Szegedi Tudományegyetem
Mesterséges Intelligencia Kutatócsoport

Sebesség és pontosság javítása automatikus beszédfelismerés során

A PhD-értekezés tézisei

Gosztolya Gábor

Témavezetők:

Dr. Csirik János
Dr. Kocsor András

Szeged
2010

Bevezetés

A beszédfelismerési problémában egy adott bemondáshoz kell meghatározni a helyes szót vagy szósorozatot. Az erre szolgáló rendszerek és módszerek fejlesztése során az alkalmazott megközelítésektől, modellektől függetlenül mindig ügyelni kellett két nagyon fontos tényezőre: a pontosságra és a sebességre. Egy beszédfelismerő rendszer pontossága nyilván létfontosságú, elvégre a beszédfelismerés lényege a bemondás pontos átiratának meghatározása. Viszont a sebességet is kiemelten kell kezelni, mert ezt az átiratot valós időben szeretnénk megkapni; emellett persze az erőforrások pocsékolását is el kell kerülni, így a sebesség fontos tényező még a valós idejű felismerés elérése után is.

A sebesség és a pontosság fontossága miatt ezt a dolgozatot ezen két fogalom köré szerveztem; a dolgozat és az eredmények is erre a két részre lettek bontva. Az **első részben** a beszédfelismerési folyamat, elsősorban a keresési eljárás **felgyorsítására** koncentrálunk (mivel a keresés viszi el a futási idő nagy részét). A keresési eljárásban rengeteg potenciális szósorozatot (és azok prefixeit, a *hipotéziseket*) illesztünk a bemondásra. A keresés felgyorsításának egyik általános módja a *keresési módszer által megvizsgált* hipotézisek számának csökkentése. E célból a multi-stack decoding keresési eljárás [1] működésére javasoltunk módosításokat, melyek az egyes ponton tárolt hipotézisek számát határozzák meg adaptívan (az eredeti konstans helyett).

A keresési eljárás gyorsítására egy másik lehetőség a *lehetséges* hipotézisek számának csökkentése további kritériumok felállításával; így keresés közben kevesebb lehetőséget kell figyelembe venni. Erre két technikát vezettünk be: az első egy *többlépéses keresési módszer* [27], melyben több keresési lépést végzünk. A lépések egyre alaposabb vizsgálatokat végeznek, de mindig csak az előző lépések által valószínűbbnek talált hipotéziseket vizsgálják, így a kevéssé valószínű lehetőségeket gyorsan ki lehet zárni. Az ebben a dolgozatban bemutatott többlépéses keresési eljárás a fonémaosztályozó hibái alapján létrehozott fonémacsoportokra épül. A keresés felgyorsításának másik módja a lehetséges fonémahatárok korlátozása volt (*szegmentálás*), amire három új módszer lett bevezetve.

A dolgozat **második részében** olyan területekkel foglalkozunk, amelyekkel **javítható a beszédfelismerés pontossága**. Az első terület a valószínűségaggregáció folyamatának alapos vizsgálata, amely két szinten folyik: szegmensekre számolt fonémavalószínűségek kiszámítása és ezekből a szegmenszintű értékekből egész hipotézisek valószínűségének meghatározása. Az itt használt eljárások általában, a függetlenségi hipotézisből levezetve, szorzást alkalmaznak; egy jobban illeszkedő modellt keresve számos hasonló operátort próbáltunk ki, és egy gyakorlati felhasználásra koncentráló új módszert is bemutatunk a *fuzzy háromszögnormák* modellezésére. A pontosság javítására irányuló másik vizsgált probléma az *antifonéma-modellezés* területe: a beszédfelismerés bizonyos megközelítései során fontos megkülönböztetni a tényleges fonémákat a bemondásokban előforduló egyéb hangdaraboktól (pl. több egymás utáni fonéma, fonémarészletek). Erre számos mód kínálkozik; a standard eljárások mellett alkalmaztam a feladatra egy általános ellenpélda-generáló eljárást is. Az alkalmazásához számos módosítás bevezetésére is szükség volt, végül azonban ez a módszer vezetett a legjobb eredményekhez.

Az irodalom nagy része a beszédfelismerési területet keretalapú megközelítésben tárgyalja, jórészt a rejtett Markov-modellre épülve. Ez a dolgozat azonban mind keret-, mind szegmensalapú kísérleteket tartalmaz, melyeket egy általános megközelítés keretében tárgyalunk [29]. A kísérletek számos, a munka során elérhetővé váló adatbázison lettek elvégezve, melyek részletes adatai a diszszertációban találhatóak.

1. A beszédfelismerés keresési folyamatának gyorsítása

A beszédfelismerési problémában adott egy bemondás egy $A = a_1 a_2 \dots a_t$ megfigyeléssorozatként, és a lehetséges fonémasorozatok (szavak vagy szósorozatok, a továbbiakban röviden szavak) egy W halmaza. A feladat megtalálni azt a szót, amely optimálisan illeszkedik a bemondásra; azaz azt a $\hat{w} \in W$ szót, melyre

$$\hat{w} = \arg\max_{w \in W} P(w|A). \tag{1}$$

A beszédfelismerés *diszkriminatív megközelítése* az (1). egyenletet használja [8], mi azonban először alkalmazzuk a Bayes-tételt:

$$\hat{w} = \arg\max_{w \in W} \frac{P(A|w)P(w)}{P(A)}. \tag{2}$$

Ezután, felhasználva, hogy $P(A)$ minden $w \in W$-re változatlan, megkapjuk, hogy

$$\hat{w} = \arg\max_{w \in W} P(A|w)P(w). \tag{3}$$

A továbbiakban a *generatív megközelítést* követjük, Eq. (3)-at alkalmazva [22]. $P(w)$-t általában egy *nyelvi modell* szolgáltatja, amit adottnak tételezünk fel és $P(A|w)$-re koncentrálunk, ami kisebb egységek valószínűségértékeire bontható: a w szót $o_1, \dots, o_n$ fonémasorozatként kezeljük majd, míg A-t nem átfedő $A_1, \dots, A_n$ szegmensekre bontjuk, ahol minden A_j a megfelelő o_j fonémához tartozik. Ezután, feltételezve, hogy a fonémák függetlenek [8], felírhatjuk, hogy

$$P(A|w) = \prod_{j=1}^{n} P(A_j|o_j). \tag{4}$$

Egy *keretalapú modellben* a $P(A_j|o_j)$ értékek tovább bontódnak, míg egy *szegmensalapú környezetben* ezek jellemzően a fonémaosztályozó közvetlen kimenetei. Ebben a dolgozatban keret- és szegmensalapú megközelítést is alkalmazunk.

