znajdować się w głównym folderze, ale są wymagane do działania Narzędzia:

- Referencyjna transkrypcja nagrań testowych (`testref`)
- Słownik dla bazy testowej (`dict`)
- Słowosieć dla bazy testowej (`wdnet`)

Wymagane są także następujące pliki, których utworzenie możliwe jest za pomocą funkcji Narzędzia:

- Lista plików treningowych (`train.scp`)
- Lista plików testowych (`test.scp`)

[1] Programy te mogą być umieszczone w innym miejscu na komputerze. Istotne jest to, aby były one uruchamialne z linii poleceń z poziomu folderu, w którym znajdują się pliki Narzędzia.

W katalogu głównym muszą znajdować się podkatalogi zawierające pliki:

- Folder nagrań treningowych
  - Pliki nagrań *.wav
- Folder plików *.lab (może być tym samym, co folder nagrań treningowych)
  - Pliki transkrypcji na poziomie fonemów *.lab
- Folder nagrań testowych
  - Pliki nagrań testowych *.wav.

3.4 Opis uruchomienia i działania skryptu

W niniejszym rozdziale zostanie dokładnie opisane działanie skryptu `test_single_linkage.m` wraz z przygotowaniem do jego działania. Założono, że wszystkie potrzebne pliki są umieszczone w odpowiednich folderach. W przykładach używane są konkretne nazwy plików, które jednak można zmienić według upodobania.

3.4.1 Kodowanie nagrań

Aby przygotować plik ze spisem nagrań do zakodowania (`codetr.scp`) należy posłużyć się skryptem `stworz_codetr.m`. Jest to plik funkcyjny, więc, zakładając, że aktualnym folderem, w którym pracuje MATLAB, jest główny folder bazy, należy wywołać z linii poleceń funkcję `stworz_codetr(folder_wej, sciezka_wyj, folder_wzgl_hcopy)`. Wywołanie może, przykładowo, przybrać postać:

```
stworz_codetr('./nagrania', './codetr.scp', 'nagrania');
```

Po wykonaniu tej funkcji stworzony zostanie plik `codetr.scp`, który zawiera listę wszystkich plików typu *.wav znajdujących się w folderze `nagrania` wraz z listą plików *.mfc zawierających współczynniki MFCC, które to pliki zostaną stworzone przez narzędzie HCopy.

Po stworzeniu pliku `codetr.scp` należy ręcznie, w linii poleceń, uruchomić procedurę kodowania nagrań, jak poniżej:

```
HCopy -T 1 -C config_c -S codetr.scp
```

Teraz w folderze nagrania powinny znajdować się, oprócz plików *.wav pliki *.mfc zawierające zakodowane nagrania.

3.4.2 Tworzenie plików do treningu i ewaluacji

Kolejnym krokiem jest przygotowanie list plików treningowych i testowych. Stworzone Narzędzie oferuje funkcje tworzące pliki do testów typu leave-one-out. W praktyce oznacza to, że w treningu biorą udział nagrania wszystkich mówców w bazie oprócz jednego, losowo wybranego, którego nagrania (i tylko jego) biorą udział w testach. Aby stworzyć zestaw dwóch plików, listy treningowej i testowej, należy zastosować funkcje `stworz_train_loo.m` i `stworz_test_loo.m`. Wywołuje się je następująco:

```
mowca_do_testow = stworz_train_loo( folder_wej, sciezka_wyj_train,
folder_wzgl_hcopy )
stworz_test_loo( folder_wej, sciezka_wyj_test, folder_wzgl_hcopy,
mowca_do_testow )
```

gdzie:

- dla funkcji stworz_train_loo
    - folder_wej – folder zawierający treningowe pliki *.mfc odniesiony do aktualnego folderu (zaczynający się od ‘./’)
    - sciezka_wyj_train – ścieżka, pod którą zapisany zostanie plik z listą plików treningowych
    - folder_wzgl_hcopy – ten sam folder, co w argumencie folder_wej, jednak bez odniesienia do aktualnego folderu (bez ‘./’).
- dla funkcji stworz_test_loo:
    - folder_wej – folder zawierający nagrania treningowe *.wav
    - sciezka_wyj_test – ścieżka, pod którą zostanie utworzony plik z listą plików testowych
    - folder_wzgl_hcopy – j.w.
    - mowca_do_testow – zmienna otrzymana z funkcji `stworz_train_loo` oznaczająca mówcę nieuwzględnionego w treningu, który ma zostać uwzględniony w testach.

