Automatikus beszédfelismerés

Mérési Segédlet

Készítette: Lükő Bálint
Budapest, BME-TTT, 1998.

TARTALOMJEGYZÉK

1. Bevezetés

Mindennapi életünk során gyakran kerülünk kapcsolatba számítógépekkel és számítógép-vezérelt eszközökkel. Ezeket a készülékeket vezérelnünk kell, és azok visszajelzést adnak. A velük való kommunikáció módja meghatározza annak hatékonyságát, s ezért törekszünk arra, hogy azt minél könnyebbé, egyszerűbbé tegyük. Az emberi beszéd kíválóan alkalmas erre a célra, mert számunkra ez a legtermészetesebb közlési forma. Tehát a gépeket meg kell tanítanunk beszélni és a beszédet megérteni.

A mérés során egy teljes beszédfelismerő rendszert ismerhetnek meg. A programot mikrofonnal és hangszóróval felszerelt számítógépen futtatva párbeszédet folytathatunk a géppel: betaníthatjuk saját hangunkra, felvehetünk neveket és telefonszámokat a telefonkönyvbe, és tárcsázhatjuk azokat a név bemondásával. A rendszer rendeltetése emberi hanggal való tárcsázás. Mobiltelefonokra gondolva, ez a szolgáltatás hasznos lehet, amikor a felhasználó nem nézheti készüléke billentyűit, mert például autót vezet. Ilyenkor sokat segít, ha a gombok nyomkodása helyett elég a hívott fél telefonszámát, sőt csak a nevét kimondani, és a készülék automatikusan kapcsol.

Ahhoz, hogy a felismerés biztonságát javítani tudjuk, szükséges, hogy azonos körülmények között felismerési teszteket hajthassunk végre. A rendszerben különféle felismerési algoritmusokat könnyen ki lehet próbálni, és eredményüket össze lehet vetni. A program teljesen automatikusan tud teszteket végrehajtani, és az eredményeket naplózni.

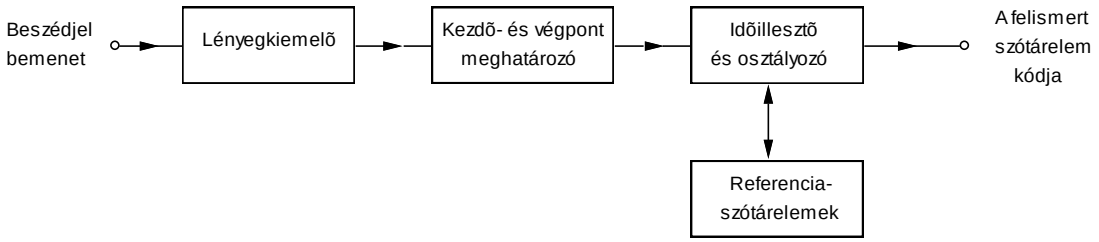
2. A beszédfelismerőkről általában

2.1 Alapvető beszédfelismerési módszerek

A beszédjel az információt részben akusztikus, részben nyelvi szinten hordozza, így a felismerés meglehetősen nehéz, hiszen csak az akusztikus szint figyelembe vétele nem elég. Ezért minden felismerő megpróbálja a beszéd különböző nyelvi jellemzőit egyértelműen meghatározni és a jellemzőket számokká alakítani. Ezekkel a számokkal utána könnyen elvégezhető az összehasonlítás, illetve a döntés.

2.1.1 Izolált szavas beszédfelismerők

Az izolált szavas beszédfelismerők rövid szünetekkel elválasztott szavakat vagy szócsoportokat tudnak feldolgozni. A szóközi szünetekkel a beszéd könnyen felbontható szavakra, hiszen a szókezdetek és a szóvégek meghatározhatók.



2.1. ábra *Izolált szavas beszédfelismerő blokkvázlata*

A szokásos beszédfelismerők elemeinek feladatai:

- **Lényegkiemelés:** Az időtartománybeli jelből megkísérli meghatározni a beszéd tartalmát hordozó mennyiségeket, és kiküszöbölni a felismerés szempontjából érdektelen információkat (zaj, fázis, torzítások). A digitalizált (beszéd)jelből egy diszkrét idejű, adott dimenziójú lényegvektor-sorozatot alkot. Néhány lehetséges eljárás: lineáris predikció, Fourier transzformáció, sávszűrők, kepsztrális analízis.
- **Kezdő- és végpont meghatározás:** A szünet és a beszéd elkülönítése, szétválasztása. Történhet a jel energiája, nullátmeneteinek száma és/vagy egyéb jellemzők alapján. Fontos, hogy ez a modul jól működjék, hiszen ezen múlik a későbbi feldolgozás sikere is.

- **Időillesztés:** A különböző beszédsebességekből adódó eltéréseket korrigálja az időtengely nyújtásával ill. zsugorításával (vetemítőfüggvény), valamint időnormálást végez (a beszéd lényegvektorának és a referenciaelemek lényegvektorának típusa és dimenziója meg kell egyezzen).
- **Osztályozás:** Kiválasztja referencia szótárelemek közül azt az elemet, amelyiknek lényegvektor-sorozata leginkább hasonlít a beszéd lényegvektor-sorozatához. A hasonlóság valamilyen távolságfüggvény (például az Euklidészi távolság) segítségével mérhető.

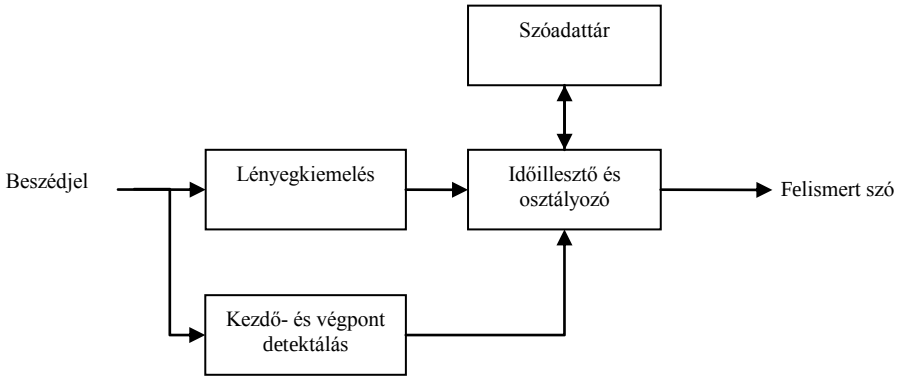
2.1.2 Folyamatos beszéd felismerése

Folyamatos beszéd felismerésére ma szinte kizárólag a rejtett Markov modellt (HMM) használják. Ebben a modellben a mondatokat és szavakat fonéma, félszótag vagy szó szinten állapotok sorozatával írják le. A bejövő lényegvektor-sorozatot, amelyet a lényegkiemelő modul szolgáltat, az ún. Viterbi algoritmussal kiértékelik, azaz meghatározzák, hogy az egyes modellek milyen valószínűek a megfigyelés (a bejövő vektorok) alapján. A modellt nyelvtannal is kiegészítik, amely a nyelvi szintű ellenőrzéshez szükséges és az alapegységek (szavak) megengedett összekapcsolásait (mondatait) adja meg, hasonlóan a kapcsolt szavas felismerőnél leírtakhoz.