A legtöbb használt keresési módszer balról jobbra halad, egyszerre egy fonémát illesztve az aktuális szóprefix végére. Ezek a prefixek az ún. *hipotézisek*, melyek $(o_1 \dots o_j, [A_1, \dots, A_j])$ alakú objektumpárok. Egy új fonéma és új befejező időpont hipotézis végére illesztését a hipotézis *kiterjesztésének* nevezzük.

A dolgozatban a multi-stack decoding és a Viterbi beam search keresési eljárásokat használtuk, melyek egymás változatainak is tekinthetők. Mindkettő megvalósítható úgy, hogy adatstruktúrákat (*vermeket*) rendelünk minden lehetséges fonémahatárhoz; ezek a vermek (melyek nem a jólismert LIFO- vagy FIFO-típusú vermek) a hipotéziseket azok valószínűsége szerint rangsorolják. Egy korlátozott méretű verem csak a legvalószínűbb n hipotézist tartja meg, az ezen fölülieket egyszerűen eldobja. A multi-stack decoding algoritmus ilyen korlátozott méretű vermeket használ, míg a Viterbi beam search eljárás csak a legvalószínűbb hipotézist tartja meg, valamint azokat, amelyek (valószínűségeik alapján) ehhez közel esnek. Ez a „közelség" a módszer paraméterével (*sugárszélesség*) szabályozható. A többi tekintetben a két keresőeljárást tekinthetjük azonosnak.

Először a kezdőhipotézis kerül a bemondás kezdetéhez tartozó verembe, majd növekvő időrendben bejárjuk a vermeket. Minden lépésben kivesszük a hipotéziseket az aktuális veremből, mindegyiket minden lehetséges módon kiterjesztjük, majd a generált hipotéziseket belehelyezzük az (új) befejezési idejükhöz tartozó verembe. Az eljárás kimenete az utolsó veremben található hipotézisek közül a legvalószínűbb olyan lesz, melynek fonémasorozata megfelel a szótár egy elemének [1].

Módszerkombinációk	Sebességjavítás Számadatbázis	Sebességjavítás Telefonos adatbázis
Viterbi beam search	100.00%	100.00%
multi-stack decoding	68.65%	86.98%
multi-stack + i	241.60%	482.57%
multi-stack + ii	147.77%	358.48%
multi-stack + iii	351.54%	86.98%
multi-stack + iv	166.58%	379.47%
multi-stack + $i + ii + iii + iv$	756.55%	704.97%

1. táblázat. A bevezetett javítások által elért sebességjavítás mértéke a Számadatbázisra és a Telefonos adatbázisra (mindkettő izoláltszavas felismerés, szegmensalapú eset), a felismerési pontosság megőrzése mellett. A sebességjavítás fordítottan arányos a futási sebességgel.

1.1. A multi-stack decoding eljárás javítása

A multi-stack decoding keresési algoritmus gyenge pontja, hogy egyenlő méretű vermeket használ az egész bemondáson, mely méret azonban csak pár ponton szükséges. (A Viterbi beam search módszerre ugyanez igaz a konstans sugárszélességével.) A veremméret jobb meghatározásával pontosságvesztés nélküli sebességnövekedést érhetnénk el, melyre a következő technikákat alkalmaztuk:

i) Kombináltunk a multi-stack decoding és a Viterbi beam search eljárásokat: minden veremben csak az n legjobb hipotézist tartottuk meg, de a legjobbnál sugárszélességnél rosszabbakat is eldobtuk.

ii) A bemondás kezdetén nagyon kevés információnk van annak eldöntésére, hogy mely hipotéziseket érdemes megtartanunk, így itt nagyobb vermekre van szükség, mint később. Erre egy egyszerű megoldást alkalmaztunk: a t_i időponthoz tartozó verem mérete $s \cdot m^i$ volt, ahol $0 < m < 1$ és s az első verem mérete.

iii) A következő javítás a vermek egy módosítása. Igen gyakori, hogy több olyan hipotézisünk is van, amelyek fonémasorozata és befejező időpontja is megegyezik (viszont valamelyik korábbi fonémahatáruk eltér). Ilyen esetekben elég a legvalószínűbbet megtartani, hiszen a további kiterjesztések csak a befejező időponttól és a fonémasorozattól függnek. A vermek ilyen működése könnyen megoldható, azonban ez minden új hipotézis eltárolásakor egy időigényes vizsgálathoz vezet. Emiatt egy egyszerűbb, közelítő változatot alkalmaztunk: a vermek több ilyen ,,egyező" hipotézist is tárolhatnak, a vizsgálatot a hipotézisek kiterjesztésekor alkalmazzuk. Ha az aktuális hipotézis fonémasorozata megegyezik az előtte kiterjesztett hipotézisével, a hipotézist elvetjük.

iv) Ez az ötlet abból a megfigyelésből ered, hogy csak tényleges fonémahatárokon van szükség nagyméretű vermekre. Amennyiben meg tudnánk határozni az egyes időpontok fonémahatárságának valószínűségét, csökkenteni tudnánk a bejárt keresési tér méretét. Erre a feladatra egy neuronhálót tanítottunk a 13 MFCC Δ jellemzőn, kimenetét egy p valószínűségnek tekintve. Az aktuális veremméret p-től függött: a bonyolultabb módszer egy exponenciális függvényt alkalmazott erre, míg az egyszerűbb egy lépcsős függvény volt azokra az esetekre, ahol könnyen illeszthető változatra volt szükség.

Ezt a négy javítást két adatbázison teszteltük, azonos pontosság mellett minél nagyobb sebességjavítást megcélozva (ld. 1. táblázat). Megmutattuk, hogy össze is kombinálhatóak, és így még nagyobb sebességjavítást értünk el. Az elért sebességjavítás mértéke rámutat a standard keresési eljárások gyenge pontjára: a bemondáson belüli pozíció figyelmen kívül hagyására.

Alkalmazott lépések				d^1 (minimum)		d^2 (átlag)	
$\mathcal{P}_0$	$\mathcal{P}_1$	$\mathcal{P}_2$	$\mathcal{P}_3$	$\mathcal{D}_{\min}$	$\mathcal{D}_{\max}$	$\mathcal{D}_{\min}$	$\mathcal{D}_{\max}$
•	○	○	○	100.00%	100.00%	100.00%	100.00%
•	•	○	○	187.81%	188.96%	270.96%	233.32%
•	○	•	○	217.46%	188.23%	–	160.61%
•	○	○	•	–	–	–	**338.70%**
•	•	•	○	162.76%	229.64%	–	172.62%
•	•	○	•	–	–	–	189.22%
•	○	•	•	–	–	–	248.47%
•	•	•	•	–	–	–	194.60%

2. táblázat. A bevezetett többlépéses keresőeljárás sebességjavítási eredményei a Gyerekadatbázison (izoláltszavas felismerés, szegmensalapú eset), a multi-stack decoding eljáráshoz képest. „•" jelzi az adott lépés használatát, „○" a kihagyását, „–" azt, hogy nem sikerült elérni az elvárt pontosságot.