Przykładowe wywołanie:

```
mowca_do_testow = stworz_train_loo( ‘./nagrania’, ‘./train.scp’,
‘nagrania’ )
stworz_test_loo( ‘./testy/nagrania’, ‘./test.scp’, ‘testy/nagrania’,
mowca_do_testow )
```

Powyższe wywołanie skutkuje utworzeniem w głównym katalogu dwóch plików: `test.scp` i `train.scp`, zawierających odpowiednio listę nagrań treningowych i listę nagrań testowych.

Powyższe funkcje zakładają, że nazwa każdego pliku dźwiękowego rozpoczyna się od 5-znakowego ciągu znaków identyfikującego mówcę oraz, że w bazie nagrań treningowych i bazie nagrań testowych znajdują się ci sami mówcy. Przykładowe nazwy nagrań dwóch różnych mówców wypowiadających dwóch różne zdania każdy:

```
mwc01zdanie1.wav        mwc02zdanie1.wav
mwc01zdanie2.wav        mwc02zdanie2.wav
```

3.4.3 Konfiguracja funkcji badającej poszczególne kombinacje

W celu ułatwienia wywołania funkcji badającej poszczególne kombinacje, większość jej parametrów nie jest do niej przekazywana jako argumenty, a ustalana jest w pierwszych jej linijkach. Dlatego przed rozpoczęciem optymalizacji zbioru fonemów należy ręcznie zmodyfikować plik `zbadaj_kombinacje2.m` i ustalić wartości zmiennych według wzoru i objaśnień znajdujących się w tym pliku. W szczególności należy pamiętać, aby ręcznie utworzyć katalog dla zmienionych plików *.lab.

3.4.4 Uruchomienie procesu optymalizacji zbioru fonemów

W celu uruchomienia całego procesu szukania optymalnego zbioru fonemów metodą single linkage należy skonfigurować i uruchomić skrypt `test_single_linkage.m`. Konfiguracja polega na edycji wartości 3 zmiennych – `plik_monophones`, gdzie należy wpisać ścieżkę do pliku z listą fonemów, `plik_wyniki`, gdzie należy wpisać ścieżkę, pod którą ma się znaleźć plik z wynikami oraz `prog_skutecznosci` – praca zakończy się, jeśli nie będzie można znaleźć kombinacji fonemów dającej większą bądź równą tej wartości trafność. Uzasadnionym jest wpisanie tu trafności otrzymanej dla niezmienionego zbioru fonemów, obliczonej za pomocą skryptu `test_bez_zmian.m`. Należy zauważyć, że jeśli istnieje już plik o nazwie wpisanej do zmiennej `plik_wyniki`, zostanie on nadpisany. To samo dotyczy wszystkich plików tworzonych przez Narzędzie. Po skonfigurowaniu skryptu należy uruchomić go przez okno programu MATLAB, bądź z linii poleceń. Jeśli wszystkie etapy konfiguracji zostały wykonane poprawnie, rozpocznie się proces optymalizacji zbioru fonemów. Skrypt informuje o postępach w pracy przez standardowe wyjście oraz zapisywanie kolejnych kroków w pliku wynikowym.

3.5 Szczegółowy opis działania skryptu

Poniżej znajduje się opis działania funkcji `zbadaj_kombinacje2.m`. Jest to rozwinięcie bloku „Znajdź parę fonemów dającą największą trafność” z Rys. 7, który wywołuje tę funkcję dla każdej pary fonemów, a następnie wybiera tę, dla której otrzymano największą trafność rozpoznania. Proces ten nie wymaga interakcji z użytkownikiem.

1. Przygotowanie do działania – usunięcie folderów, które będą tworzone od nowa.
2. Zmiana pliku monophones – utworzenie nowego pliku z listą fonemów, w której to liście każde łączone fonemy stanowią jeden fonem.
3. Zmiana pliku dict – utworzenie nowego pliku słownika, w którym każde łączone fonemy stanowią jeden fonem.
4. Zmiana plików *.lab – utworzenie nowych plików transkrypcji na poziomie fonemów, w których każde łączone fonemy stanowią jeden fonem.
5. Utworzenie pliku proto – stworzenie pliku prototypowego modelu HMM.
6. Inicjalizacja HMM – ustalenie początkowych parametrów HMM za pomocą narzędzia HInit.
7. Utworzenie pliku macros - jest on używany przez narzędzia reestymacyjne.
8. Utworzenie pliku hmmdefs – stworzenie pliku z definicjami osobnych modeli dla każdego fonemu. Definicje są na tym etapie identyczne.
9. Kolejne reestymacje modeli, dopóki trafność rozpoznawania wzrasta
    a. Reestymacja modeli z użyciem narzędzia HERest.
    b. Rozpoznanie – uruchomienie rozpoznawania nagrań testowych z użyciem narzęcia HVite.
    c. Określenie trafności z użyciem narzędzia HResults.
10. Największa otrzymana trafność → trafność otrzymana dla danej kombinacji.