Ez az eljárás felhasználható pl. szakértői rendszerekben, ahol kötött a téma, következésképpen korlátozottabb a nyelvtan és a szótár, de diktáló rendszerekben is, ahol sokkal kevesebb a megkötés a bemondott szöveget illetően. A HMM-módszer előnye a komolyabb felmerési feladatoknál a dinamikus idővetemítéshez (DTW) képest a jóval nagyobb hatékonyság, és a beszélő független felismerés lehetősége. A DTW előnye, hogy kisszótáras beszélőfüggő feladatoknál hatékonyabb és nyelvfüggetlen. További különbség a két módszer között, hogy a HMM-alapú beszédfelismerőket “gyárilag” tanítják nagy mennyiségű (több száz órányi) beszéddel, a DTW felismerőket a felhasználó tanítja a szótárelemek egyszeri bemondásával.

3. A *Vdial* program működése

A programnak kétféle üzemmódja van: egyszerű szóbetanítás és felismerés, valamint parancsállomány végrehajtás. Az első esetben akár file-ból, akár közvetlenül mikrofonnal betaníthatunk és felismertethetünk szavakat. Míg a parancsállományok futtatásának célja, hogy beszédfelismerési kísérleteket könnyen lehessen végrehajtani. Egy parancsállományban utasításokat adhatunk meg, amelyeket a program végrehajt. A program jegyzőkönyvet is vezet, amely a paraméter-beállításokat és a felismerési eredményeket tartalmazza. Kísérlet végrehajtása során a beszédfelismerő felismerési hibaarányát vizsgáljuk különböző paraméter-beállítások mellett. Ennek segítségével megtalálhatjuk a legjobb beszédfelismerési algoritmusokat és azok legmegfelelőbb paraméterezéseit.



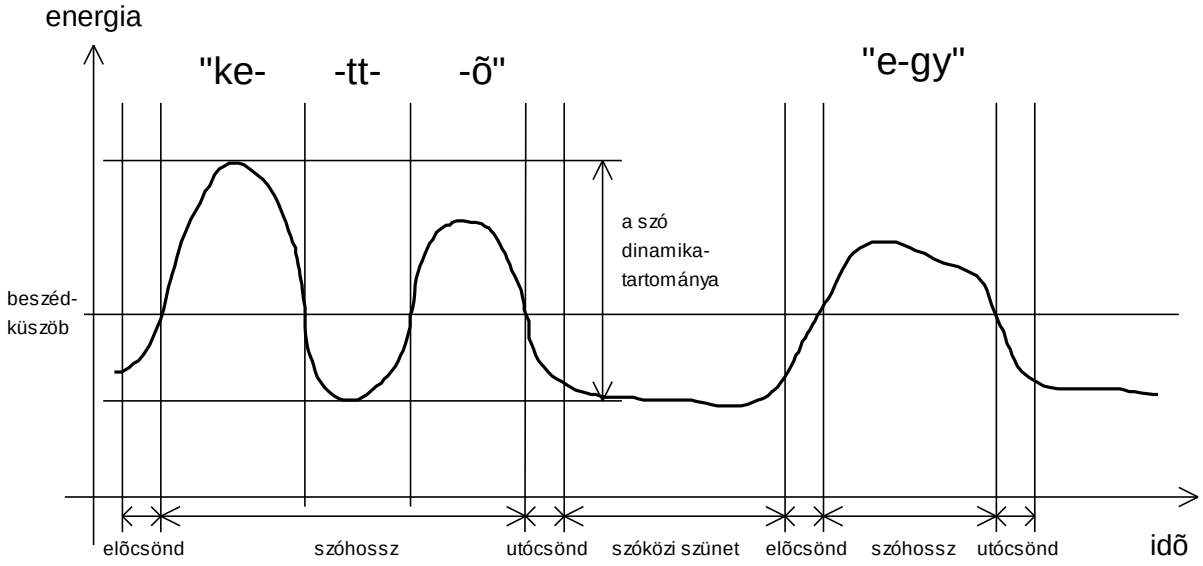
3.1. ábra *A rendszer blokkvázlata*

A következőkben ismertetem a rendszer egyes funkcionális elemeit.

3.1 Kezdő- és végpont detektálás

A beszéd kezdő- és végpont detektáló egység a bejövő jelből keresi meg azokat a szakaszokat, amelyek a bemondott szót, szavakat tartalmazzák. A beszédhang észlelése a jel energiája alapján történik: ha az egy bizonyos küszöbérték felett van, akkor azt beszédhangnak vesszük. A küszöbérték adaptív, mindenkori értéke az eddigi legkisebb energia szintje egy előre beállított, dB-ben megadott értékkel megnövelve *(ebből következik, hogy menet közben a mikrofont nem szabad ki-be kapcsolni!)*. Így a küszöbszint a mindenkori zajviszonyokhoz igazodik.

További megkötés a bejövő jel energiájával kapcsolatban az, hogy a küszöbértéket egy előre megadott időnél tovább kell, hogy meghaladja, különben a szókereső nem tekinti szónak az adott magas energiájú szakaszt. Ezzel a módszerrel a rövid, hirtelen zajokat, zörejeket szűrjük ki. Ennek a előfeltételnek a párját is megvan: túl hosszú (egy megadott időnél hosszabb) hangos szakaszt sem tekintünk szónak, nehogy valamilyen hosszú időtartamú zajforrás megzavarja a rendszert. Ezért ha túl hosszan tart már a nagy hangerejű rész, akkor úgy vesszük, hogy az eddigi jel is csak zaj volt, és a küszöbszintet a jelenlegi energiaszinthez képest állapítjuk meg.



3.2. ábra *A szókeresés néhány paramétere*

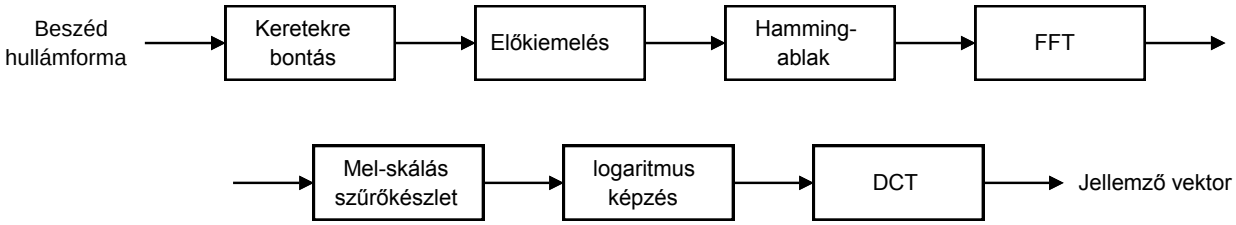
Még egy további fontos szempont, hogy a rövid szüneteket, csendet tartalmazó szavakat is egy szónak vegyük. Például a “kettő” szóban a hosszú “t” hang mintegy 200 ms csendet foglal magában. Ha nem ügyelnénk rá, akkor a szókereső a “kettő” szót két külön szónak venné.