1.2. Egy többlépéses keresőeljárás

Léteznek **többlépéses keresőeljárások** is: ezek először a kevésbé valószínű hipotéziseket vetik el valamilyen kis számításigényű feltétel alapján. Ezután a későbbi lépésekben csak a fennmaradó hipotéziseket vizsgálják meg bonyolultabb, megbízhatóbb eljárásokkal, melyek pontosabban becslik meg a hipotézis-valószínűségeket [27]. Mi a következőkben egy fonémacsoportosításon alapuló többlépéses keresési eljárást készítünk: így szavakat vonunk össze, kisebb szótárban pedig gyorsabban lehet keresni. A későbbi lépésekben csak a megelőző lépésben valószínűbbnek talált szavak maradnak a szótárban, melyeket már alaposabban is megvizsgálhatunk.

A fonémahalmazt egy standard klaszterező eljárással klaszteretztük, amely egy, az eredeti fonémák közti d távolságra épül. Kezdetben minden fonéma külön klasztert alkot; ezután minden lépésben összevonjuk a két legközelebbi klasztert, amíg ez a minimális távolság el nem ér egy L küszöböt. Két klaszter távolságának választhatjuk elemeik páronként vett távolságának minimumát ($\mathcal{D}_{\min}$) vagy maximumát ($\mathcal{D}_{\max}$) [19]. Módszerünk a $P(a_i|o_j)$ (keretszintű) vagy $P(A_j|o_j)$ (szegmensszintű) valószínűségeket szolgáltató fonémaosztályozó eljárás hibáira épül: annak M tévesztési mátrixát használjuk, ahol $m_{i,j}$ az ω_j osztályba tartozó, de ω_i osztályba sorolt elemek száma. Először *normalizáljuk*, hogy a félreosztályozott elemek *arányát* tükrözze; azaz

$$m'_{i,j} = \frac{m_{i,j}}{\sum_k m_{k,j}}, \quad 1 \leq i,j \leq K, \tag{5}$$

ahol K az osztályok száma. Ezután a d fonémák közti távolságfüggvényt M' alapján, két változatban számoljuk:

$$d^1_{i,j} = \begin{cases} 0 & \text{ha } i = j \\ \infty & \text{ha } m'_{i,j} = m'_{j,i} = 0 \text{ és } i \neq j \\ -\log(m'_{i,j}) & \text{ha } m'_{j,i} = 0 \text{ és } m'_{i,j} \neq 0 \\ -\log(m'_{j,i}) & \text{ha } m'_{i,j} = 0 \text{ és } m'_{j,i} \neq 0 \\ \min\big(-\log(m'_{i,j}), -\log(m'_{j,i})\big) & \text{egyébként,} \end{cases} \tag{6}$$

és

$$d^2_{i,j} = \begin{cases} 0 & \text{ha } i = j \\ \infty & \text{ha } m'_{i,j} = m'_{j,i} = 0 \text{ és } i \neq j \\ -\log\big((m'_{i,j} + m'_{j,i})/2\big) & \text{egyébként.} \end{cases} \tag{7}$$

Alkalmazott lépések			d^1 (minimum)		d^2 (átlag)	
$\mathcal{P}_0$	$\mathcal{P}_1$	$\mathcal{P}_2$	$\mathcal{D}_{\min}$	$\mathcal{D}_{\max}$	$\mathcal{D}_{\min}$	$\mathcal{D}_{\max}$
•	○	○	100.00%	100.00%	100.00%	100.00%
•	•	○	174.82%	**241.20%**	204.61%	203.09%
•	○	•	160.95%	216.21%	151.91%	**239.59%**
•	•	•	113.29%	184.68%	126.18%	168.30%

3. táblázat. A bevezetett többlépéses keresőeljárás sebességjavítási eredményei a Telefonos adatbázison (izoláltszavas felismerés, keretalapú eset), a multi-stack decoding eljáráshoz képest.

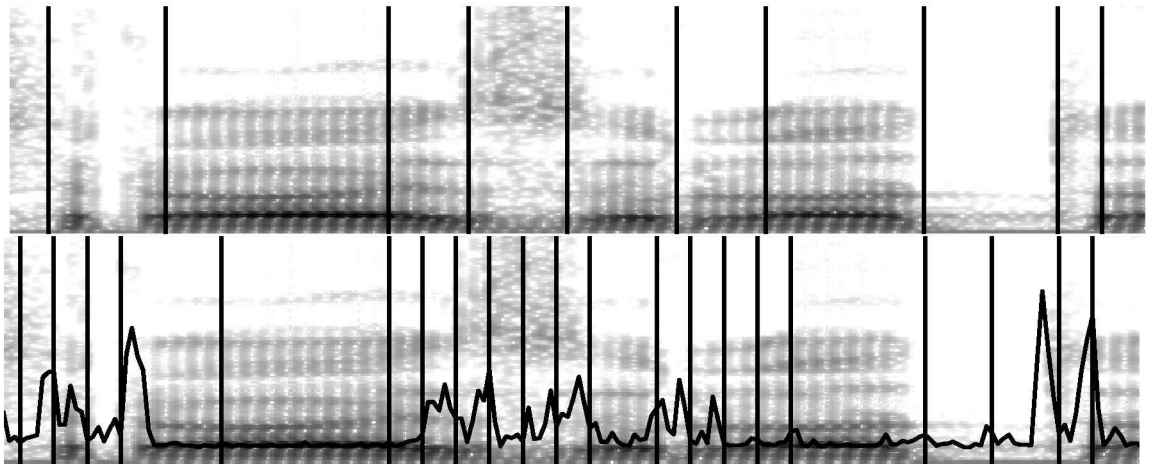
Ezután legyen D' egy, D^1-en vagy D^2-n működő legrövidebbút-kereső eljárás kimenete; D' tartalmazza a klaszterezéshez használt fonématávolságokat. Így négy klaszterezési változatunk van a távolságfüggvénytől (d^1 vagy d^2) és a klaszterezési eljárástól ($\mathcal{D}_{\min}$ vagy $\mathcal{D}_{\max}$) függően.