4 Wyniki

Podczas testów narzędzia otrzymano wiele statystyk dla różnych parametrów treningu i ewaluacji. Parametry treningu i testu systemu ASR, dla których przedstawiono główne wyniki, to:

- Liczba komponentów mikstury gaussowskiej reprezentującej prawdopodobieństwo emisji: 4
- Liczba współczynników MFCC: 12
- Nie stosowano pierwszych ani drugich pochodnych MFCC

4.1 Wyniki główne

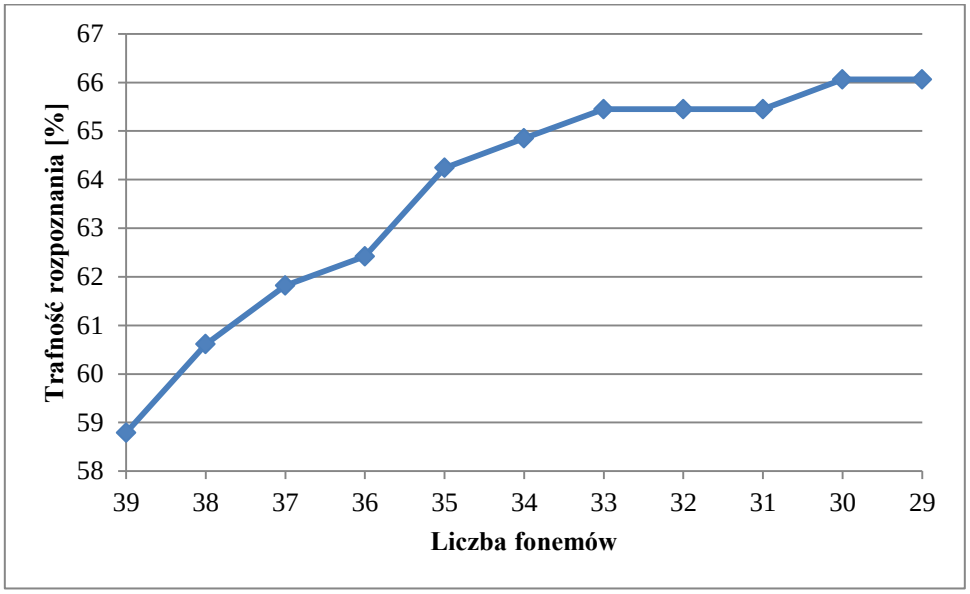
Jako pierwszy wykonano test określający sprawność systemu dla niezmienionego zbioru 39 fonemów. Otrzymano sprawność:

0. *39 fonemów*..... 58,79%

Następnie wykonano test metodą single linkage, która w każdym kroku wybierała do połączenia parę fonemów dającą największą sprawność systemu (należy zauważyć, że nie była w żaden sposób obliczana ani brana pod uwagę odległość pomiędzy modelami fonemów). W kolejnych krokach otrzymywano następujące kombinacje:

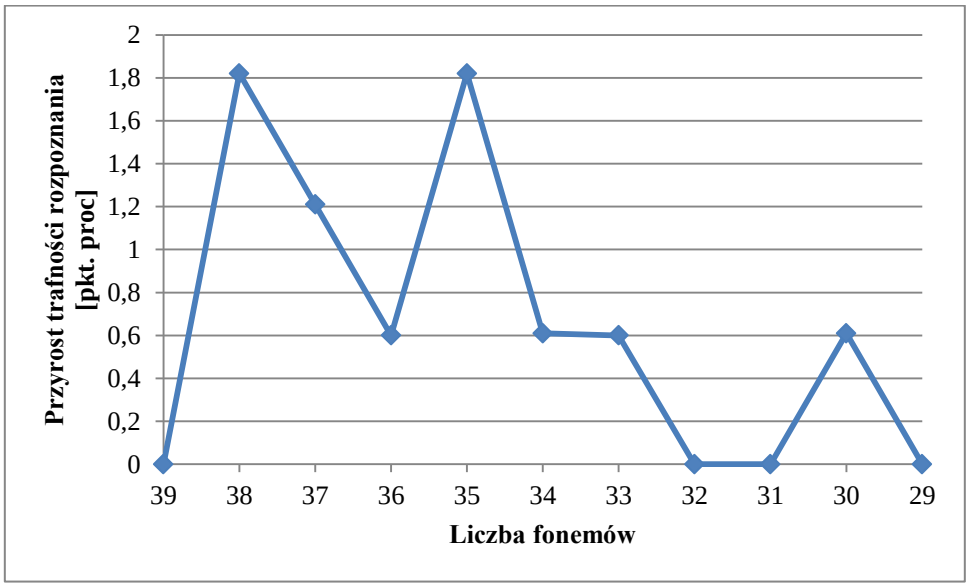
1. *u+w*..... 60,61%
2. *u+w, n+l*..... 61,82%
3. *u+w, n+l, sil+rz* 62,42%
4. *u+w, n+l, sil+rz, j+t* 64,24%
5. *u+w, n+l, sil+rz, j+t, ą+g*..... 64,85%
6. *u+w, n+l, sil+rz, j+t, ą+g, drz+dz* 65,45%
7. *u+w, n+l, sil+rz+sp, j+t, ą+g, drz+dz* 65,45%
8. *u+w, n+l+ę, sil+rz+sp, j+t, ą+g, drz+dz*..... 65,45%
9. *u+w, n+l+ę, sil+rz+sp, j+t, ą+g, drz+dz, k+z*..... 66,06%
10. *u+w, n+l+ę, sil+rz+sp, j+t, ą+g, drz+dz, k+z, ci+s*..... 66,06%

Wyniki te przedstawia Wykres 1. (na tym i następnych wykresach punkty połączono linią w celu zwiększenia czytelności):



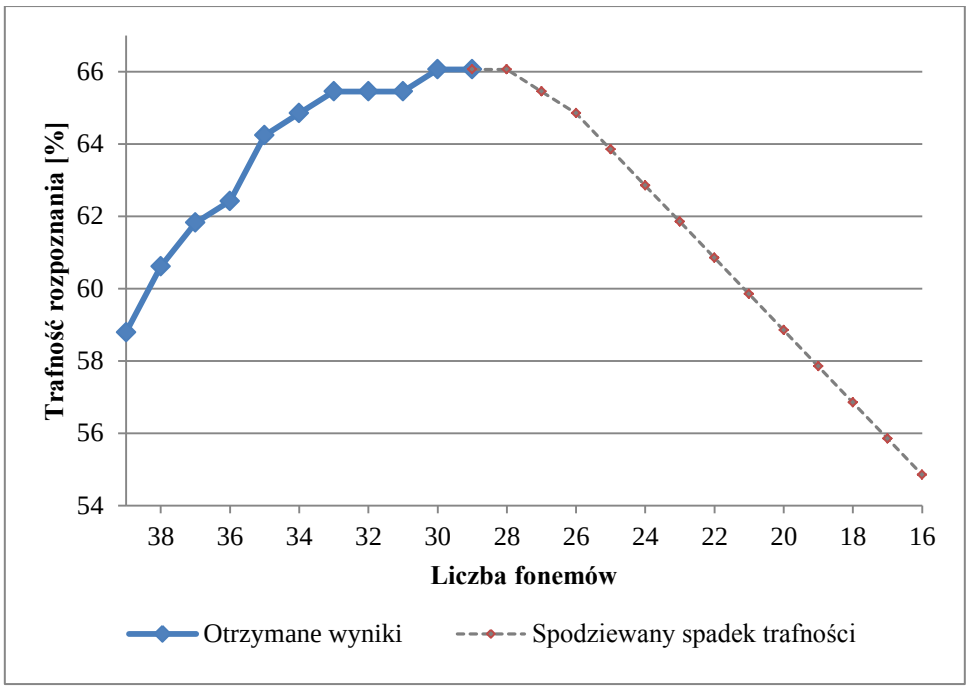
Wykres 1. Trafność rozpoznania względem liczby fonemów

Wykres 2. przedstawia przyrost trafności rozpoznania w każdym kroku względem kroku poprzedniego:



Wykres 2. Przyrost trafności rozpoznania w każdym kroku w stosunku do kroku poprzedniego w zależności od liczby fonemów

Na powyższych wykresach widoczna jest tendencja wzrostowa, która słabnie w ostatnich krokach. W kolejnych krokach spodziewany jest spadek trafności rozpoznania, co ilustruje Wykres 3.