Arra is gondolnunk kell, hogy egy szó nem ott kezdődik ill. végződik, ahol a jel energiája a küszöbszintet átlépi, ugyanis gyakran a mássalhangzók olyan halkak, hogy energiájuk nem éri el a küszöböt. A felismerés szempontjából azonban fontosak lehetnek ezek a kisebb energiájú részek is, ezért a szókereső egység a megtalált szót megtoldja elöl is és hátul is rövid jelszakaszokkal. Ez persze azzal jár, hogy szóközi szünetet, csendet is beleveszünk a szóba. Az időillesztő feladata, hogy ezeket a felismerésnél figyelmen kívül hagyja.

Végül még egy kritériumot figyelembe vesz a szókereső egység: elég nagy-e a bemondott szó dinamikája? Ha az túl kicsi, akkor valószínűleg csak valamilyen hosszantartó zaj, például a mikrofonba való szuszogás indította el a bemenet figyelését. Ez esetben a szókereső érvénytelennek tekinti a bemondást.

3.2 A lényegkiemelő egység

A lényegkiemelő egység feladata a beszédjelből kinyerni azon információkat, amelyek a bemondott szó felismeréséhez szükségesek. Törekszünk arra, hogy a különböző körülmények okozta zavaró hatásokat minél jobban kiszűrjük. Ezt megfelelő transzformáció megválasztásával elérhetjük.



3.3. ábra *A lényegkiemelő egység blokkvázlata*

3.2.1 Keretezés

A bejövő beszédjel telefonminőségnél kicsit jobb: 16 bites, 8 kHz mintavételi frekvenciájú. A jelet 32 ms-os keretekre bontjuk fel. Ez azért szükséges, mert a beszédjel folyamatosan változik, és ezen változásokat nyomon szeretnénk követni és a feldolgozás blokkokban történik. Túl nagy keretméret esetén a gyors változások kiátlagolódnak, túl kis keretméret esetén pedig a beszéd jellemzőit pontatlanul tudjuk csak meghatározni.

A keretek 50%-ban átlapoltak, azaz a következő keret kezdete az aktuális keret vége elé esik. Erre azért van szükség, hogy a feldolgozás során a beszédjellemzők változását jól nyomon tudjuk követni.

3.2.2 Előkiemelés

Az előkiemelés az alacsony frekvenciájú komponenseket elnyomja a jelben, míg a magasakat kiemeli. A használt előkiemelő egy elsőfokú FIR szűrő, amelynek átviteli függvénye:

$$W(z) = 1 - 0.95z^{-1}$$

Kiszámítási módja:

$$y[n] = x[n] - 0.95 \cdot x[n-1]$$

3.2.3 Hamming ablak

A diszkrét Fourier transzformáció (DFT) előtt ablakozni kell, mert nem periodikus jellel dolgozunk. Nem mindegy azonban, hogy az ablak milyen formájú. A legegyszerűbb ablak, a négyszögablak, azért nem jó, mert pl. egy tiszta szinusz jel esetén négyszög ablakot használva a DFT-vel előállított spektrum "szétkent" lesz, nem csak a szinusz jel frekvenciájának megfelelő helyen találunk vonalat, azaz nullától különböző értéket. A Hamming ablak azonban a spektrumot "élesíti", mert szűri a spektrumot. Az ablakfüggvénnyel való szorzás az időtartományban konvolúciónak felel meg a frekvenciatartományban, a véges konvolúció pedig azonos egy FIR szűrővel való szűréssel. Tehát az ablakozás a jel Fourier transzformáltját szűri. Ezen FIR szűrő együtthatóit az ablakfüggvény Fourier transzformáltja adja meg. Hamming ablak esetében a frekvenciatartománybeli ekvivalens szűrő egy felülátereszto szűrő.

A Hamming ablak függvénye:

$$h[n] = 0.54 - 0.46 \cos \frac{2\pi n}{N}$$

ahol n = 0 ... N-1, és N az ablak mérete.

3.2.4 Diszkrét Fourier transzformáció

A diszkrét Fourier transzformációval térünk át időtartományból frekvenciatartományba. Ez azért szükséges, mert a beszédhangok jellemzői, a formánsfrekvenciák csak a beszédjel spektrumán ismerhetők fel. Ezen kívül a bemenő jel számtalan torzulást szenved: véletlenszerű fázistolás a különböző frekvenciákon, additív zaj, torzítás (konvolúciós zaj), és ezeket frekvenciatartományban, illetve további transzformációk segítségével lehet csak kiküszöbölni.

A DFT-t gyors algoritmussal (FFT) érdemes számolni, ez ugyanis összehasonlíthatatlanul gyorsabb. A keletkezett komplex spektrum abszolút érték négyzetét tartjuk csak meg, a fázisinformáció a beszéd tartalma szempontjából lényegtelen, sőt csak

zavaró tényező. Mivel a teljesítmény- (abszolútérték-négyzet) spektrum szimmetrikus, ezért a felét elhagyjuk, a Nyquist tétel értelmében nem is hordoz a bemeneti valós jelről információt.

A DFT együtthatóinak meghatározása:

$$F_k = \sum_{i=0}^{N-1} x[i] e^{-2\pi j \cdot i \cdot k}$$

ahol x[i] a transzformálandó jel időfüggvénye, N a transzformáció mérete, míg F_k a keresett Fourier együtthatók (k = 0 ... N-1 ill. esetünkben k = 0 ... N/2-1).

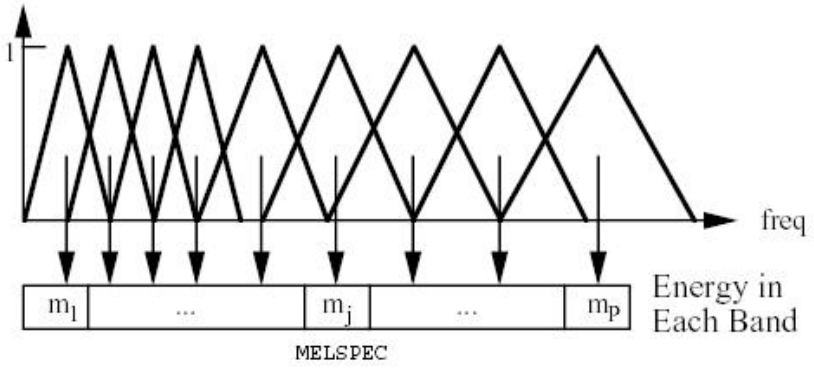
3.2.5 Mel-skála szerinti sávszűrőkészlet

Az ember hangérzékelése a frekvencia függvényében változik. Magasabb frekvenciákon szélesebb frekvenciaközű hangokat képes csak megkülönböztetni, mint alacsonyabb frekvenciákon. Ez a megkülönböztető képesség (frekvencia felbontás) 1000 Hz alatt közelítőleg lineárisan változik, míg e fölött logaritmikusan *(tehát 1000 Hz fölött a sávok szélessége a frekvencia függvényében exponencilisan növekszik)*. Ezt hívják mel-skálának. Mivel az ember hallása az emberi beszéd megértésére kíváló, ezért célszerű ezt utánozni. Ez a skála tulajdonképpen azt mutatja, hogy a tartalomra jellemző információ a felismerés szempontjából milyen sűrűséggel helyezkedik el a frekvenciatengely mentén.

