Model akustyczny języka polskiego na potrzeby automatycznego rozpoznawania mowy.

Wersja: 1.0

Data: 2009-03-26

Autor: Łukasz Olczak

Status: ALFA

Historia wersji

|  |  |  |  |
| --- | --- | --- | --- |
| Wersja | Data | Autor | Zmiany |
| 1.0.0 | 2009-03-26 | Łukasz Olczak | Utworzenie pierwszej wersji |

[1 Wstęp 5](#_Toc225845792)

[2 Wprowadzenie 5](#_Toc225845793)

[2.1 Definicja problemu rozpoznawania mowy 5](#_Toc225845794)

[2.2 HMM – ukryte modele Markova 7](#_Toc225845795)

[2.3 EM algorytm - Baum-Welch 8](#_Toc225845796)

[2.4 Wady algorytmu Baum-Welch 9](#_Toc225845797)

[2.5 Trening dyskryminujący 9](#_Toc225845798)

[2.6 Podsumowanie – wady HMM-GMM 11](#_Toc225845799)

[3 Architektura systemu ASR typu HMM-NN 12](#_Toc225845800)

[3.1 Wstęp 12](#_Toc225845801)

[3.2 Sieć neuronowa w ASR 12](#_Toc225845802)

[3.3 Wnioskowanie statystyczne za pomocą MLP 13](#_Toc225845803)

[3.4 Architektura – hybryda HMM-MLP 14](#_Toc225845804)

[3.5 Ekstrakcja cech akustycznych mowy 16](#_Toc225845805)

[3.6 Jednostki fonetyczne 16](#_Toc225845806)

[3.7 Trening 17](#_Toc225845807)

[3.8 Początkowa segmentacja 18](#_Toc225845808)

[3.9 MLP jako estymator prawdopodobieństw emisji w HMM 19](#_Toc225845809)

[3.10 Wyszukiwanie Viterbi’ego 19](#_Toc225845810)

[4 Szczegóły implementacyjne 20](#_Toc225845811)

[4.1 Parametry 20](#_Toc225845812)

[4.2 Technologie 21](#_Toc225845813)

[4.2.1 Spring Framework 21](#_Toc225845814)

[4.2.2 Spring RCP 21](#_Toc225845815)

[4.2.3 Joone - Java Object Oriented Neural Engine 21](#_Toc225845816)

[4.2.4 JCR Jack Rabbit 21](#_Toc225845817)

[4.2.5 TestNG 21](#_Toc225845818)

[4.2.6 Maven2 22](#_Toc225845819)

[5 Do zrealizowania 22](#_Toc225845820)

[5.1 Wydzielone zadania do wykonania 22](#_Toc225845821)

[5.1.1 Szkielet Gui 22](#_Toc225845822)

[5.1.2 Obsługa słownika 22](#_Toc225845823)

[5.1.3 Obsługa mikrofonu 22](#_Toc225845824)

[5.1.4 Budowa korpusu językowego 22](#_Toc225845825)

[5.1.5 Frontend (MFCC, PLP) 23](#_Toc225845826)

[5.1.6 Trening – Connectionist Viterbi training 23](#_Toc225845827)

[5.1.7 Segmentacja początkowa 23](#_Toc225845828)

[5.1.8 Dekodowanie 23](#_Toc225845829)

[5.1.9 Bazowe struktury danych 23](#_Toc225845830)

[5.1.10 Wizualizacja 23](#_Toc225845831)

[5.1.11 Automatyzacja 23](#_Toc225845832)

[5.1.12 Dokumentacja 23](#_Toc225845833)

[5.2 Planowany harmonogram 23](#_Toc225845834)

[6 Literatura 24](#_Toc225845835)

# Wstęp

Celem dokumentu jest opisanie najważniejszych zagadnień systemu automatycznego rozpoznawania mowy (ASR). Dokument zawiera zarówno aspekty teoretyczne jak i praktyczne. Ma on za zadanie usystematyzowanie pozyskanej do tej pory wiedzy, przygotowanie fundamentów pod dokument pracy magisterskiej, wyłapanie nieścisłości, potencjalnych problemów, oraz wydzielenie najważniejszych podzadań i temat‏ów do zrealizowania. Zatem dokument jest próbą przymiarki pod kompletną specyfikację systemu ASR, który planuję zaimplementować w ciągu semestru X.

# Wprowadzenie

## Definicja problemu rozpoznawania mowy

Problem rozpoznawania mowy można w skrócie streścić do:

*Jaka jest najbardziej prawdopodobna sentencja W z danego języka L (przestrzeni słów) dla próbki danych akustycznych O, które chcemy rozpoznać?*

Przy czym sentencję W traktujemy jako ciąg słów W = w1,w2,w3,…,wn; natomiast akustyczne dane wejściowe O=o1,o2,o3,…,ot są kolejnymi wektorami cech wyekstrahowanymi z fali akustycznej sygnału mowy.

Sformułowanie problemu rozpoznawania mowy jako wyszukanie najbardziej prawdopodobnego słowa dla danej sekwencji cech akustycznych (wypowiedzianej frazy) nasuwa klasyfikację problemu rozpoznawania do grupy problemów z dziedziny wnioskowania statystycznego. A więc implementacja systemu ASR polega na opracowaniu odpowiedniego modelu probabilistycznego – modelu akustycznego i modelu języka.

Jeżeli ciąg cech akustycznych O potraktujemy jako zbiór zdarzeń zachodzących w doświadczeniu polegającym na wymawianiu kolejnych fonemów, a zbiór wymawianych słów jako zbiór dopuszczalnych hipotez to rozpoznawanie mowy możemy potraktować jako szczególny przypadek wnioskowania (klasyfikacji) w oparciu o regułę Bayesa.