A klaszterezési eljárás megállási küszöbének (L-nek) meghatározásához egy diagrammon ábrázoltuk a klaszterek számát L függvényében, majd olyan L értékeket választottunk, ahol két klaszterösszevonás között jelentős szakadás látszott. Így számos lehetőséget kaptunk, melyek mindegyikéhez egy fonémacsoportosítás és egy lehetséges, $\mathcal{P}_i$-vel jelölt keresési lépés tartozott ($\mathcal{P}_0$ az eredeti lépés). A felhasznált lépéseket és a hozzájuk tartozó veremméreteket teszteléssel választottuk ki, két izoláltszavas felismeréses adatbázison: egyet szegmens-, egyet pedig keretalapú megközelítésben. Az utóbbinál kevesebb lehetséges keresési lépés adódott a kevesebb fonéma miatt, mivel a rövid-hosszú párok nem voltak megkülönböztetve. A tesztek során változtattuk az alkalmazott lépéseket és a veremméreteket; a minimális felismerési pontosságot elérő konfigurációk közül a leggyorsabbat választottuk (ld. a 2. és 3. táblázatot). Az elért sebességjavítási értékek azt mutatják, hogy a keresési eljárás igen hatékony. A Gyerekadatbázis számos sikertelen tesztesete valószínűleg a túl szűk $\mathcal{P}_3$ fonémacsoportosításnak tudható be, viszont $\mathcal{P}_3$ bizonyult a leggyorsabbnak d^2-t és $\mathcal{D}_{\max}$-ot használva , és ez a konfiguráció (a legszűkebb fonémacsoport ($\mathcal{P}_2$) d^2-t és $\mathcal{D}_{\max}$-ot használva) jól teljesített a másik adatbázison is.

1.3. Szegmensalapú beszédfelismerés felgyorsítása szegmentálással

A keresési eljárás során hipotézisekkel dolgozunk, melyek fonémákat rendelnek szegmensekhez; ezekre az A_i szegmensekre hivatkozhatunk $[t_{i-1}, t_i]$ határpontjaikkal is. Eddig a t_i értékekre csak a triviális megkötések vonatkoztak (egész számok növekvő sorrendben és $1 \leq t_i \leq t$). A következőkben csak egy T halmaz elemeit engedélyezzük; T a lehetséges szegmenshatárok halmaza, míg a T-t meghatározó eljárás a *szegmentáló algoritmus*. Elég valószínűtlen, hogy 1 és t között az összes értékre szükség van, fölösleges értékek T-ben tartása viszont a keresési eljárás futási idejének haszontalan megnövekedésével jár. Ráadásul ideális esetben T egybeesik a valós fonémahatárok halmazával, melyek remélhetőleg meghatározhatóak jelfeldolgozási eszközökkel [26].

Az alap szegmentáló algoritmus minden k. keretnél javasol egy szegmenshatárt: ez egy primitív, ámde kielégítően működő módszer. A következőkben bevezetünk három új eljárást, amelyek minden a_i kerethez egy b_i határvalószínűséget várnak; a b_i értékeket esetünkben egy, a klasszikus 13 MFCC Δ jellemzővektort használó neuronháló szolgáltatta. Logikusnak tűnhet, hogy a b_i értékek meghatározása megoldja az egész szegmentálási problémát; ezt a gondolatmenetet követve az első két eljárás működése elég egyszerű. A **Küszöbölő algoritmus** szintén minden k. keretnél javasol határt, ám csak ha a megfelelő b_i határvalószínűség átlép egy $p_{\min}$ küszöböt (ld. az 1. ábrát). Ez



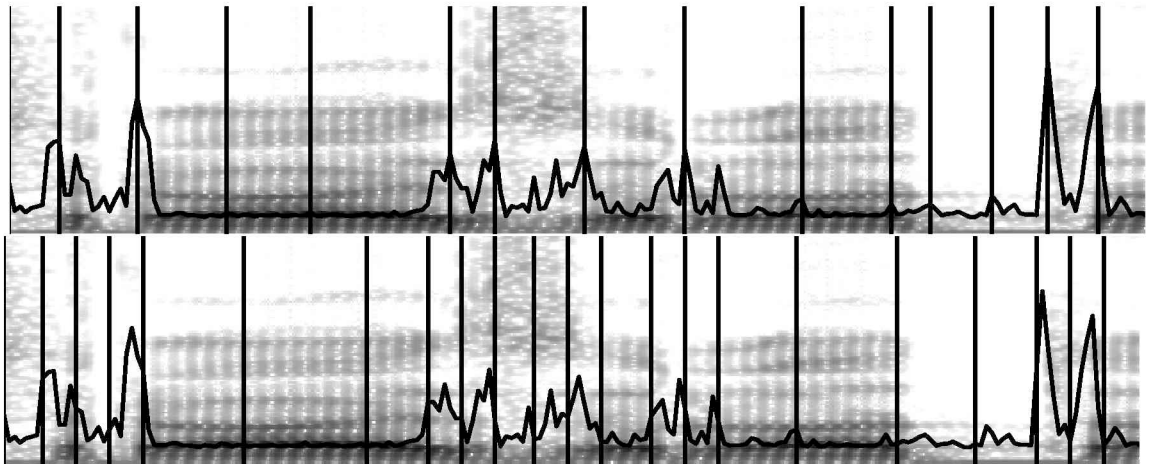
1. ábra. Fent: egy bemondásrészlet spektruma a valós szegmentálással. Lent: ugyanaz a spektrum a b_i értékekkel és a Küszöbölő algoritmus kimenetével.

Algoritmus	Paraméterek	Helyesség	Pontosság	Sebességjavítás
Küszöbölő	$k = 6$, $p = 0.065$	96.06%	94.94%	110.83%
Küszöbölő	$k = 6$, $p = 0.070$	97.49%	95.70%	87.70%
Maximum	$k = 3$	93.90%	90.68%	78.29%
Maximum	$k = 4$	95.34%	92.83%	137.02%
LIF	$k = 6$, $p = 1.75$	97.85%	96.42%	155.57%
LIF	$k = 7$, $p = 1.30$	98.20%	95.69%	127.26%
Referenciaérték	—	96.42%	95.34%	100.00%

4. táblázat. A Küszöbölő, Maximum és LIF algoritmusok néhány fontosabb eredménye az Orvosi adatbázison (mondatfelismerés, szegmensalapú eset), az alkalmazott paraméterekkel.

az eljárás elég rosszul teljesített: paraméterhangolással el lehetett érni egy kis sebességnövekedést, de ekkor a pontosságértékek romlottak; más beállításokat használva javult a pontosság, ám csak lassulás mellett. Ennek oka valószínűleg az volt, hogy a spektrum változása (mely a b_i értékekben tükröződik) gyakran hosszabb idő alatt következik be relatíve alacsony b_i értékek mellett, melyeket ez a módszer képtelen volt észlelni. Második eljárásunk, a **Maximum algoritmus** olyan pontokat keres, ahol a b_i értékek egy adott intervallumon lokális maximumukat veszik fel. A vizsgált környezet méretét (k) módosítva beállíthatunk a határok között egy minimális távolságot (ld. a 2. ábrát). Bár ez az eljárás tűnt a legígéretesebbnek, az eredmények nem ezt igazolták: kevés fölösleges határt javasolt ugyan, de elhagyott néhány szükségeset is, ami nagyon komoly hiba. Emellett az algoritmust csak mérsékelten lehetett hangolni, mivel egyetlen paramétere egy kis egész szám.