Wykres 3. Spodziewany spadek trafności rozpoznania po większej liczbie łączeń

4.2 Pozostałe wyniki

Oprócz powyższych wyników otrzymanych metodą single linkage, przeprowadzono także kilka innych testów. Pozwoliły one znaleźć kombinacje dające wysoką skuteczność, których nie znaleziono metodą SL.

Metoda SL, jak opisane to zostało w rozdz. 3.1.1, w przypadku znalezienia w danym kroku kilku kombinacji dających taką samą trafność rozpoznania wybiera do połączenia tę, która pojawiła się jako pierwsza. Gdyby jednak wybrana została inna kombinacja, dalsze kroki łączenia fonemów mogą wyglądać zupełnie inaczej. Wykonano test sprawdzający połączenia różnych trójek fonemów. Poprawę sprawności otrzymano m.in. dla kombinacji:

- *ę + rz + sz*..... 60,61%
- *ł + w + N*..... 60,61%

Następnie sprawdzono połączenie obu tych trójek naraz. Otrzymano sprawność:

- *ę + rz + sz; ł + w + N*..... 61,82%

Sprawdzono także kilka kroków algorytmu single linkage w przypadku, gdyby zmienić jego decyzję w drugim kroku, to znaczy zamiast połączeń *u+w, n+l* połączyć fonemy *u+w+c*. Otrzymano następujące wyniki:

- *u+w+c* 60,61%
- *u+w+c; n+r* 61,21%
- *u+w+c+ą; n+r* 63,03%

Pierwszy krok tego algorytmu także mógł przebiec inaczej, to znaczy mogła zostać połączona inna para fonemów, niż *u+w*, ponieważ nie tylko ona dała najwyższy wynik rozpoznania. Podobny wynik dały połączenia:

- *k+rz* 60,61%
- *k+z* 60,61%
- *r+w* 60,61%

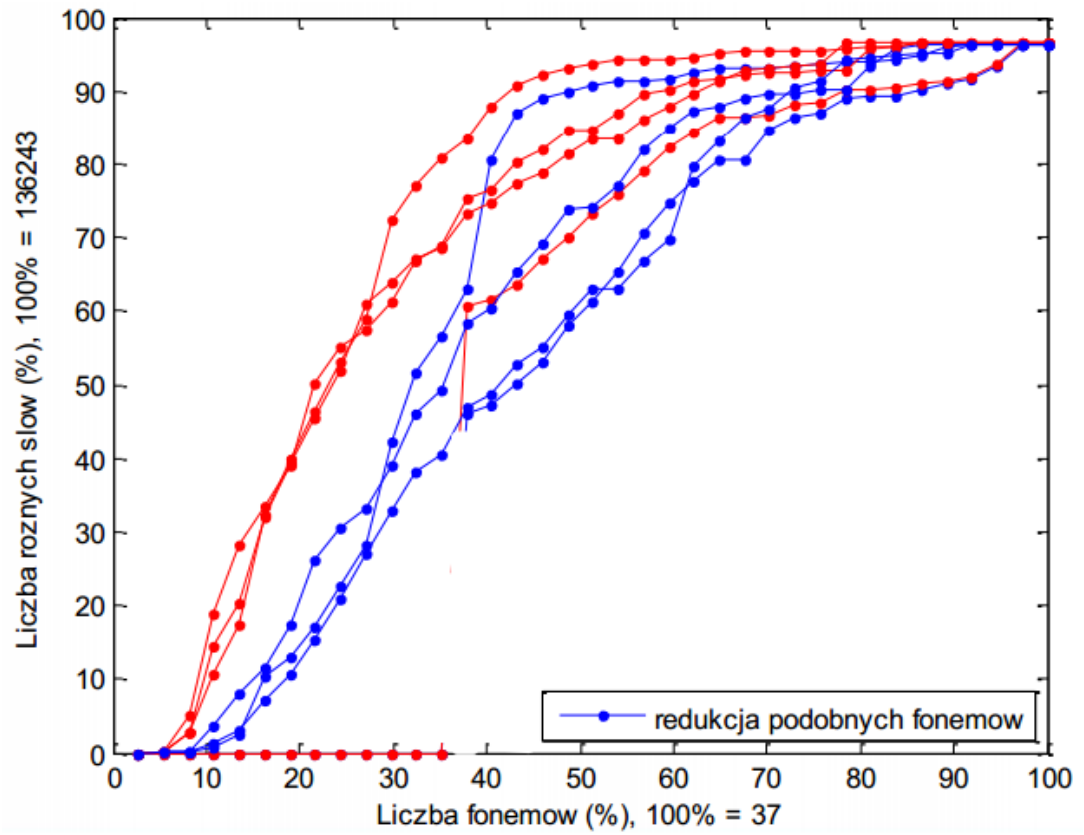
Sprawności otrzymane dla wszystkich połączeń sprawdzonych przez algorytm single linkage dane są w załączniku 1. Można zauważyć, że liczba ścieżek, które mógł obrać algorytm (gdyby niekoniecznie wybierał pierwszą kombinację spośród dających najlepszy wynik), jest bardzo duża.