A mel-skála képlete:

$$f_{mel} = 2595 \cdot \log_{10}\left(1 + \frac{f_{lin}}{700\,Hz}\right)$$

Az általam használt sávszűrők száma 40, és a teljes frekvenciatartományt (0 - 4 kHz) lefedik. Kiszámításuk a teljesítményspektrumból történik, összegzéssel.



3.4. ábra *A mel-skála szerinti sávszűrők (9 sávra)*

3.2.6 Logaritmusképzés és diszkrét koszinusz transzformáció

A jelfeldolgozás utolsó két lépése a kepsztrum meghatározására szolgál. A "hagyományos" kepsztrumot lineáris skálázású logaritmikus spektrumból (abszolút érték DFT logaritmusából) inverz DFT-vel számítják. Ezzel szemben az úgynevezett mel-kepsztrumot a fent leírt mel-skálájú szűrőkészlet logaritmikus kimenetéből számítják DFT-vel vagy diszkrét koszinusz transzformációval (DCT). Ez utóbbi transzformáció a képfeldolgozásban használatos, és fontos tulajdonsága, hogy a bemeneti jel fázisát megőrzi, és szemben a DFT-vel, csak valós értékeket szolgáltat. (A bemeneti jel itt nem a beszéd időfüggvénye, hanem a logaritmikus spektrum. Míg a beszéd időfüggvénye szinusz-komponenseinek a fázisa érdektelen számunkra, addig a logaritmikus spektrum szinusz-komponenseinek a fázisa fontos információt hordoz például a beszédformánsok helyzetére vonatkozólag.)

A DCT együtthatóinak meghatározása:

$$c_m = \sum_{i=0}^{M-1} f_i \cos\left(\frac{m(i-0.5)\pi}{M}\right)$$

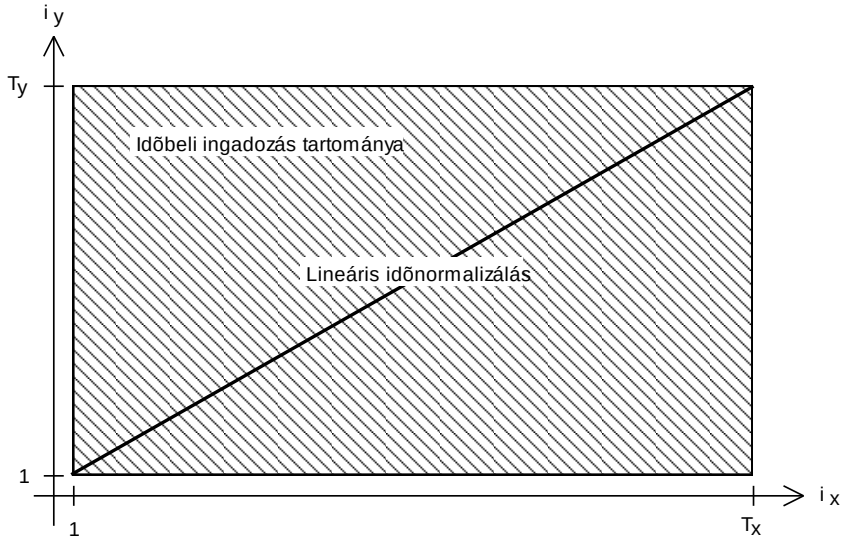
ahol M a sávszűrők száma. Nem számítjuk ki az összes DCT együtthatót, csak 12-t. Ezzel megkaptuk a beszéd felismeréséhez szükséges jellemző vektort. *(A DCT alkalmazásának valódi célja, hogy dekorrelálja a bemenetül kapott vektort. Így a DCT transzformált vektor a magasabb dimenziók elhagyásával is hatékonyan reprezentálhatja az eredeti vektor lényegi információtartalmát).*

3.3 Az időillesztő és osztályozó egység

Az időillesztő és osztályozó egység a bemondott szóhoz tartozó jellemzővektor-sorozat és az összes tárolt szó vektorsorozata közötti távolságot számolja ki, és a legkisebb távolságra lévőt adja meg, mint felismert szót *(jellemzővektor, lényegvektor: szinonimák)*. Az időillesztés a dinamikus idővetemítésnek (angol rövidítése DTW) nevezett eljárással történik [2].

A dinamikus idővetemítő eljárás bemenete két vektorsorozat, kimenete pedig a lehető legjobb idővetemítés mellett a két vektorsorozat megfelelő vektorai közötti távolságok összege. A feladat megoldásához először rajzoljunk fel egy koordináta rendszert, amelynek két tengelyén a két vektorsorozathoz tartozó (diszkrét) idő van feltüntetve, míg a rácspontokban a rácsponthoz tartozó két vektor távolsága van. Két vektor távolságán azok euklidészi távolságát értjük, azaz különbségük tagonkénti négyzetösszegét:

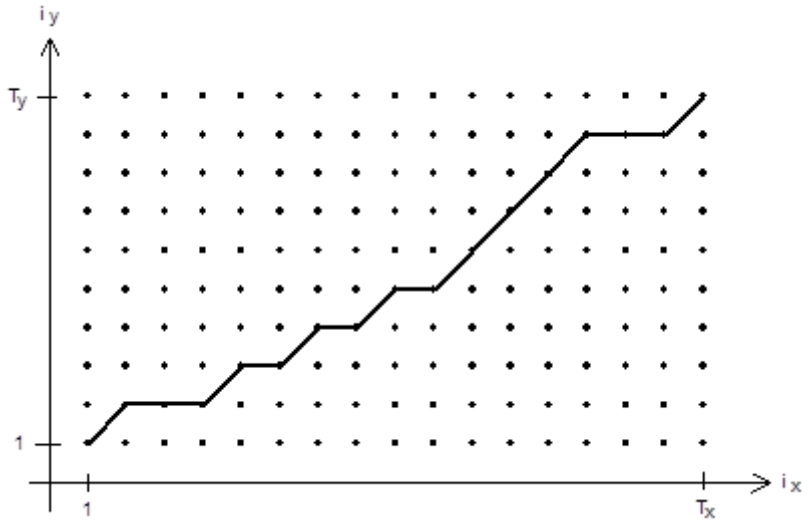
$$d(x,y) = \sum_{k=1}^{N} \left(x_k - y_k\right)^2$$