Pamiętając, że reguła Bayesa opisana jest wzorem:

,

gdzie

Hi – zbiór hipotez, będących układem zdarzeń zupełnych

P(Hi) – to prawdopodobieństwo a priori

P(E) – tzw. prawdopodobieństwo brzegowe zajścia zdarzenia E (obserwacji)

P(Hi|E) – prawdopodobieństwo a posteriori zajścia zdarzenia Hi wiedząc że zaszło zdarzenie E

P(E|Hi) – prawdopodobieństwo warunkowe zajścia zdarzenia E wiedząc że zaszło zdarzenie Hi

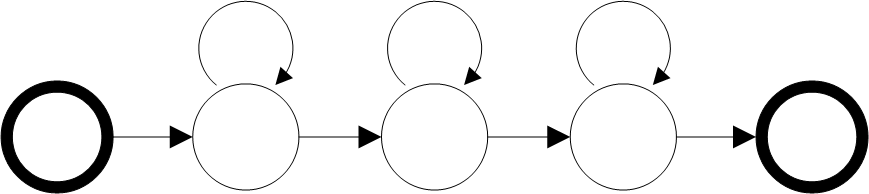
Zatem możemy zdefiniować nasz problem rozpoznawania mowy bardziej formalnie:

A więc przeszukiwanie utworzonego modelu probabilistycznego odbywa się na podstawie kryterium MAP (Maximum a Posteriori), gdzie P(W) mamy z modelu języka natomiast P(O|W) z modelu akustycznego. Ponieważ P(O) nie zmienia się dla każdej rozważanej sekwencji W jest więc stałe. A więc szukając maksimum a posteriori możemy pominąć mianownik.

## HMM – ukryte modele Markova

Generalizując problem rozpoznawanie mowy można powiedzieć, że polega on na dobraniu dla każdego wypowiedzianego wyrażenia najlepiej pasującego wzorca z korpusu językowego. Jednak nawet dla tego samego człowieka struktura akustyczna tej samej wypowiedzi, w szczególności czasowa, może się zmieniać. A więc proste porównywanie wypowiedzi ze wzorcem staje się skomplikowane z dwóch powodów. Po pierwsze czas trwania poszczególnych fonemów (elementarna część mowy) różni się u poszczególnych osób. Rozwiązaniem tego problemu może być normalizacja wypowiedzi i wzorca. Jednakże drugą przeszkodą jest fakt, że tempo wymowy poszczególnych fonemów nie jest stałe w danym słowie. Czyli algorytm optymalnego wyrównania fonemów wzorca i wypowiedzi nie jest liniowy, co komplikuje problem skutecznego rozpoznawania mowy. Problem optymalnego wyrównania fonemów znany jest w literaturze pod pojęciem *Dynamic Time Warping* [4] [5].

Jednakże najbardziej elastycznym i skutecznym rozwiązaniem dopasowywania wymiaru czasu pomiędzy wzorcem i wypowiedzią do rozpoznania są ukryte modele Markova HMM. Ukryty model Markova jest parą procesów stochastycznych: ukrytego łańcuchu Markova i obserwowalnego procesu opisanego rozkładem prawdopodobieństwa dla każdego stanu ukrytego. Innymi słowy: obserwowalne zdarzenia (np. cechy akustyczne) są modelowane za pomocą rozkładu prawdopodobieństwa emisji dla poszczególnych ukrytych stanów, dyskretnego w czasie, pierwszego rzędu procesu Markova.



Ukryty model Markova opisują następujące komponenty:

Q =q1,..,qN – zbiór stanów ukrytych

A=a00a02…an1..ann – macierz prawdopodobieństw przejść ze stanu i do j

O=o1o2..oN – zbiór obserwacji z zadanego słownika

B=bi(ot) – zbiór prawdopodobieństw emisji zdarzenia (obserwacji) ot dla stanu i

Często dodaje się również sztuczne stany qstart i qend

W zależności od rozmiarów słownika model HMM reprezentuje słowo lub fonem. W aplikacjach ASR struktura sieci HMM jest typu od lewej do prawej tzw. sieć Bakis’a.

## EM algorytm - Baum-Welch

Powszechnie stosowanym algorytmem budowy modelu statystycznego do rozpoznawania mowy jest algorytm Baum’a-Welch’a (Forward-Backward algorithm) [3]. Algorytm ten należy do szerszej klasy algorytmów EM (Expectation - maximization). Algorytm expectation-maximazation jest popularną metodą w statystyce używaną do dopasowania modelu probabilistycznego do danych treningowych. Parametry modelu statystycznego są tak dopasowywane, aby wiarygodność estymatora nieznanej wartości parametru (parametry modelu są nieobserwowalnymi zmiennymi ukrytymi) była największa dla danych treningowych. Innymi słowy tak dobieramy parametry modelu, aby model jak najlepiej pasował do danych treningowych. Jest to tzw. metoda największej wiarygodności wprowadzona przez Fishera [6].