4.3 Analiza wyników

Łączone pary fonemów nie wydają się oczywiste z punktu widzenia wymowy. Fonemy *n, l, ę*, podobnie jak *j, t*, czy *ą, g* nie brzmią podobnie do siebie. Za podobnie brzmiące można uznać fonemy *drz* i *dz*. Spośród wszystkich łączonych fonemów większość stanowią odpowiadające spółgłoskom.

Ze względu na długi czas trwania obliczeń nie udało się wykazać eksperymentalnie spodziewanego spadku skuteczności. Czasochłonność obliczeń spowodowana jest koniecznością estymacji wszystkich modeli od początku dla każdej sprawdzanej kombinacji. Obliczenia trwają długo nawet pomimo zastosowania serwera obliczeniowego Wavelet2 zespołu DSP (o parametrach technicznych: 2x Xeon X5550 @2.66GHz, 24GB RAM, SSD).

Spodziewany spadek trafności rozpoznania podyktowany jest wstępnymi, dotychczas niepublikowanymi, badaniami zespołu DSP, które wykazały, że przy mniejszej niż ok. 16 liczbie fonemów spada gwałtownie jednoznaczność kodowania słownika słów zmniejszoną liczbą znaków fonetycznych, co ilustruje Wykres 4. Przedstawia on wiele możliwych sekwencji łączenia fonemów (niebieskie linie), wśród których najlepsza dała ww. rezultat (spadek jednoznaczności przy 16 fonemach).



Wykres 4 Zdolność jednoznacznego kodowania słów w zależności od rozmiaru alfabetu. (wykres za: [8])

Do testów wykorzystano stosunkowo „trudny” zestaw nagrań – początkowa sprawność nie jest wysoka – aby uwidocznić pożądany efekt wzrostu sprawności systemu. Ponadto wykorzystano tylko 4 komponenty GMM do modelowania prawdopodobieństw emisji w celu zwiększenia szybkości obliczeń, co obniżyło początkową sprawność systemu (dla niezmienionego zbioru fonemów) i nie pozwoliło w pełni wykorzystać zwiększania ilości danych treningowych przypadających na dany połączony model akustyczny.

5 Dalsza praca

Stworzone Narzędzie będzie rozwijane, aby było w stanie szybciej i efektywniej znaleźć najlepszy z możliwych zbiorów fonemów. Wprowadzone zostaną nowe funkcjonalności wymienione poniżej.

5.1 Reestymacja modeli z użyciem narzędzia HRest

W chwili obecnej reestymacja modeli HMM odbywa się za pomocą narzędzia HERest, wchodzącego w skład systemu HTK. Narzędzie to nie wykorzystuje jednak wszystkich informacji, które zawarte są w bazie nagrań mowy Corpora. HERest automatycznie przyporządkowuje wektory obserwacji do kolejnych emitujących stanów modelu tak, aby otrzymać największą wiarygodność $P(\boldsymbol{O}|\lambda)$, trenując wszystkie modele fonemów występujących w danej wypowiedzi jednocześnie. W bazie Corpora, w plikach transkrypcji nagrań na poziomie fonemów, oprócz samej sekwencji fonemów są także zaznaczone dokładne granice czasowe wystąpienia każdego fonemu – to znaczy: wiemy, od której do której próbki pliku dźwiękowego wypowiadany jest dany fonem. HERest nie wykorzystuje tej informacji w żaden sposób, w przeciwieństwie do narzędzia HRest. HRest trenuje modele pojedynczo – trenując model danego fonemu za sekwencję obserwacji bierze wyłącznie te wektory MFCC, które zostały obliczone z fragmentu nagrania zawierającego ten fonem, według transkrypcji na poziomie fonemów.