3.5. ábra *Lineáris időillesztés két különböző hosszúságú sorozatra*

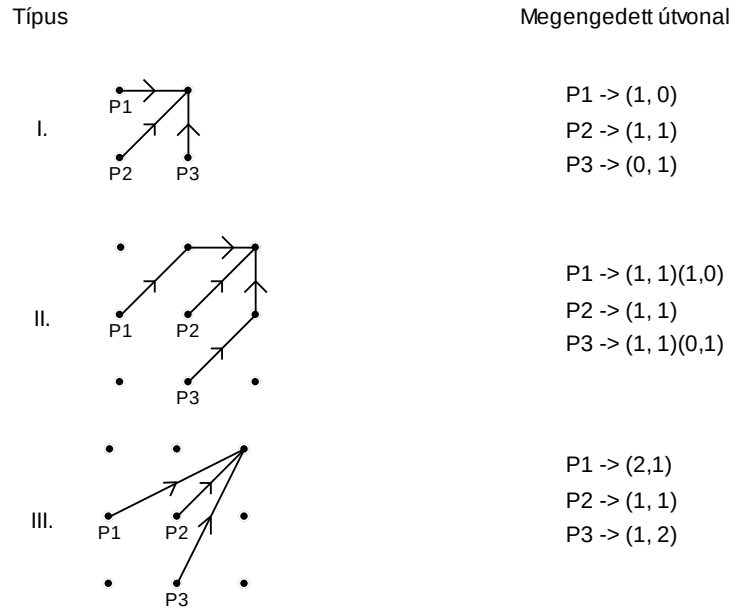
A 3.5. ábrán a vastag vonal jelzi azt az utat, amely mentén haladva egyenletesen nyújtjuk ill. zsugorítjuk a bemenő vektorsorozatot az összehasonlításhoz. Ez a *lineáris* idővetemítés. A vonalkázott területre kilépve az egyik szó egy részét egyenetlenül megnyújtanánk a másikhoz képest. Általános esetben ez utóbbi áll fenn, hiszen a magyar (és más nyelv) hangjai más-más arányban nyúlnak meg, ha ugyanazt a szót hosszabban

vagy rövidebben ejtjük. Például a ‘k’ hang zörej részének hossza nem sokat változik, szemben a magánhangzókkal. **Tehát a vetemítés útvonala nem az átló lesz** (lásd a 3.6. ábrát).



3.6. ábra *Időillesztés görbe út mentén*

A vetemítés útvonala nem lehet tetszés szerinti. Nem haladhat visszafelé. Ezen kívül az előre haladást is sokféleképpen korlátozhatjuk, attól függően, hogy mekkora ingadozást engedünk meg az illesztés vonalán. A 3.7. ábra néhány lehetőséget bemutat. A mi rendszerünkben a legelsőt használjuk, mert az tetszőleges hosszúságú nyújtást enged meg, és így a szóeleji és szóvégi csend részeket képes átlépni.



3.7. ábra *Néhány helyi folytonossági korlát és a hozzájuk tartozó útvonal megadása*

Az optimális útvonal definiálásához vezessünk be néhány jelölést! A vetemítő függvényeket jelöljük ϕ_x - szel és ϕ_y - nal, amelyek a két vektorsorozat i_x és i_y indexei és k között teremtenek kapcsolatot:

$$i_x = \phi_x(k), \qquad k = 1, 2, ..., T$$

és

$$i_y = \phi_y(k), \qquad k = 1, 2, ..., T$$

ahol T a két vektorsorozat "normalizált" időtartama. Globális minta-különbözőség mértéket definiálhatunk, amely a $\phi = (\phi_x, \phi_y)$ vetemítő függvénypár függvénye, és amely a teljes vektorsorozatra megadja a távolságot:

$$d_\phi(X,Y) = \sum_{k=1}^{T} d\left(\phi_X(k), \phi_Y(k)\right)$$

Ezek után X és Y különbözőségét így definiálhatjuk:

$$d(X,Y) := \min_{\phi} d_\phi(X,Y)$$

ahol Φ - nek teljesítenie kell bizonyos feltételeket:

- kezdőpont:	$\phi_x(1) = 1$	$\phi_y(1) = 1$
- végpont:	$\phi_x(T) = T_x$	$\phi_y(T) = T_y$
- monotonitás:	$\phi_x(k+1) \geq \phi_x(k)$	$\phi_y(k+1) \geq \phi_y(k)$
- helyi folytonosság:	$\phi_x(k+1) - \phi_x(k) \leq 1$	$\phi_y(k+1) - \phi_y(k) \leq 1.$

$d(X, Y)$ kiszámítása dinamikus programozással történik. A résztávolság az (1, 1) és (i_x, i_y) pontokat összekötő útvonalon:

$$D(i_x, i_y) := \min_{\phi_x, \phi_y, T'} \sum_{k=1}^{T'} d\left(\phi_X(k), \phi_Y(k)\right)$$

feltéve, ha teljesülnek, hogy

$$\phi_x(T') = i_x \qquad \text{és} \qquad \phi_y(T') = i_y.$$

Így a következő rekurzív formulát nyerjük:

$$D(i_x, i_y) := \min_{i_x', i_y'} \left[D(i_x', i_y') + \zeta\left((i_x', i_y'), (i_x, i_y)\right) \right] \qquad (3.1)$$

Általános helyi folytonossági korlátra: (*csak a téma iránt érdeklődőknek*)

$\zeta((i_x', i_y'), (i_x, i_y))$ a helyi távolság a (i_x', i_y') és a (i_x, i_y) pontok között:

$$\zeta\left(\left(i_x', i_y'\right), \left(i_x, i_y\right)\right) = \sum_{l=1}^{L_S} d\left(\phi_X\left(T'-l\right), \phi_Y\left(T'-l\right)\right)$$

ahol L_s a lépések száma (i_x', i_y') - ból (i_x, i_y) - ba ϕ_x és ϕ_y szerint. Megint csak teljesülnie kell, hogy

$$\phi_x (T' - L_s) = i_x' \qquad \text{és} \qquad \phi_y (T' - L_s) = i_y'.$$

Az I. típusú folytonossági korlátra:

A ζ növekményes távolságot csak a helyi folytonossági korlát által megengedett útvonalakon számítjuk ki, hogy a dinamikus programozási algoritmus minél hatékonyabb legyen. Más szóval a (i_x', i_y') tartománya az (3.1)-es kifejezés minimumkeresésében azokra a pontokra van korlátozva, amelyek megengedett kezdőpontok a helyi folytonossági korlátok halmazában. Az általunk alkalmazott helyi folytonossági korlátra:

$$D (i_x, i_y) = \min \{ \; D (i_x, - 1, i_y) + d(i_x, i_y),$$
$$D (i_x, - 1, i_y - 1) + d(i_x, i_y),$$
$$D (i_x, i_y - 1) + d(i_x, i_y) \; \}.$$

A teljes algoritmus tehát az alábbi lépésekből áll:

1. Inicializálás

$D_A (1, 1) = d (1, 1).$

2. Rekurzió

Minden i_x-re és i_y-ra, amelyekre teljesül, hogy $1 \le i_x \le T_x$ ill. $1 \le i_y \le T_y$, kiszámolandó

$D_A (i_x, i_y) = min \{D_A (i_x,-1, i_y) + d(i_x, i_y), D_A (i_x,-1, i_y-1) + d(i_x, i_y), D_A (i_x, i_y-1) + d(i_x, i_y)\}$

3. Befejezés

$d (X, Y) = D_A (T_x, T_y).$

Látható, hogy minden oszlop és sor csak az azt megelőzőtől függ, és a végeredmény az utolsó sor utolsó oszlopában van. Ezt úgy használhatjuk ki, hogy nem tároljuk az egész táblázatot a memóriában, hanem csak egy oszlopát, vagy sorát, és mindíg azt írjuk felül az új adatokkal. Ezzel jelentős memóriát spórolhatunk meg.