Algorytm EM jest algorytmem iteracyjnym, estymującym w każdym kroku co raz dokładniej pewne nieznane wielkości na podstawie znanych obserwacji skorelowanych z estymowanymi parametrami. W pierwszym kroku estymowane parametry są inicjalizowane wartościami losowymi albo wartościami początkowymi jeśli posiadamy pewną wiedzę a priori o parametrach modelu. Następnie każda iteracja składa się z dwóch kroków:

* E-step – estymuje oczekiwane wartości zmiennych ukrytych, na podstawie wartości estymat parametrów z poprzedniej iteracji
* M-step – re-estymuje parametry rozkładu prawdopodobieństwa (tutaj macierze A i B modelu HMM) tak aby zmaksymalizować wiarygodność danych treningowych na podstawie estymat zmiennych ukrytych z fazy E-step tej samej iteracji

Algorytm Baum’a-Welch’a wykorzystywany jest do wyznaczenia nieznanych parametrów modelu HMM – macierzy A i B. Nauka odbywa się trybie nienadzorowanym jednak wartości początkowe A i B są bardzo ważne. Toteż w aplikacjach rozpoznawania mowy struktura modelu HMM jest ustawiana ręcznie (sieć Bakis’a – struktura od lewej do prawej) natomiast typ rozkładu prawdopodobieństwa emisji (B) jest zadany z góry.

## Wady algorytmu Baum-Welch

Algorytm Baum’a-Welch’a koncentruje się na dopasowaniu najlepszego modelu HMM (dla każdego słowa mamy osobny model HMM) do danych treningowych zamiast na wyróżnieniu (ang. discriminate) najlepszego modelu z pozostałych modeli.

Dodatkowo algorytm Baum’a-Welch’a wprowadza założenie co do kształtu rozkładu cech akustycznych dla fonemów. Najczęściej są to komponenty GMM – wielowymiarowego rozkładu normalnego. Często takie założenie jest zbyt daleko idące. Toteż na chwilę obecną rzadko stosuje się w wydajnych systemach ASR algorytm Baum’a-Welch’a + GMM.

## Trening dyskryminujący

Przypomnijmy, za [2], że w metodzie największej wiarygodności (MLE) trenujemy parametry modelu akustycznego (A i B) tak aby zmaksymalizować wiarygodność danych treningowych. Rozważmy pewną sekwencję obserwacji O i pewien model HMM Mk odpowiadający sekwencji Wk.

Kryterium MLE maksymalizuje wiarygodność zdefiniowaną jako:

Naszym celem podczas implementacji systemu rozpoznawania mowy jest maksymalizacja poprawnych transkrypcji dla jak największych próbek dźwiękowych. Jednakże powyższe kryterium nam tego nie gwarantuje. Kryterium MLE zapewnia największe prawdopodobieństwo emisji obserwacji O (wymowa słowa Wk) dla modelu HMM Mk odpowiadającemu słowy Wk.

A więc celem konstruktorów systemów rozpoznawania mowy jest takie wytrenowanie parametrów modelu akustycznego aby prawdopodobieństwa wyboru prawidłowego modelu Mk dla wypowiedzi Ok było większe niż prawdopodobieństwo wyboru modelu Mj gdzie j≠k.

Kryterium CMLE maksymalizuje prawdopodobieństwo warunkowe:

Stosując twierdzenie Bayes’a:

Zapisując mianownik jako sumę modelu poprawnego i modeli niepoprawnych dla obserwacji O:

Zatem aby zmaksymalizować musimy inkrementacyjnie zmieniać λ tak aby zwiększać prawdopodobieństwo poprawnego modelu (wyrażenie w liczniku) i jednocześnie zmniejszać prawdopodobieństwo wyboru niepoprawnego modelu (suma w mianowniku).

Zatem kryterium dyskryminujące minimalizuje współczynnik błędu.

Powyższy trening posiada tzw. własność dyskryminacji niepoprawnych modeli/hipotez.

W skrócie trening dyskryminujący [12] zmierza do zamodelowania brzegów klas – uczy się różnic pomiędzy klasami – zamiast konstruować dokładny model dla każdej klasy. Minimalizuje prawdopodobieństwo wyboru niepoprawnej klasy jednocześnie maksymalizując prawdopodobieństwo wyboru poprawnego modelu, zamiast estymować dokładnie parametry lub rozkład prawdopodobieństwa modelu klasy.

## Podsumowanie – wady HMM-GMM

* Założenie, że wszystkie prawdopodobieństwa zależą tylko i wyłącznie od aktualnego stanu – co jest nieprawdą dla aplikacji rozpoznawania mowy.
* Mowa nie jest procesem stacjonarnym. Dlatego modele HMM nie potrafią dobrze zamodelować koartykulacji, gdzie rozkład prawdopodobieństwa mocno zależy od poprzednich stanów.
* Założenie że sąsiednie obserwacje O są nieskorelowane – ponieważ HMM przetwarza jedynie jedna ramkę jednocześnie należy wektor cech rozszerzyć o informację kontekstową (np. delta pierwszego drugiego rzędu pomiędzy sąsiednimi ramkami)
* Założenie że poszczególne składowe wektora cech są niezależne.
* Założenie co do rozkładu prawdopodobieństwa emisji obserwacji – nie zawsze jest to prawidłowo podejście. Pomimo, że rozkład GMM może aproksymować dowolny rozkład prawdopodobieństwa – to nie wiemy ile należy użyć komponentów Gausa, a więc liczba ta jest dobierana podczas procesu uczenia i należy ją traktować jako kolejną hipotezę.
* Kryterium MLE prowadzi do słabych właściwości dyskryminujących dla modelu akustycznego. Zmiana kryterium na maksymalizację informacji wzajemnej poprawia tę jakoś rozpoznawania ale jest skomplikowana i trudna do implementacji.