5.2 Wielowątkowość

Obliczenia koordynowane przez Narzędzie są bardzo czasochłonne. To, jak długo wykonują się operacje zmieniania plików *.lab, inicjalizacji i reestymacji HMM, czy testowego rozpoznawania, zależy oczywiście od rozmiaru używanej bazy danych. Baza użyta przez autora jest, mówiąc kolokwialnie, niemała, co skutkowało długotrwałymi testami Narzędzia. Obecnie komputery posiadają zazwyczaj wielordzeniowe procesory, które mogą prowadzić wiele obliczeń jednocześnie. Fakt ten można wykorzystać do przyśpieszenia działania Narzędzia. System HTK domyślnie nie obsługuje wielowątkowości – uruchomiony program wykorzystuje tylko 1 rdzeń procesora. Istnieje jednak możliwość podzielenia zadania reestymacji na wiele rdzeni poprzez manipulację parametrami narzędzia HERest i koordynację działania wielu jego instancji. Możliwość ta daje pole do działania przy rozwoju Narzędzia.

Wielordzeniowy procesor można także wykorzystać w nieco inny sposób – zamiast wykonywać jedną operację na wielu rdzeniach, można zlecić każdemu rdzeniowi wykonanie innej operacji. Metoda ta może znaleźć zastosowanie w Narzędziu poprzez podział zakresu badanych kombinacji fonemów. Obecnie, w procesie określania odległości pomiędzy fonemami (rozdział 3.1.1), jeden rdzeń procesora sprawdza po kolei wszystkie możliwe połączenia dwóch fonemów. W przyszłości zadanie to będzie podzielone na wiele rdzeni – każdy będzie sprawdzał tylko pewien podzbiór możliwych połączeń, a następnie wyniki będą łączone w całość i największa trafność będzie wyszukiwana globalnie.

5.3 Manipulacja liczbą komponentów GMM

Obecnie prawdopodobieństwa emisji obserwacji w modelach wszystkich fonemów reprezentowane są przez GMM o 4 komponentach. W momencie połączenia dwóch fonemów w jeden otrzymywany jest model, który trenowany jest danymi zarówno pierwszego, jak i drugiego z połączonych fonemów. Ma on zatem „do dyspozycji" około 2 razy więcej danych treningowych („około", ponieważ zależy to od liczby łączonych fonemów i długości fragmentów nagrań reprezentujących je). Im więcej danych treningowych dostępnych jest dla modelu, tym więcej przypada ich na 1 komponent mikstury gaussowskiej. Aby średnia ilość danych treningowych dla 1 komponentu była stała w ciągu całego procesu poszukiwania najlepszej kombinacji, po połączeniu fonemów należy, proporcjonalnie do ilości danych treningowych dla otrzymanego połączenia, zwiększyć liczbę komponentów mikstur reprezentujących prawdopodobieństwa emisji.

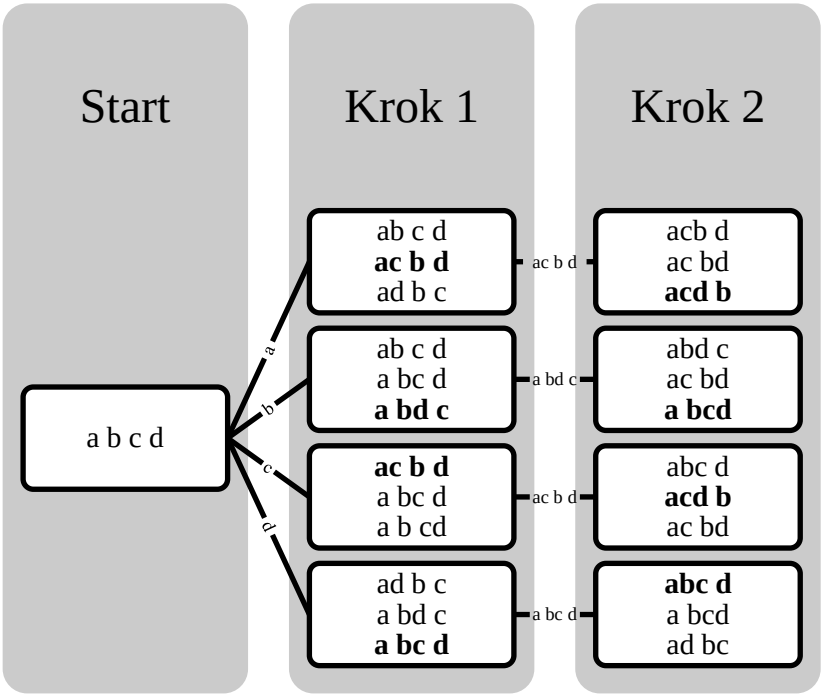
Gdyby liczba komponentów została zbytnio zwiększona już na samym początku procesu, kiedy żadne z fonemów nie są połączone, na jeden komponent mikstury gaussowskiej przypadałoby statystycznie mniej elementów treningowych. Jeśli liczba ta byłaby zbyt mała, model nie został by dobrze wytrenowany, co rzutowałoby bezpośrednio na skuteczność systemu.