3.4 A szóadattár egység

A szóadattár egység a memóriában tárolja a felismerendő szavak jellemzővektor sorozatait a felismeréshez. Betanításkor eltárolja az új bemondáshoz tartozó vektorsorozatot, és címkével látja el. A címke parancsszó esetén (lásd később) maga a szó, míg név esetén az előfizető telefonszáma.

Háttértárolóra is el tudja menteni az adatokat, és onnan vissza tudja tölteni. Erre azért van szükség, mert így sok felhasználó használhatja a rendszert felváltva.

3.5 A parancsállomány

A parancsállomány egysoros parancsokat, utasításokat tartalmaz, amelyeket egy értelmező végrehajt. Célja, hogy felismerési kísérleteket könnyen, automatikusan hajthassunk végre. Egy példa parancsállományt láthatunk a függelék 1. részében. A parancsállomány végrehajtása során megmaradó kimenetként jegyzőkönyv-állomány keletkezik, amelyben a betanítás lépéseit, a paraméter-beállításokat és felismerés eredményeit dokumentálhatjuk megfelelő utasítások használatával.

Minden utasításnak új sorban kell kezdődnie. Az üres sorokat, valamint a fölösleges szóköz és tabulátor karaktereket az értelmező figyelmen kívül hagyja. Az utasítások és használatuk a 3.1. táblázatban vannak leírva.

Utasítás	Paramétere	Mit csinál
Train	WAVE állományok nevei szóközzel elválasztva	Az állományokat sorban beolvassa, a bennük tárolt szavakat megkeresi, elvégzi rajtuk a lényegkiemelést, és az így kapott jellemzővektor-sorozatokat eltárolja a szóadattárban
TrainFromMic	nincs	Ugyanaz, mint az előbbi, csak nem állományból dolgozik, hanem mikrofonba kell bemondani a betanítandó szót.
Test	WAVE állományok nevei szóközzel elválasztva	Az állományokat sorban beolvassa, a bennük tárolt szavakat felismeri, azaz összehasonlítja őket a szóadattárban lévőkkel és a legközelebb állókat adja meg.
Play	egy WAVE állomány neve	Lejátsza a megadott állományt. Szóbeli üzenetek közlésére való.
Rem	tetszőleges szöveg	Ez után a jel után megjegyzéseket írhatunk, amiket a parancsértelmező figyelmen kívül hagy.
Stop	nincs	A parancsállomány végrehajtását abbahagyja.
Call	eljárásnév	Meghív egy eljárást.
Proc	eljárásnév	Egy eljárás kezdetét adja meg.
EndProc	nincs	Egy eljárás végét adja meg.
Echo	tetszőleges szöveg	A megadott szöveget a parancsértelmező a jegyzőkönyvbe beírja. Ide például a kísérletek körülményeit írhatjuk.
ForgetTemplates	nincs	A memórában tárolt szóadatokat kitörli. Ez általában új kísérlet-szakaszok előtt szükséges.
ClearStatistics	nincs	Az eddig felgyülemlett statisztikai adatokat törli.
ShowStatistics	nincs	Beírja az eddigi felismerési statisztikát a jegyzőkönyvbe.
Set Path	elérési útvonal	A WAVE állományok elérési útvonalát adhatjuk meg. Ha nem adunk meg útvonalat, akkor törli a jelenlegit.
Set VectorType	**FilterBank** vagy **MelCep**	A betanításhoz és felismeréshez használt jellemző-vektor fajtáját állíthatjuk be sávszűrőkre vagy mel-kepsztrumra.
Set FilterBankSize	egy egész szám	A sávszűrők számát állíthatjuk be. Amennyiben a beállított jellemzővektor fajtája a sávszűrőkészlet, akkor ez a szám egyben a jellemzővektor dimenzióját is adja. Ha a beállított jellemzővektor a mel-kepsztrum, akkor ez a szám a mel-kepsztrum bemenő vektorának a dimenziója.
Set MelCepSize	egy egész szám	A mel-kepsztrum fokszámát adhatjuk meg vele. Ha a beállított jellemzővektor nem a mel-kepsztrum, akkor ez a beállítás érdektelen.

3.1. táblázat *A parancsállomány utasításkészlete*

4. A *Vdial* program használata

4.1 Menüpontok

4.1.1 A **Templates** menü

Az ebben található menüpontokkal a szóadattár tartalmát menthetjük lemezre, tölthetjük fel lemezről és törölhetjük.

4.1.2 A **Run** menü

- Az **Analyze from mic** menüpontot kiválasztva a program a mikrofonból érkező beszédjelen lényegkiemelést végez, valamint a szavak határát próbálja megtalálni.
- Az **Analyze file...** menüponttal a program egy hang állományt (WAVE típusút) dolgoz fel hasonlóan az előbbi menüponthoz.
- **Train from mic** esetén minden bemondott szót eltárol a szótárába az általunk megadott néven. Többször is szerepelhet ugyanaz a szó a szótárban.
- **Train file...** esetén az általunk kiválasztott hangállományból dolgozik. A hangállománynak kell legyen leíró állománya! (lásd később)
- **Recognize from mic** a mikrofonba mondott szavakat a szótárban található összes szóval összehasonlítja, és a legközelebbit adja meg.
- **Recognize file...** hangállományból való szófelismerés. Ha van leírója a hangállománynak, akkor a leíróban található szavakat sorrendben összehasonlítja a felismert szavakkal, és jelzi a felismerés sikerességét. Ez felismerési kísérletek futtatása esetén lehetővé tesz automatikus statisztika készítést.
- **Run command file...** parancsállomány futtatása.

4.1.3 Az **Options** menü

- **Step by step** ha ez a menüpont aktív, akkor a program a szóköz billentyű nyomvatartása alatt számol csak, a szóközt elengedve pedig várakozik (lépésenkénti üzemmód).
- **Word by word** ha ez a menüpont aktív, akkor a program a szóköz billentyű megnyomása után egy szót végigszámol, majd várakozó állapotba kerül a szóköz billentyű következő lenyomásáig (szavankénti üzemmód).

- **Do next frame** ez a menüpont a szóköz billentyű lenyomásával egyenértékű. Lépésenkénti üzemmódban a következő lépést (számolási műveletet) számolja ki.
- **Do next word** ez a menüpont is egyenértékű a szóköz billentyű lenyomásával. Szavankénti üzemmódban a következő szót számolja ki.
- **Pause** ez a menüpont szintén azonos a szóköz billentyű lenyomásával. Ha se nem vagyunk lépésenkénti üzemmódban, se nem szavankénti üzemmódban (ez az alapállapot), akkor ezzel a számolást felfüggeszthetjük ill. tovább engedhetjük.
- **Stop** az éppen futó művelet megszakítását eredményezi. Az Escape billentyűvel ugyanezt érjük el.
- **Playback** ha ez a menüpont aktív, akkor a program minden feldolgozott szó után az adott szót kimondja, azaz lejátsza a bemondott vagy a hangállományból beolvasott szót.
- **Isolated word recognition, Connected word recognition, Continuous recognition** izolált szavas, kapcsolt szavas, folyamatos szófelismerés. Az utóbbi kettő kisérleti jellegű egyelőre.