# Architektura systemu ASR typu HMM-NN

## Wstęp

Ponieważ klasyczne metody HMM opierają się na wielu niepoprawnych założeniach, które pociągają za sobą niezadowalającą skuteczność rozpoznawania mowy, dlatego badacze szybko znaleźli rozwiązanie wyżej wymienionych problemów stosując sieci neuronowe. Główne cechy hybryd sieć neuronowa – HMM to umiejętność nauki skomplikowanych analitycznie funkcji, wydajne generalizowanie, równoległość oraz trening dyskryminujący. Sieć neuronowa doskonale nadaje się do budowy modelu akustycznego, lecz słabo modeluje wymiar czasu (różne czasy trwania fonemów dla poszczególnych mówców) oraz nie nadaje się do tworzenia kompozycji, dlatego powszechnie stosuje się w tym celu hybrydę NN-HMM. Sieć neuronowa dokonuje klasyfikacji akustycznej (dla wektora cech wejściowych przypisuje prawdopodobieństwa a posteriori poszczególnych klas (fonemów)) natomiast HMM modeluje wymiar czasu [7]. Eksperymenty pokazują że hybryda NN-HMM posiada o wiele większą skuteczność niż klasyczne HMM-GMM.

## Sieć neuronowa w ASR

System połączeniowy (ang. connectionist system) składa się z połączonych ze sobą prostych węzłów (w przypadku SSN zwanych sztucznymi neuronami) realizujących elementarne operacje na sygnałach wejściowych. W sieci tej sygnał wyjściowy jest propagowany na wejścia węzłów kolejnej warstwy.

Cała informacja pozyskana z danych wejściowych jest reprezentowana przez strukturę i parametry węzłów w sieci. A więc sieć ta modeluje złożone analitycznie zależności pomiędzy danymi wejściowymi i wyjściowymi bez jawnych założeń statystycznych (np. niezależność danych wejściowych, kształt rozkładu).

Aby obejść problemy przedstawione w punkcie 2.6. w systemach ASR powszechnie stosuje się hybrydę HMM – wielowarstwowy perceptron (MLP). Zastosowanie MLP w systemach ASR jest atrakcyjne z następujących powodów:

* sieć MLP doskonale nadaje się do uczenia
* trening dyskryminujący, który minimalizuje liczbę błędów
* potrafi w teorii wygenerować dowolną nieliniową funkcję danych wejściowych
* nie wymaga żadnych mocnych założeń co do rozkładu danych wejściowych
* łatwo można dołączyć informację kontekstową
* można łączyć różne wektory cech mowy (np. MFCC, PLP)
* zdolność adaptacyjna sieci

Niestety sieć MLP nie nadaje się do modelowania wzorców posiadających wymiar czasu. Sieć neuronowa może być tutaj wykorzystana jedynie do klasyfikacji elementarnych segmentów mowy takich jak fonemy, subfonemy itp.

Ponieważ HMM doskonale reprezentuje wymiar czasu oraz reprezentuje podejście hierarchiczne (zdania modelujemy za pomocą sekwencji słów, słowa za pomocą sekwencji fonemów, fonemy za pomocą sekwencji stanów procesu Markova) wskazane jest tutaj połączenie obu technik. Dlatego w [7] zaproponowano wykorzystanie hybrydy HMM-MLP, w której to sieć neuronowa dokonuje decyzji lokalnych, natomiast HMM dokonuje decyzji globalnych. Pomysł ten został rozwinięty w pracach [8, 9, 10, 11]. Ponieważ wiedza dziedzinowa na temat sygnału mowy jest zbyt uboga aby dokonać właściwych założeń dlatego ilość wiedzy a priori ograniczamy do minimum – naszą jedyną wiedzą a priori jest struktura cech MFCC, PLP