5.4 Wykorzystanie algorytmu Viterbiego do poszukiwania najlepszej kombinacji fonemów

Metoda single linkage pozwala uzyskać większą, niż wyjściowa, sprawność systemu ASR, jednak kombinacja fonemów przez nią znaleziona nie musi być kombinacją

optymalną (najlepszą). Algorytm Viterbiego pozwoli na znalezienie optymalnej kombinacji fonemów redukując liczbę kombinacji, które będą musiały być sprawdzone.

W każdym kroku łączenia fonemów sprawdzane są wszystkie możliwości połączenia dla każdego fonemu. W następnym kroku operacja ta jest powtarzana, jednak nie dla wszystkich otrzymanych poprzednio kombinacji, ale tylko dla tych, które dały najlepszy rezultat w obrębie kombinacji otrzymanych z łączenia jednego fonemu. Metodę tę ilustruje poniższy schemat:



Rys. 6 Schemat działania algorytmu Viterbiego zastosowanego do znajdowania optymalnej kombinacji fonemów. Przykład przedstawiono na 4-literowym alfabecie. Pogrubioną czcionką oznaczono kombinacje, które spośród danego podzbioru dały najlepszą trafność rozpoznania. Kombinacje te są rozwijane w następnym kroku, co przedstawione jest również poprzez podpisy linii.

Optymalna kombinacja fonemów to najlepsza kombinacja spośród uzyskanych w ostatnim kroku.

6 Podsumowanie i wnioski

Podczas tworzenia niniejszej pracy szczegółowo zapoznano się z procesem przetwarzania mowy – zagadnieniami teoretycznymi związanymi z tematem oraz praktycznym zastosowaniem niektórych algorytmów w systemie HTK.

Stworzone zostało narzędzie programowe w środowisku MATLAB pozwalające na znalezienie kombinacji fonemów dającej lepszą skuteczność rozpoznawania mowy przez system HTK. Narzędzie to poszukuje coraz lepszych kombinacji iteracyjnie, za pomocą metody single linkage.

Przedstawiono wyniki przeprowadzonych za pomocą stworzonego narzędzia testów, które to wyniki wykazały jego skuteczność. Otrzymano przyrost skuteczności rozpoznawania mowy wynoszący 7,27 punktu procentowego (początkowa skuteczność wynosiła 58,79%, końcowa – 66,06%).

W pracy umieszczono wyniki testów przeprowadzonych tylko przy jednym zestawie parametrów, to znaczy przy stałej liczbie komponentów GMM modelującej prawdopodobieństwo emisji w modelach fonemów oraz przy liczbie współczynników MFCC równej 12 (bez użycia pochodnych). Wyniki dla innych niż użyte w testach parametrów mogą być inne. Zostanie to sprawdzone w przyszłości.

Przestawiono także możliwości dalszego rozwijania stworzonego narzędzia. Zostaną one w przyszłości zaimplementowane, co skutkować będzie poprawieniem wydajności i efektywności narzędzia.

Bibliografia:

[1] edycja: Keller E.: *Fundamentals of Speech Synthesis and Speech Recognition*, Chichester, New York, Brisbane, Toronto, Singapore, John Wiley & Sons 1994

[2] Szostek K., rozprawa doktorska: *Rozpoznawanie mowy metodami niejawnych modeli Markowa HMM*, Kraków 2006

[3] Young S., Evermann G., Gales M., Hain T., Kershaw D., Liu X. et al.: *The HTK Book*, http://htk.eng.cam.ac.uk/docs/docs.shtml, 2006

[4] HTK Speech Recognition Toolkit, http://htk.eng.cam.ac.uk/, (24.12.2012)

[5] Rabiner L., Juang B-H.: *Fundamentals of speech recognition*, Prentice-Hall, New Jersey 1993

[6] An Introduction to Artifficial Intelligence, www.ai-class.com, (24.12.2012)

[7] Gałka J., rozprawa doktorska: *Optymalizacja parametryzacji sygnału w aspekcie rozpoznawania mowy polskiej*, Kraków 2008

[8] Wydajny psychoakustyczny schemat dekompozycji falkowej, http://149.156.196.114/~jgalka/dsp/Galka_Seminarium_DSP.pdf, (13.01.2013)