A két eljárás kudarca alapján némileg bonyolultabb megoldásra van szükségünk. A **LIF algoritmus** dinamikusan próbálja úgy beállítani a határokat, hogy sűrűségük arányos legyen a lokális b_i értékekkel. Erre egy megoldás a „leaky integrate-and-fire" (LIF) neuronmodell alkalmazása [6]. Működése elég egyszerű: a neuron addig halmozza a bemeneti értékeket, míg összegük el nem ér egy küszöböt; ekkor a neuron kisül, potenciálja lenullázódik és az egész eljárás elölről kezdődik. A halmozás „lyukassá" is tehető, ekkor az összeg egy időkonstanssal csökken. A modellbe egy kifáradási periódus is belefoglalható, mely alatt a neuron nem gerjeszthető. Szegmentálóalgoritmusunkban a b_i értékek szolgálnak a neuron bemeneteként: ahol ezek az értékek magasabbak, összegük hamarabb eléri az aktivációs



2. ábra. Fent: az 1. ábrán is látható spektrum a b_i értékekkel és a Maximum algoritmus kimenetével. Lent: ugyanaz a spektrum a b_i értékekkel és a LIF algoritmus kimenetével.

küszöböt, így a kisülések gyakoribbá válnak. A határok gyakorisága a p küszöbbel állítható, míg a k periódus változtatásával a határok közti minimális távolság határozható meg (ld. a 2. ábrát). A másik két eljárással ellentétben ez az algoritmus jelentős gyorsulást ért el pontosságcsökkenés nélkül, sőt a pontosságértékeken és a sebességértékeken egyszerre is tudott javítani (ld. a 4. táblázatot). Míg a Maximum algoritmus nagyon jól találta meg a potenciális szegmenshatárokat, nem lehetett hangolni; a Küszöbölő algoritmus két paraméterével nagyon jól hangolható, maga az eljárás azonban túl egyszerű volt. A LIF algoritmus lehet az arany középút: két hangolható paraméterrel rendelkezik, a Küszöbölő algoritmusnál intelligensebb pozíciókra helyezi határait, és figyelembe veszi a környezetet.

A fejezet eredményeinek tézisszerű összefoglalása

I/1. A szerző létrehozott egy többlépéses keresési eljárást, hogy felgyorsítsa a beszédfelismerési feladat keresési folyamatát. Az eljárás több, hierarchikus fonémacsoportot alkalmaz, melyek az eredeti fonémakészletnek a fonémaosztályozó tévesztési mátrixán alapuló klaszterezésével lettek meghatározva. A fonémacsoportok hierarchikus tulajdonságát kihasználva a megfelelő keresési lépések könnyen implementálhatóak, mivel egy tömörebb fonémacsoportosítás szavakat von össze. Ezzel az eljárással sikerült a keresési folyamatot jelentősen felgyorsítani (publikálva: [13; 14; 25]).

I/2. A szerző megkonstruált egy eljárást a fonémák határainak pontosabb detektálására. Erre alapozva elkészített egy, a fonémák közti lehetséges határokat kiszámító algoritmust. Az eredmények tükrében ez a módszer valóban alkalmazható a beszédfelismerési eljárás felgyorsítására a standard szegmentálási eljáráshoz képest, ezenfelül a módszer pontos paraméterbeállításával a jelentős gyorsítás mellett sikerült a felismerési pontosságon is javítani (publikálva: [18]).

I/3. A szerző tanulmányozta a multi-stack decoding és a Viterbi beam search kereső eljárásokat, melyek közösek abban, hogy hipotéziseket tárolnak minden lehetséges fonémák közötti határnál. A szerző számos módszert vezetett be, melyek a tárolt hipotézisek számát pontonként korlátozták, és így képes volt felgyorsítani a beszédfelismerési probléma keresési folyamatát azonos vagy csak kicsit alacsonyabb felismerési pontosság mellett (publikálva: [11; 12; 14; 16; 25]).

2. A beszédfelismerés pontosságának növelése

A dolgozat második részében olyan területekkel foglalkozunk, melyek növelhetik a beszédfelismerés pontosságát. Mivel igen sok ilyen terület létezik, néhány ilyenre kell korlátoznunk vizsgálatainkat: különböző aggregációs operátorokat fogunk felhasználni a hipotézisköltségek kalkulációja során, és megvizsgáljuk az antifonéma-problémát, mivel ezek gyakorlatilag nem növelik a futási időt.

2.1. Hipotézisköltségek számítása aggregációs operátorokkal

Beszédfelismerés során azt a $\hat{w} \in W$ szót keressük, melyre $P(A|w)P(w)$ maximális. Az első együttható szétbontható a következőképpen:

$$P(A|w) = \prod_{j=1}^{n} P(A_j|o_j), \tag{8}$$

ahol a w szót a $o_1, \ldots, o_j$ fonémasorozattal azonosítottuk, míg $A = A_1, \ldots, A_n$ az $A = a_1, \ldots, a_t$ beszédjel szétbontása fonémánként egy-egy nem átfedő szegmensre. Ez a képlet a függetlenségi feltevésen alapszik, amiről azonban már többször bebizonyosodott [31], hogy hamis; így más, a problémához esetleg a szorzásnál jobban illeszkedő operátorok is alkalmazhatók. Ez lehetővé teszi egy általánosabb megközelítés használatát [29], nevezetesen

$$P(A|w) = g_2\big(P(A_1|o_1), P(A_2|o_2), \ldots, P(A_n|o_n)\big), \tag{9}$$

ahol a g_2 operátor a fonémaszintű valószínűségekből számít hipotézisszintűeket. Ugyanígy egy másik operátor (g_1) használható a fonémaszintű $P(A_j|o_j)$ valószínűségek alacsonyabb szintű forrásokból történő meghatározásához. Egy tipikus *keretalapú* környezetben (mint például a Hidden Markov Model (HMM) alkalmazásakor) g_1 jellemzően az a_j kereteken alapszik, és a következőképpen definiált:

$$P(A_j|o_j) = g_1(A_j, o_j) = g_1(a_{t_{j-1}}, \ldots, a_{t_j}, o_j) = \prod_{l=t_{j-1}}^{t_j} c_{o_j}^{t_j - t_{j-1}} \cdot P(a_l|o_j), \tag{10}$$

ahol c_{o_j} az o_j fonémához tartozó állapotban maradás valószínűsége, és általában egy Gaussian Mixture Model (GMM) szolgáltatja a $P(a_l|o_j)$ értékeket. *Szegmensalapú* esetben g_1 tipikusan a fonémaosztályozó eljárás megfelelő kimenetének értéke hossznormalizálás után [29]. A következőkben a g_1 vagy g_2 operátorokat fogjuk változtatni; így befolyásoljuk a szó-szegmens párok valószínűségét, és remélhetőleg növeljük a felismerési pontosságot.