## Wnioskowanie statystyczne za pomocą MLP

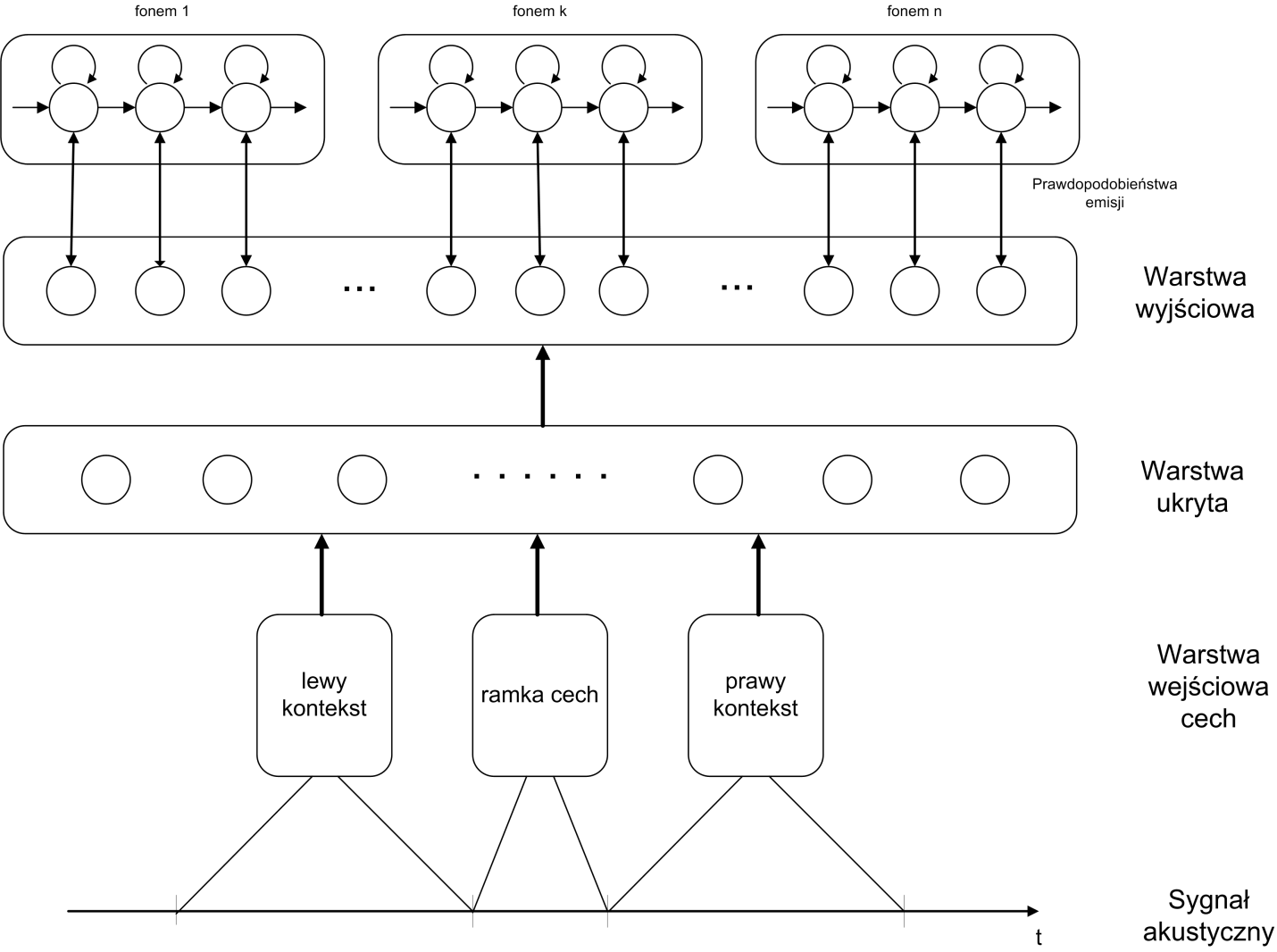
Celem uczenia MLP jest skojarzenie wektora wejściowego z pożądanym wektorem wyjściowym. Jeżeli naszym zadaniem jest klasyfikacja N klas wtedy sieć z N wyjściami powinna zostać użyta (po jednym wyjściu dla każdej klasy). Wtedy podczas nauki jedynka na wyjściu powinna oznaczać poprawną klasę, zero powinno zostać przyporządkowane do pozostałych klas. Nauczona sieć MLP wg powyższego schematu za pomocą algorytmu wstecznej propagacji estymuje prawdopodobieństwa a posteriori klas, tzn. że po nauce typu 1-z-N wartości wyjść wielowarstwowego perceptronu dla danego wektora wejściowego x będą estymatorami prawdopodobieństwa a posteriori P(ci|x) dla klasy ci skojarzonej z i-tym wyjściem. Powyższe twierdzenie można udowodnić dla różnorodnych błędów wykorzystywanych podczas algorytmu wstecznej propagacji [1][7].

## Architektura – hybryda HMM-MLP

Ze względu na trening dyskryminujący hybryda HMM-MLP wymaga wyraźnie odseparowanych, niezależnych fonemów. Toteż fonemy zależne od kontekstu (trifony, bifony) nie dają takiej poprawy błędu rozpoznawania jak w przypadku klasycznych systemów HMM (ich klasy nie są dobrze od siebie odseparowane). Dlatego w pracy zamierzam użyć fonemów niezależnych od kontekstu zwiększając ich „odległości” akustyczne poprzez wprowadzenie osobnych fonemów dla części stacjonarnej mowy i koartykulacji.

Hybryda HMM-MLP łączy zdolność rozpoznawania wzorców czasowych (za pomocą HMM) z klasyfikacją wzorców statycznych (za pomocą wielowarstwowego perceptronu). Słowa w takim systemie są modelowane za pomocą ukrytego modelu Markova (struktura od lewej do prawej, prawdopodobieństwa przejść pomijamy) do dekodowania wykorzystywany jest algorytm Viterbi’ego, natomiast sieć neuronową delegujemy do wyznaczania prawdopodobieństw emisji.

W systemach hybrydowych wykorzystuje się dwa typy MLP: o strukturze rekurencyjnej i sieć jednokierunkowa. Jak można przeczytać w [8] struktura sieci jest zazwyczaj dobierana eksperymentalnie w zależności jakie części sygnału akustycznego modelujemy. Jeżeli modelujemy całe słowa to sieć rekurencyjna daje najlepsze rezultaty, natomiast gdy modelujemy fonemy to sieć typu jednokierunkowego jest wydajniejsza [8].



Warstwa wejściowa jest podzielona na trzy bloki, jeden dla ramki środkowej i po jednym dla lewego i prawego kontekstu. Każdy blok jest podzielony na sub-bloki każdy dla innego typu cech akustycznych [8] (ponieważ empirycznie udowodniono że taka struktura posiada o wiele większą wydajność). Warstwa ukryta jest w pełni połączona z warstwą wyjściową, która estymuje prawdopodobieństwa emisji stanów HMM (oczywiście po przekształceniu wg reguły Bayesa). Każde wyjście wielowarstwowego perceptronu odpowiada jednemu stanowi procesu Markova.

## Ekstrakcja cech akustycznych mowy

Ekstrakcja cech akustycznych jest jedyną wiedzą a priori w architekturze hybrydowej HMM-MLP. Ponieważ wiedza dziedzinowa na temat akustyki mowy jest w dalszym ciągu niewystarczająca aby dokonać ekstrakcji odpowiednich cech z sygnału akustycznego mowy, toteż odpowiedni algorytm ekstrakcji cech dopiera się eksperymentalnie. Ze względów historycznych większości współczesnych systemów ASR używa wektora cech MFCC (omówionego szczegółowo w [2]). Jednakże jak pokazują doświadczenia algorytm PLP lub RASTA-PLP [13] [14] daje znacznie lepsze rezultaty niż MFCC [15].