4.1.4 A **Settings** menü

- **Find word settings** a szókeresés paramétereit itt állíthatjuk (előcsönd, utócsönd, min. szóhossz, max. szóhossz, a szavak közötti szünet min. értéke, beszédküszöb, a szó min. dinamikatartománya).
- **Signal processing settings** a mintavételi frekvenciát (ha mikrofonból dolgozunk), a lényegkiemelés paramétereit (előkiemelés, Hamming-ablak, keretméret, keretkövetési távolság, sávszűrők száma, kepsztrum együtthatók száma), a jellemző vektor típusát (sávszűrő/mel-kepsztrum) és az additív zajt (WAVE állomány neve, szorzó konstans) állíthatjuk itt.
- **Plot settings** a megjelenítendő jeleket lehet kíválasztani (hullámforma, energia, FFT spektrum, sávszűrők, mel-kepsztrum), illetve a megjelenés színeit (fekete-fehér színskála, minden rajz fekete-fehér).

4.2 A hangállományt leíró állomány

A hangállománnyal azonos könyvtárban, ugyanazon a néven, de **.TXT** kiterjesztéssel ellátott szöveges állomány, amelyben szóközökkel ill. soremelésekkel elválasztva a hangállományban felvett szavak vannak sorrendben megadva.

5. Javasolt irodalom

[1]: Gordos Géza - Takács György:
Digitális beszédfeldolgozás
Műszaki Könyvkiadó, Budapest, 1983

[2]: Biing-Hwang Juang and Lawrence Rabiner:
Fundementals of Speech Recognition
PTR Prentice-Hall Inc., 1993

6. Függelék

6.1 Egy parancsállomány és futási eredménye

TEST1.CMD:

```
ClearTemplates
Set VectorType = FilterBank
Set FilterBankSize = 8
Call Test12
Set FilterBankSize = 12
Call Test12
Set FilterBankSize = 20
Call Test12
Set FilterBankSize = 30
Call Test12

Set VectorType = MelCep
Set MelCepSize = 8
Set FilterBankSize = 8
Call Test12
Set FilterBankSize = 12
Call Test12
Set FilterBankSize = 20
Call Test12
Set FilterBankSize = 30
Call Test12

Set MelCepSize = 12
Set FilterBankSize = 8
Call Test12
Set FilterBankSize = 12
Call Test12
Set FilterBankSize = 20
Call Test12
Set FilterBankSize = 30
Call Test12

Stop


Proc Test12

Call Test1
Call Test2

EndProc


Proc Test1

Set Path = WAVES\SZAMOK

ClearStatistics
Echo Train files: lb1 - lb4, test files: lb1 - lb4
ClearTemplates
Train lb1
Test lb2 lb3 lb4
ClearTemplates
Train lb2
Test lb1 lb3 lb4
ClearTemplates
Train lb3
Test lb1 lb2 lb4
ClearTemplates
Train lb4
Test lb1 lb2 lb3
ShowStatistics

ClearStatistics
Echo Train files: lb5 - lb8, test files: lb5 - lb8
ClearTemplates
Train lb5
Test lb6 lb7 lb8
ClearTemplates
Train lb6
Test lb5 lb7 lb8
ClearTemplates
Train lb7
Test lb5 lb6 lb8
ClearTemplates
Train lb8
Test lb5 lb6 lb7
ShowStatistics

ClearStatistics
Echo Train files: lb9 - lb12, test files: lb9 - lb12
ClearTemplates
Train lb9
Test lb10 lb11 lb12
ClearTemplates
Train lb10
Test lb9 lb11 lb12
ClearTemplates
Train lb11
Test lb9 lb10 lb12
ClearTemplates
Train lb12
Test lb9 lb10 lb11
ShowStatistics

Echo
ClearTemplates

EndProc


Proc Test2

Set Path = WAVES\SZAMOK

ClearStatistics
Echo Train files: lb1 - lb4, test files: lb1 - lb12
ClearTemplates
Train lb1
Test lb2 lb3 lb4 lb5 lb6 lb7 lb8 lb9 lb10 lb11 lb12
ClearTemplates
Train lb2
Test lb1 lb3 lb4 lb5 lb6 lb7 lb8 lb9 lb10 lb11 lb12
ClearTemplates
Train lb3
Test lb1 lb2 lb4 lb5 lb6 lb7 lb8 lb9 lb10 lb11 lb12
ClearTemplates
Train lb4
Test lb1 lb2 lb3 lb5 lb6 lb7 lb8 lb9 lb10 lb11 lb12
ShowStatistics

ClearStatistics
Echo Train files: lb5 - lb8, test files: lb1 - lb12
ClearTemplates
Train lb5
Test lb1 lb2 lb3 lb4 lb6 lb7 lb8 lb9 lb10 lb11 lb12
ClearTemplates
Train lb6
Test lb1 lb2 lb3 lb4 lb5 lb7 lb8 lb9 lb10 lb11 lb12
ClearTemplates
Train lb7
Test lb1 lb2 lb3 lb4 lb5 lb6 lb8 lb9 lb10 lb11 lb12
ClearTemplates
Train lb8
Test lb1 lb2 lb3 lb4 lb5 lb6 lb7 lb9 lb10 lb11 lb12
ShowStatistics

ClearStatistics
Echo Train files: lb9 - lb12, test files: lb1 - lb12
ClearTemplates
Train lb9
Test lb1 lb2 lb3 lb4 lb5 lb6 lb7 lb8 lb10 lb11 lb12
ClearTemplates
Train lb10
Test lb1 lb2 lb3 lb4 lb5 lb6 lb7 lb8 lb9 lb11 lb12
ClearTemplates
Train lb11
Test lb1 lb2 lb3 lb4 lb5 lb6 lb7 lb8 lb9 lb10 lb12
ClearTemplates
Train lb12
Test lb1 lb2 lb3 lb4 lb5 lb6 lb7 lb8 lb9 lb10 lb11
ShowStatistics

Echo
ClearTemplates

EndProc
```