2.1.1. Középaggregációs operátorok használata

Első lépésben olyan operátorokat kerestünk, melyek viselkedése tág határok között módosítható, így fordultunk a fuzzy *középaggregációs operátorokhoz*; először a *hatványközép operátort* [4] alkalmaztuk:

$$G_\alpha(x_1, \ldots, x_j) = \left(\frac{x_1^\alpha + \ldots + x_j^\alpha}{j}\right)^{\frac{1}{\alpha}}, \qquad \alpha \in \mathbb{R}. \tag{11}$$

Jólismert [20], hogy ha $\alpha \to -\infty$, $G_\alpha \to \min(x_1, \ldots, x_j)$; G_{-1} a harmonikus közép; ha $\alpha \to 0$, G_α a

Alkalmazott operátor	α	Pontosság
Semmi (referenciaérték)	–	77.83%
G_α	0.6	84.35%
$G_{\alpha,\lambda}$ $\lambda = 0.7$	0.4	83.04%
$G_{\alpha,\lambda}$ $\lambda = 0.8$	0.5	83.91%
$G_{\alpha,\lambda}$ $\lambda = 0.9$	0.5	84.78%
$G_{\alpha,\lambda}$ $\lambda = 1.0$	0.6	84.35%

5. táblázat. A hatványközép operátor és a csökkenő hatványközép operátor legjobb eredményei a Gyerekadatbázison (izoláltszavas felismerés, szegmensalapú eset).

Alkalmazott operátor	α g_1-ként	α g_2-ként	Pontosság
Semmi (referenciaérték)	–	–	94.20%
G_α g_1-ként	0.94	–	95.13%
G_α g_2-ként	0.94	1.00	95.13%

6. táblázat. A hatványközép operátor legjobb eredményei a Telefonos adatbázison (izoláltszavas felismerés, keretalapú eset). Első lépésben a g_1 operátor α-ját optimalizáltuk, majd a g_2 operátorét.

mértani középhez tart; G_1 a számtani közép; és ha $\alpha \to \infty$, $G_\alpha \to \max(x_1, \ldots, x_j)$. Az α paramétert változtatva folyamatos átmenetet kapunk minimumtól maximumig, tehát ez az operátor igen rugalmas, ami egy alkalmazásnál praktikus tulajdonság. Ezután az operátor egy változatát vezetjük be, a *csökkenő hatványközép operátort*:

$$G_{\alpha,\lambda}(x_1, \ldots, x_j) = \left(\frac{\lambda^{j-1} x_1^\alpha + \lambda^{j-2} x_2^\alpha + \ldots + \lambda x_{j-1}^\alpha + x_j^\alpha}{j} \right)^{\frac{1}{\alpha}}, \tag{12}$$

ahol $\alpha \in \mathbb{R}$ ugyanaz, mint a hatványközép operátor esetében, $\lambda \in [0, 1]$ pedig egy súlyozási paraméter a későbbi értékek fontosabbá tételéhez. Esetünkben az utolsó érték a hipotézis utolsó fonémájához vagy keretéhez tartozik, így ez az operátor ideillő változatnak tűnik.

Következő lépésben ezeket az operátorokat szeretnénk g_1-ként vagy g_2-ként alkalmazni, amihez be kell állítanunk a paramétereiket. Az egyparaméteres hatványközép operátor elég könnyen optimalizálható: először előzetes tesztekkel egy olyan intervallumot határoztunk meg, amelyben kielégítően működött, majd egy kis lépésközzel bejártuk azt. A csökkenő hatványközép operátor esetében először rögzítettünk néhány λ értéket, ezután a fennmaradó α paramétert már beállíthattuk az előző módon.

Szegmensalapú esetben természetesen csak g_2 cserélhető le, hiszen ekkor g_1 nem valódi operátor. A G_α hatványközép operátor és a $G_{\alpha,\lambda}$ csökkenő hatványközép operátor alkalmazásával elért legjobb eredmények, valamint az ezekhez tartozó paraméterértékek az 5. táblázatban olvashatóak. G_α használata igen jelentős pontosságjavuláshoz vezetett, ami azt sugallja, hogy jobban illeszkedik a problémához, mint az alapértelmezett szorzás; a csökkenő hatványközép operátor viszont ehhez képest csak minimális javulást ért el, ami azt jelzi, hogy nincs igazi előnye a hatványközép operátorral szemben, viszont jóval nehezebb megfelelően paraméterezni. A keretalapú esetben mind g_1-ben, mind g_2-ben használhattuk ezeket az operátorokat. Az egyszerűség kedvéért nem párhuzamosan állítottuk be mindkettő paraméterét, hanem először g_1-et optimalizáltuk, g_2-ként szorzást használva; ezután g_2 paraméterét állítottuk be, g_1-et az optimális értéken hagyva (ld. a 6. táblázatot). Tapasztalataink szerint g_1 lecserélése jelentős javulást hozott, azonban ezután g_2 módosítása már egyáltalán semmit.

t-norma-család	T_P	T_L	T_λ^{SS}	T_λ^{H}	T_λ^{Y}	T_λ^{D}	T_λ^{AA}
Legjobb pontosság	92.17%	0.26%	93.47%	92.60%	89.45%	93.47%	93.04%
Rel. hibacsökkenés	0.00%	—	16.60%	5.49%	—	16.60%	11.11%

7. táblázat. A standard háromszögnorma-családok alkalmazásával elért felismerési pontosságok és relatív hibacsökkenések a Gyerekadatbázison (izoláltszavas felismerés, szegmensalapú eset).

t-norma-család	T_P	T_L	T_λ^{SS}	T_λ^{H}	T_λ^{Y}	T_λ^{D}	T_λ^{AA}
Legjobb pontosság	91.69%	0.69%	92.61%	92.50%	—	92.25%	92.03%
Rel. hibacsökkenés	0.00%	—	11.07%	9.74%	—	6.73%	9.74%
Legjobb helyesség	92.03%	0.81%	93.31%	93.19%	—	93.03%	92.73%
Rel. hibacsökkenés	0.00%	—	16.06%	14.55%	—	12.54%	8.78%

8. táblázat. A standard háromszögnorma-családok alkalmazásával elért pontosság- és helyességértékek, valamint relatív hibacsökkenések az Orvosi adatbázison (mondatfelismerés, szegmensalapú eset).