W pracach [15] [16] [17] przedstawiono propozycję znacznej redukcji błędu rozpoznania poprzez ekstrakcję kilku różnorodnych wektorów cech akustycznych z sygnału mowy i wprowadzenie każdego do warstwy wejściowej hybrydy HMM-MLP z punktu 3.4 – każdemu wektorowi odpowiada oddzielny sub-blok warstwy wejściowej wielowarstwowego perceptronu.

W własnej implementacji systemu ASR planuje wykorzystanie kodu ekstrakcji cech MFCC oraz algorytmu PLP (jeśli nie zabraknie czasu) z systemu Sphinx4 plus porównanie redukcji błędu dla różnych konfiguracji.

## Jednostki fonetyczne

Ponieważ sieć neuronowa wykorzystywana do klasyfikacja wymaga wyraźnie odseparowanych, nienachodzących na siebie klas dlatego należy użyć fonemów niezależnych od kontekstu (context independent phonems). W pracy [8] zaproponowano wykorzystanie jednostek fonetycznych Stationary-Transitional Units (STU) opisanych w [19]. Fonemy te można wykorzystać w architekturze z punktu 3.4. Jednostki te zostały podzielone na dwie części: jednostki stacjonarne i jednostki przejściowe. Pierwsze są stacjonarnymi częściami sygnałami akustycznego fonemów niezależnych od kontekstu, natomiast druga klasa to wszystkie dopuszczalne przejścia pomiędzy jednostkami stacjonarnymi. Dla języka włoskiego wszystkich fonemów STU jest 375.

Na przykład wg słownika [20] transkrypcja fonetyczna w alfabecie IPA słowa marchew wygląda następująco [marxev], stosując jako jednostki fonetyczne STU słowo to zostanie zamodelowane za pomocą następujących fonemów:

<@-m> <m> <m-a> <a> <a-r> <r> <r-x> <x> <x-e> <e> <e-v> <v> <v-@>. Fonemy <m> <a> <r> <x> <e> <v> są jednostkami stacjonarnymi (@ - oznacza ciszę), natomiast <m-a> <a-r> <r-x> <x-e> <e-v> są jednostkami przejściowymi pomiędzy częściami stacjonarnymi.

## Trening

W klasycznym systemie ASR opartym o HMM podczas nauki estymujemy prawdopodobieństwo emisji obserwacji O dla modelu M wykorzystując metodę największej wiarygodności która uwzględnia tylko poprawną klasę. Natomiast w hybrydzie HMM-MLP estymujemy prawdopodobieństwo a posteriori P(M|O) uwzględniając podczas nauki wszystkie klasy (dyskryminując klasy niepoprawne) – a więc podczas nauki sieć neuronowa stara się jasno odseparować wszystkie klasy w przestrzeni - gdzie n jest długością wektora cech akustycznych. Z tego powodu algorytm nauki modelu HMM-MLP jest zupełni inny niż klasyczny algorytm Baum’a-Welch’a.

Do nauki hybrydy HMM-MLP zamierzam wykorzystać algorytm connectionist viterbi training (CVT) przedstawiony po raz pierwszy w [11] a zaimplementowany i wdrożony w systemie produkcyjnym po raz pierwszy w [8].

CVT jest algorytmem iteracyjnym, który polega na stopniowym przesuwaniu granic segmentacji aż do uzyskania segmentacji optymalnej.

Oto kolejne kroki algorytmu:

Inicjalizacja:

* Inicjalizujemy MLP małymi losowymi wagami
* Posiadamy początkową segmentację (bootstrap segmentation)

Kolejne iteracje:

* Ładujemy aktualną segmentacje
* Dla segmentacji przypisanej tej iteracji uczymy sieć neuronową za pomocą algorytmu wstecznej propagacji błędu
* Wykorzystując sieć neuronową z poprzedniego kroku wyznaczamy nową segmentację (danych treningowych)
* Aktualizujemy segmentację za pomocą funkcji :

snew=asold+(1-a)scomp

gdzie: snew – segmentacja dla kolejnej iteracji, sold – segmentacja z poprzedniej iteracji, scomp – nowa segmentacja wyznaczona przez MLP.

* Zaczynamy naukę z wartością parametru a=1.0 i stopniowo zmniejszamy jego wartość podczas nauki.

Naukę kończymy w momencie gdy wartość błędu się ustabilizuje. Wartość błędu wyznaczamy na podstawie danych testowych (nie treningowych)

## Początkowa segmentacja

Zazwyczaj początkową segmentację wyznaczamy za pomocą klasycznego algorytmu Baum’a-Welch’a. A więc uczymy model HMM z komponentami GMM i przeprowadzamy wyrównanie transkrypcji z ramkami sygnału akustycznego na podstawie ścieżki Viterbi’ego. W tym celu można skorzystać z gotowych algorytmów Sphinx3 lub segmentacji Loquendo.

Alternatywą dla algorytmu Baum’a-Welch’a może być opracowanie wstępnej segmentacji za pomocą samoorganizujących się sieci Kohonena. W tym przypadku trzeba opracować odpowiednią metrykę dla wektora cech która umożliwi wyznaczenie odległości pomiędzy sąsiednimi ramkami i przypisanie im odpowiednich etykiem odpowiadającym poszczególnym stanom HMM. Ponieważ nie znalazłem w literaturze takie rozwiązania zostawiam sobie takie podejście do przetestowania na koniec w ramach eksperymentu.