TEST1.LOG:

```
Project OVSR - CMD Log file
VectorType = FilterBank
FilterBankSize = 8
Train files: lb1 - lb4, test files: lb1 - lb4
Word error rate: 2% (3 of 120)
Train files: lb5 - lb8, test files: lb5 - lb8
Word error rate: 5% (6 of 120)
Train files: lb9 - lb12, test files: lb9 - lb12
Word error rate: 7% (9 of 120)

Train files: lb1 - lb4, test files: lb1 - lb12
Word error rate: 12% (56 of 440)
Train files: lb5 - lb8, test files: lb1 - lb12
Word error rate: 11% (51 of 440)
Train files: lb9 - lb12, test files: lb1 - lb12
Word error rate: 14% (65 of 440)

FilterBankSize = 12
Train files: lb1 - lb4, test files: lb1 - lb4
Word error rate: 2% (3 of 120)
Train files: lb5 - lb8, test files: lb5 - lb8
Word error rate: 4% (5 of 120)
Train files: lb9 - lb12, test files: lb9 - lb12
Word error rate: 7% (9 of 120)

Train files: lb1 - lb4, test files: lb1 - lb12
Word error rate: 12% (57 of 440)
Train files: lb5 - lb8, test files: lb1 - lb12
Word error rate: 9% (41 of 440)
Train files: lb9 - lb12, test files: lb1 - lb12
Word error rate: 14% (63 of 440)

FilterBankSize = 20
Train files: lb1 - lb4, test files: lb1 - lb4
Word error rate: 0% (1 of 120)
Train files: lb5 - lb8, test files: lb5 - lb8
Word error rate: 5% (6 of 120)
Train files: lb9 - lb12, test files: lb9 - lb12
Word error rate: 6% (8 of 120)

Train files: lb1 - lb4, test files: lb1 - lb12
Word error rate: 10% (47 of 440)
Train files: lb5 - lb8, test files: lb1 - lb12
Word error rate: 8% (36 of 440)
Train files: lb9 - lb12, test files: lb1 - lb12
Word error rate: 12% (57 of 440)

FilterBankSize = 30
Train files: lb1 - lb4, test files: lb1 - lb4
Word error rate: 0% (1 of 120)
Train files: lb5 - lb8, test files: lb5 - lb8
Word error rate: 4% (5 of 120)
Train files: lb9 - lb12, test files: lb9 - lb12
Word error rate: 5% (7 of 120)

Train files: lb1 - lb4, test files: lb1 - lb12
Word error rate: 9% (42 of 440)
Train files: lb5 - lb8, test files: lb1 - lb12
Word error rate: 8% (37 of 440)
Train files: lb9 - lb12, test files: lb1 - lb12
Word error rate: 12% (55 of 440)

VectorType = MelCep
MelCepSize = 8
FilterBankSize = 8
Train files: lb1 - lb4, test files: lb1 - lb4
Word error rate: 0% (0 of 120)
Train files: lb5 - lb8, test files: lb5 - lb8
Word error rate: 1% (2 of 120)
Train files: lb9 - lb12, test files: lb9 - lb12
Word error rate: 2% (3 of 120)

Train files: lb1 - lb4, test files: lb1 - lb12
Word error rate: 5% (23 of 440)
Train files: lb5 - lb8, test files: lb1 - lb12
Word error rate: 2% (12 of 440)
Train files: lb9 - lb12, test files: lb1 - lb12
Word error rate: 4% (21 of 440)

FilterBankSize = 12
Train files: lb1 - lb4, test files: lb1 - lb4
Word error rate: 0% (1 of 120)
Train files: lb5 - lb8, test files: lb5 - lb8
Word error rate: 1% (2 of 120)
Train files: lb9 - lb12, test files: lb9 - lb12
Word error rate: 3% (4 of 120)

Train files: lb1 - lb4, test files: lb1 - lb12
Word error rate: 8% (37 of 440)
Train files: lb5 - lb8, test files: lb1 - lb12
Word error rate: 2% (11 of 440)
Train files: lb9 - lb12, test files: lb1 - lb12
Word error rate: 6% (28 of 440)

FilterBankSize = 20
Train files: lb1 - lb4, test files: lb1 - lb4
Word error rate: 0% (0 of 120)
Train files: lb5 - lb8, test files: lb5 - lb8
Word error rate: 1% (2 of 120)
Train files: lb9 - lb12, test files: lb9 - lb12
Word error rate: 3% (4 of 120)

Train files: lb1 - lb4, test files: lb1 - lb12
Word error rate: 9% (41 of 440)
Train files: lb5 - lb8, test files: lb1 - lb12
Word error rate: 3% (14 of 440)
Train files: lb9 - lb12, test files: lb1 - lb12
Word error rate: 8% (36 of 440)

FilterBankSize = 30
Train files: lb1 - lb4, test files: lb1 - lb4
Word error rate: 0% (0 of 120)
Train files: lb5 - lb8, test files: lb5 - lb8
Word error rate: 1% (2 of 120)
Train files: lb9 - lb12, test files: lb9 - lb12
Word error rate: 3% (4 of 120)

Train files: lb1 - lb4, test files: lb1 - lb12
Word error rate: 9% (40 of 440)
Train files: lb5 - lb8, test files: lb1 - lb12
Word error rate: 3% (14 of 440)
Train files: lb9 - lb12, test files: lb1 - lb12
Word error rate: 7% (33 of 440)

MelCepSize = 12
FilterBankSize = 8
Train files: lb1 - lb4, test files: lb1 - lb4
Word error rate: 0% (0 of 120)
Train files: lb5 - lb8, test files: lb5 - lb8
Word error rate: 1% (2 of 120)
Train files: lb9 - lb12, test files: lb9 - lb12
Word error rate: 3% (4 of 120)

Train files: lb1 - lb4, test files: lb1 - lb12
Word error rate: 4% (21 of 440)
Train files: lb5 - lb8, test files: lb1 - lb12
Word error rate: 3% (14 of 440)
Train files: lb9 - lb12, test files: lb1 - lb12
Word error rate: 4% (19 of 440)

FilterBankSize = 12
Train files: lb1 - lb4, test files: lb1 - lb4
Word error rate: 0% (1 of 120)
Train files: lb5 - lb8, test files: lb5 - lb8
Word error rate: 1% (2 of 120)
Train files: lb9 - lb12, test files: lb9 - lb12
Word error rate: 3% (4 of 120)

Train files: lb1 - lb4, test files: lb1 - lb12
Word error rate: 8% (38 of 440)
Train files: lb5 - lb8, test files: lb1 - lb12
Word error rate: 3% (15 of 440)
Train files: lb9 - lb12, test files: lb1 - lb12
Word error rate: 5% (24 of 440)

FilterBankSize = 20
Train files: lb1 - lb4, test files: lb1 - lb4
Word error rate: 0% (0 of 120)
Train files: lb5 - lb8, test files: lb5 - lb8
Word error rate: 1% (2 of 120)
Train files: lb9 - lb12, test files: lb9 - lb12
Word error rate: 4% (5 of 120)

Train files: lb1 - lb4, test files: lb1 - lb12
Word error rate: 9% (43 of 440)
Train files: lb5 - lb8, test files: lb1 - lb12
Word error rate: 3% (16 of 440)
Train files: lb9 - lb12, test files: lb1 - lb12
Word error rate: 6% (29 of 440)

FilterBankSize = 30
Train files: lb1 - lb4, test files: lb1 - lb4
Word error rate: 0% (0 of 120)
Train files: lb5 - lb8, test files: lb5 - lb8
Word error rate: 1% (2 of 120)
Train files: lb9 - lb12, test files: lb9 - lb12
Word error rate: 4% (5 of 120)

Train files: lb1 - lb4, test files: lb1 - lb12
Word error rate: 8% (37 of 440)
Train files: lb5 - lb8, test files: lb1 - lb12
Word error rate: 3% (16 of 440)
Train files: lb9 - lb12, test files: lb1 - lb12
Word error rate: 6% (28 of 440)
```