2.1.2. Háromszögnormák használata

Nagyon sokféle operátort használhatunk akár g_1, akár g_2 esetében, de ezeknek csak töredékétől várhatunk jó működést. Emellett könnyen alkalmazható és hangolható operátorokat célszerű vizsgálnunk, így két feltételt állítottunk fel: működésüket könnyen lehessen módosítani (egy vagy több paraméterrel), és, mivel az eredeti operátorunk a szorzás, legyenek „szorzásszerűek". A *háromszögnormák* (triangular norms) a fuzzy logika standard operátorai [4; 23] és mindkét elvárást teljesítik, így őket alkalmaztuk. Mellettük szól az is, hogy a fuzzy logikában az ÉS-kapcsolatot képviselik, és a mi problémánkban is ÉS-kapcsolatok vannak (az A_1 szegmens x_1 valószínűséggel o_1 fonéma ÉS az A_2 szegmens x_2 valószínűséggel o_2 fonéma, stb.).

Definíció 1 *Egy $T : [0,1]^2 \to [0,1]$ függvény akkor és csak akkor háromszögnorma (t-norma), ha:*

(T1)	$T(1,x) = x$	*minden $x \in [0,1]$ esetén,*	*(határfeltétel)*
(T2)	$T(x,y) = T(y,x)$	*minden $x, y \in [0,1]$ esetén,*	*(kommutativitás)*
(T3)	$T(x,y) \leq T(u,v)$	*bármely $0 \leq x \leq u \leq 1$, $0 \leq y \leq v \leq 1$ esetén,*	*(T mindkét argumentumát tekintve nemcsökkenő)*
(T4)	$T(x,T(y,z)) = T(T(x,y),z)$	*minden $x, y, z \in [0,1]$ esetén.*	*(asszociativitás)*

Könnyen látható, hogy a szorzás operátor is háromszögnorma, így ezután T_P-vel jelöljük. Továbbá

Definíció 2 *Egy T t-norma* *folytonos* *ha T függvényként folytonos $[0,1]$-en;*
Archimédeszi *ha $T(x,x) < x$ minden $x \in (0,1)$-re.*

Ez a két tulajdonság elég fontosnak tűnik egy g_1 vagy g_2 operátorként használt függvény számára, így a következőkben csak olyan háromszögnormákat vizsgálunk, melyek mindkét kritériumot teljesítik. Számos, Klement, Mesiar és Pap [23] által felsorolt háromszögnormát teszteltünk. Egy t-norma kétoperandusú művelet, mi pedig tetszőleges számú operandust kezelni tudó operátort keresünk, ez azonban az asszociativitási tulajdonság (T4) használatával könnyen feloldható. A tesztelt normák egyparaméteresek (kivéve a Lukasiewicz normát, amelynek nincs paramétere), így kellően rugalmasak, ám még mindig elég könnyen hangolhatók. (A paraméterbeállítás ugyanúgy történt, mint a hatványközép

operátor esetében.) A standard háromszögnormák csak szegmensalapú környezetben, g_2-ként lettek tesztelve, de izoláltszavas (ld. a 7. táblázatot) és mondatfelismerési feladatban is (ld. a 8. táblázatot). A használt háromszögnormától függően igen jelentős pontosságnövekedést tudtunk elérni.

2.1.3. Kétparaméteres háromszögnorma használata

A következő lépésben a standard háromszögnormáknál általánosabb operátor használatát céloztuk meg. Dombi bevezetett egy általánosított háromszögnorma-családot, mely a következő formában írható le:

$$T_{GD,\gamma}^{(\alpha)} = \frac{1}{1 + D_\gamma(x)} \qquad \alpha > 0, \tag{13}$$

ahol

$$D_\gamma(x) = \left(\frac{1}{\gamma} \left(\prod_{i=1}^{n} \left(1 + \gamma \left(\frac{1 - x_i}{x_i} \right)^\alpha \right) - 1 \right) \right)^{1/\alpha} \tag{14}$$

és $\gamma > 0$ [3]. Ez az operátor sok ismert és népszerű háromszögnorma-családot magában foglal (pl. Dombi, Hamacher, Einstein operátor, vagy a szorzás művelet) speciális esetként.

Alkalmazása némileg bonyolultabb volt, mint a standard háromszögnormáké, mivel mindkét paraméterét (α és γ) be kellett állítani, szemben a korábbi eggyel (λ). Ráadásul, mivel ezt a két paramétert párhuzamosan kellett behangolni, a korábban alkalmazott módszerek nem voltak alkalmazhatók. Kézenfekvő megoldás lett volna mindkét paraméternek egy megfelelő intervallum meghatározása, majd ezen terület felderítése két egymásbaágyazott ciklussal, azonban ez nagyon megnövelte volna a szükséges tesztek számát.

Emiatt más gondolatmenetet követtünk: ha az alkalmazott norma két paraméterén kívül a beszédfelismerő rendszer összes beállítását rögzítjük, a pontosságot tekinthetjük a két paraméter függvényének, amit maximalizálunk. Esetünkben vagy csak g_1-et, vagy csak g_2-t módosítjuk, így ez egy kétdimenziós térben történő keresés, amire a SNOBFIT csomagot használtuk [21]. A kísérletek egy szófelismerési problémán, keretalapú megközelítésben folytak, g_1-et változtatva és g_2-t szorzásként hagyva. 500 iterációt engedtünk meg, amely az előző stratégia által követelt tesztszám töredéke. A felismerési hibát nagymértékben sikerült csökkenteni (ld. a 9. táblázatot): 50% fölötti relatív hibacsökkenést értünk el, ami a standard háromszögnormák eredményénél sokkal jobb érték.

2.1.4. Egy általánosabb megközelítés: a logaritmikus generátorfüggvény

A következőkben bevezetjük a háromszögnormák reprezentálásának egy általánosabb módját, és adunk egy alkalmazásközpontú módszert a norma adott feladathoz igazítására; ehhez azonban először egy újabb definícióra lesz szükségünk.

Tétel 1 *Egy T t-norma akkor és csak akkor folytonos és archimédeszi, ha létezik egy folytonos, szigorúan monoton csökkenő $f : [0,1] \to [0,\infty]$ függvény, melyre $f(1) = 0$, valamint*

$$T(x,y) = f^{(-1)}\big(f(x) + f(y)\big),$$

ahol $f^{(-1)}$ f pszeudóinverze:

$$f^{(-1)}(x) = \begin{cases} f^{-1}(x) & \textit{ha } x \le f(0) \\ 0 & \textit{egyébként.} \end{cases}$$

Ezen formula n-szeres kiterjesztése nyilván

$$T^N(p_1, p_2, \ldots, p_N) = f^{-1}\Big(\sum_i f(p_i)\Big). \tag{15}$$

Ez a reprezentáció konstans szorzótól eltekintve egyértelmű. Ha a t-norma szigorúan monoton növekvő, akkor f szigorúan monoton csökkenő, és az $f^{(-1)}$ pszeudóinverz függvény a szokásos f^{-1} inverz. A továbbiakban ezt az esetet alkalmazzuk; ekkor az f függvény a T norma *additív generátora* [4].