## MLP jako estymator prawdopodobieństw emisji w HMM

Jak wspomniano wcześniej MLP wyuczony wg schematu 1-z-N za pomocą algorytmu wstecznej propagacji estymuje prawdopodobieństwa a posteriori na wyjściach ostatniej warstwy. A więc na i-tym wyjściu ostatniej warstwy sieci neuronowej otrzymujemy prawdopodobieństwo P(ci|o), gdzie ci – jest stanem z procesu HMM a o – wektorem cech podawanym na wejściu MLP. Jednak prawdopodobieństwo emisji obserwowalnego zdarzenia ukrytego procesu Markova to prawdopodobieństwo warunkowe P(o|ci). Stosując regułę Bayesa otrzymujemy:

Zatem dzieląc każde wyjście ostatniej warstwy perceptronu przez częstość względną występowania klasy w danych treningowych, otrzymujemy przeskalowane prawdopodobieństwo emisji, które możemy wykorzystać podczas rozpoznawania.

## Wyszukiwanie Viterbi’ego

Do rozpoznawania mowy wykorzystujemy standardowy algorytm Viterbi’ego [21] (szczegółowo opisany pod kątem ASR w [2]). Jeżeli przeskalujemy wyjścia sieci neuronowej (patrz punkt 3.9) to otrzymamy przeskalowane prawdopodobieństwo P(o|ci)/P(o). Ponieważ podczas rozpoznawania prawdopodobieństwo P(o) jest stałe dla każdej klasy (stanu HMM) możemy je pominąć i wykorzystać przeskalowane wyjścia perceptronu jako prawdopodobieństwa emisji obserwacji o dla stanu ci ukrytego procesu Markova.

# Szczegóły implementacyjne

## Parametry

* Częstotliwość próbkowania sygnału mowy – 8kHz pasmo telefoniczne
* Rozmiar okna 25ms przesuwanego co 10ms
* Podobnie jak w [8] lewy oraz prawy kontekst składa się z 3 ramek
* W zależności od redukcji błędu dobrać liczbę stanów odpowiadających jednostce STU – docelowo jeden stan dla jednostek stacjonarnych i 2 lub 1 dla jednostek przejściowych
* Pierwsza warstwa sieci nie będzie w pełni połączona tylko jak w [8] podzielona na bloki
* Podstawowy algorytm ekstrakcji cech to MFCC plus delta pierwszego i drugiego rzędu. Jeśli zostanie trochę czasu to również przeprowadzę testy dla PLP i kombinacji obu algorytmów.
* Jako słownik fonetyczny wykorzystam słownik alfabetu IPA z [20] plus zamiana afrykat które są w tym alfabecie modelowane jako dwa fonemy na pojedyncze fonemy.
* Korpus językowy opracuje na podstawie swojego głosu. Ze słownika lub jakiegoś ebooka wytypuje losowo ok. 3tys. odpowiednio zbalansowanych fonetycznie wyrazów (tak aby żaden fonem nie był liczniej reprezentowany w danych treningowych niż pozostałe) i nagram przypisując każdej próbce odpowiednią transkrypcje. Korpus zostanie podzielony na dane treningowe i dane testowe.
* Funkcja błędu wykorzystywana w algorytmie wstecznej propagacji to będzie albo błąd średnio kwadratowy albo entropia wzajemna (cross entropy).
* Liczbę perceptronów w warstwie ukrytej dobiorę eksperymentalnie.

## Technologie

### Spring Framework

Jako aplikację szkieletową zamierzam wykorzystać Spring Framework, który jest kontenerem inversion of control i dependency injection. Spring ułatwia i wymusza programowanie komponentowe, świetnie separuje abstrakcje od implementacji i pozwala pisać luźno powiązane elementy systemu. Dodatkowo posiada wiele gotowych komponentów.

### Spring RCP

Integruje Spring Framework z Java Swing. Ułatwia tworzenie konfigurowalnych i elastycznych aplikacji GUI. Implementuje wiele dobrych wzorców programowania obiektowego i tworzenia dobrych interfejsów graficznych. Główne zalety: architektura zorientowana na zdarzenia i wzorzec Command, walidacja, bindowanie danych, wsparcie dla zaawansowanych formatek, wielookienkowe aplikacje.

### Joone - Java Object Oriented Neural Engine

Gotowa implementacja mechanizmów do tworzenia, nauki testowania sieci neuronowej. Zaimplementowana w 100% w Javie. Implementuje sieci jednokierunkowe, rekurencyjne, typu time-delay, algorytm wstecznej propagacji, sieci Kohonenna itp.

### JCR Jack Rabbit

Repozytorium przechowywania danych wraz z meta danymi. Ułatwia przechowywanie, wyszukiwanie, indeksowanie danych hierarchicznych. Jest swego rodzaju typem obiektowej bazy danych. Będzie mi pomocne w przechowywaniu dokumentów i danych binarnych wraz z meta danymi.

### TestNG

Środowisko uruchomieniowe dla testów jednostkowych. Ponieważ projektowany system będzie powstawał przez kilka najbliższych miesięcy toteż aby udokumentować kontrakt metod i przetestować każdą metodę i wykryć na wczesnym etapie wszystkie błędy zamierzam na bieżąco pisać jak największą ilość testów jednostkowych.

### Maven2

Zarządza strukturą projektu oraz wszystkimi zależnościami, automatyzuje proces budowania aplikacji oraz uruchamiania testów jednostkowych.

# Do zrealizowania

## Wydzielone zadania do wykonania

### Szkielet Gui

Implementacja głównego panelu umożliwiające zarządzanie słownikiem, procesem uczenia i testowaniem systemu w czasie rzeczywistym. Będzie posiadać strukturę wielookienkową.

### Obsługa słownika

Import słownika [20]. Wydzielenie odrębnych komponentów enkapsulujących alfabet fonetyczny i format danych surowych.

### Obsługa mikrofonu

Interfejsy separujące konkretne mechanizmy odpowiedzialne za nagrywanie sygnału akustycznego z mikrofonu. Implementacja mechanizmu nagrywania w oparciu JMF.

### Budowa korpusu językowego

Algorytm typujący na podstawie wprowadzonego tekstu najbardziej reprezentatywne fonetycznie słowa. Osobne widoki odpowiedzialne za nagrywanie mowy i zapisujące wypowiedziane słowa wraz z transkrypcją i meta danymi w repozytorium JCR.

### Frontend (MFCC, PLP)

Przeniesienie algorytmów ekstrakcji cech ze Sphinx4, refaktoring odziedziczonego kodu, testy jednostkowe.

### Trening – Connectionist Viterbi training

Implementacja algorytmu opisanego w punkcie 3.7.

### Segmentacja początkowa

Implementacja któregoś sposobu wyznaczania segmentacji początkowej.

### Dekodowanie

Implementacja algorytmu Viterbi’ego.

### Bazowe struktury danych

Implementacja struktur danych dla HMM, stanów, fonemów, MLP, MFCC, PLP wraz interfejsami separującymi abstrakcje od implementacji.

### Wizualizacja

Jeśli zostanie czas to implementacja komponentów umożliwiających wizualizację procesu rozpoznawania, przekształcania sygnału akustycznego itp.

### Automatyzacja

Implementacja środowiska umożliwiającego automatyzację procesu tworzenia korpusu, nauki, dekodowania i testowanie różnych konfiguracji systemu w trybie wsadowym.

### Dokumentacja

Dokumentacja systemu, diagramy UML, brakujących części teoretycznych, analiza porównawcza różnych konfiguracji, wnioski.

## Planowany harmonogram

|  |  |
| --- | --- |
| koniec marca | 1. struktura projektu 2. szkielet gui |
| kwiecień | 1. obsługa słownika 2. mikrofon 3. korpus językowy |
| maj | 1. adaptacja wektora cech 2. segmentacja 3. trening |
| czerwiec | 1. dekodowanie 2. automatyzacja |
| lipiec | 1. dokumentacja |
| sierpień | 1. wizualizacja 2. dodatkowe elementy i pomysły |

# Literatura

|  |  |
| --- | --- |
| [1] | J.B. Hampshire, B.A. Pearlmutter “Equivalence Proofs for Multi-Layer Perceptron Classifiers and the Bayesian Discriminant Function” |
| [2] | D. Jurafsky, J.H. Martin “Speech and language processing” |
| [3] | L. E. Baum, T. Petrie, G. Soules, and N. Weiss "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains" |
| [4] | Sakoe, H. and Chiba, S., “Dynamic programming algorithm optimization for spoken word recognition” |
| [5] | C. S. Myers and L. R. Rabiner. “A comparative study of several dynamic time-warping algorithms for connected word recognition.” |
| [6] | Aldrich, John "R.A. Fisher and the making of maximum likelihood 1912-1922” |
| [7] | H. Bourlard, N. Morgan “Connectionist Speech Recognition A Hybrid Approach” |
| [8] | R. Gemello, D. Albesano, F. Mana, “Continuous Speech Recognition with Neural Networks and Stationary-Transitional Acoustic Units” |
| [9] | R. Gemello, D. Albesano, F. Mana, "Multi-source neural networks for speech recognition |
| [10] | R. Gemello, D. Albesano, F. Mana “CSELT Hybrid HMM/Neural Networks Technology for Continuos Speech Recognition” |
| [11] | M.A. Franzini, K.F. Lee and A. Waibel, "Connectionist Viterbi Training: A new hybrid method for continuous speech recognition" |
| [12] | S. Renals , N. Morgan, H. Bourlard, M. Cohen, H. Franco “Connectionist Probability Estimators in HMM Speech Recognition” |
| [13] | H. Hermansky, “Perceptual linear predictive analysis of speech” |
| [14] | H. Hermansky, N. Morgan, “RASTA Processing of Speech” |
| [15] | R. Gemello, D. Albesano, F. Mana, “Multi-source neural networks for speech recognition” |
| [16] | R. De Mori, R. Gemello, D. Albesano, F. Mana, “Ear-model derived features for automatic speech recognition” |
| [17] | R. Gemello, D. Albesano, F. Mana, P. Pegoraro, “Multi Source Neural Networks based on fixed and Multiple Resolution Analysis for Speech Recognition” |
| [18] | R. Gemello, D. Albesano, F. Mana, “Hybrid HMM-NN for speech recognition and prior class probabilities” |
| [19] | L. Fissore, F. Ravera, P. Laface, “Acoustic-Phonetic Modeling For Flexible Vocabulary Speech Recognition” |
| [20] | Otwarty słownik transkrypcji fonetycznej j. Polskiego V.11.2007 <http://www.elektronikjk.republika.pl/y11.html> |
| [21] | Andrew J. Viterbi. “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm” |
| [\*] | <http://www.cslu.ogi.edu/people/hosom/cs552/> |
| [\*\*] | <http://www.stanford.edu/class/cs224s/> |
|  |  |