Ezután áttérhetünk a norma alkalmazására. Egy tipikus beszédfelismerési környezetben az alulcsordulás elkerülése érdekében a p valószínűségek helyett a $c = -\log p$ költségértékek használatosak. Az állandó valószínűség-költség konverziók kiváltásához először beemeljük ezt az átváltást (15)-be:

$$-\log\Big(f^{-1}\Big(\sum_{i=1}^{N} f(e^{-c_i})\Big)\Big). \tag{16}$$

Mivel mindig ezt a képletet használjuk, célszerű a negatív exponenciálisfüggvény kiszámítását belevennünk f-be; így bevezetjük a logaritmikus generátorfüggvényt:

$$\phi(x) = f(e^{-x}). \tag{17}$$

Ezután beírhatjuk $\phi(x)$-et Eq. (16)-ba, azaz

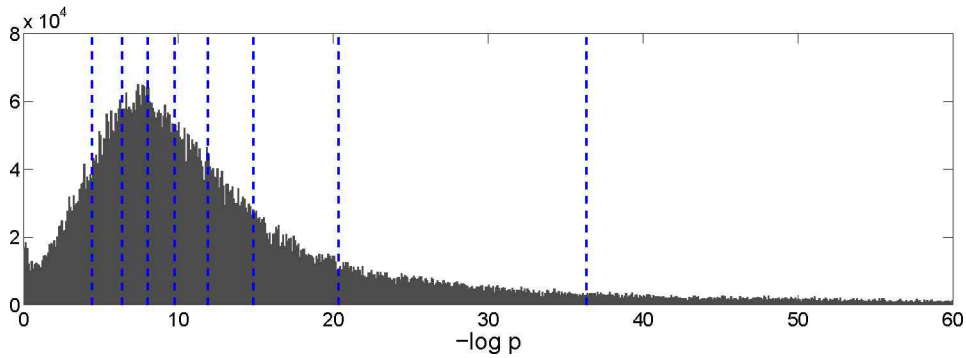
$$\phi^{-1}\Big(\sum_{i=1}^{N} \phi(c_i)\Big) = -\log\Big(f^{-1}\Big(\sum_{i=1}^{N} f(e^{-c_i})\Big)\Big) = -\log T(e^{-c_1}, e^{-c_2}, \ldots, e^{-c_N}), \tag{18}$$

tehát a $\phi(x)$ logaritmikus generátorfüggvényt alkalmazva ugyanazon a módon, ahogy az $f(x)$ additív generátorfüggvényt használtuk, ugyanazt a T t-normát kapjuk, csak mind operandusokként, mind eredményként költségértékkel valószínűség helyett. A logaritmikus generátorfüggvény szigorúan monoton növekvő, $\phi(x) : [0, \infty] \to [0, \infty]$, $\phi(0) = 0$, és konstansszorzótól eltekintve egyértelmű egy adott t-normához. $\phi(x)$ reprezentálása nagyon sok módon történhet: mi szakaszonként lineáris leírást választottunk. Legyen $\phi = \phi_{a_1,\ldots,a_n}^{m_1,\ldots,m_n} : [0, \infty] \to [0, \infty]$ egy szakaszonként lineáris, szigorúan monoton növekvő függvény az értelmezési tartományon vett $0 = a_0 < a_1 < a_2, \ldots, a_n < a_{n+1} = \infty$ töréspontokkal és $m_1, m_2, \ldots, m_n > 0$ meredekségekkel, és legyen $m_{n+1} = \lim_{x\to\infty} \phi'(x) = 1$. Azaz

$$\phi(x) = (x - a_j)m_{j+1} + \sum_{i=1}^{j}(a_j - a_{j-1})m_j, \;\; a_j \le x < a_{j+1}. \tag{19}$$

ϕ konstansszorzótól eltekintve egyértelmű; az $m_{n+1} = 1$ kikötéssel az ekvivalens reprezentációk közül választunk ki egyet. Az a_j kontrollpontokat olyan pozíciókra érdemes helyeznünk, ahol legjobban illeszkednek az adott problémához; ezután adott kontrollpontok esetén ϕ megadható a meredekségek alkotta n-elemű vektorral ($\overline{m} = (m_1, m_2, \ldots, m_n)$).

A kontrollpontok kiválasztására egy használati statisztikára alapozott eljárást vezettünk be. Feljegyeztük, hogy egy átlagos felismerés során milyen költségértékek fordulnak elő; az így előálló hisztogramot $n + 1$ egyenlő területű részre osztottuk, melyek határai lettek az a_i értékek (ld. a 3. ábrát). Ekkor, n-t lerögzítve, kiválasztva a kontrollpontokat, és $\overline{m}$-et optimalizálva (a felismerési pontosságot maximalizálva) a logaritmikus generátorfüggvény, tehát a t-norma is könnyen behangolható. A 9. táblázatban láthatók a módszer által elért eredmények egy keretalapú tesztben (g_1-ként, míg g_2 szorzás maradt): a logaritmikus generátorfüggvény teljesített a legjobban, de csak ha n értéke megfelelően nagy volt, hogy elegendő rugalmasságot biztosítson. Ráadásul a logaritmikus generátorfüggvény közvetlenül a költségértékeken működik, így ezt a legkönnyebb alkalmazni.



3. ábra. Egy tipikus beszédfelismerési folyamat során előforduló $-\log p$ értékek hisztogrammja a $[0, 60]$ intervallumon, és az $n = 8$ kontrollpont javasolt helyzete.

Módszer	Pontosság	Helyesség	Mondatok
Szorzás (referenciaérték)	96.76%	98.38%	92.66%
Dombi t-norma	97.57%	98.84%	93.33%
Általános Dombi operátor	98.49%	98.95%	94.66%
t-norma logaritmikus generátorral, $n = 8$	98.27%	98.84%	94.00%
t-norma logaritmikus generátorral, $n = 16$	98.84%	99.19%	96.00%

9. táblázat. Az egyes háromszögnormák legjobb elért pontosság- és helyességértékei az Orvosi adatbázison (mondatfelismerés, keretalapú eset), és a tökéletesen eltalált mondatok aránya.

2.2. Az antifonéma-probléma

Végül az antifonéma-problémával foglalkozunk, mely csak szegmensalapú megközelítésben szokott felmerülni. Ebben a cél azt eldönteni, hogy adott bemondásrészletek valamely ,,valós" fonémának felelnek-e meg. Ehhez számos pozitív példa adott egy kézileg címkézett adatbázis formájában, azonban nincsenek egyértelmű ellenpéldák, melyekbe minden más beletartozik, ami előfordulhat egy felvételben (pl. zaj, egy fonémánál rövidebb vagy hosszabb szegmens, stb.); az utóbbiak az ,,antifonémák" [7]. Mivel csak a pozitív osztályhoz vannak példáink, ez az *egyosztályos tanulás* [28] problémakörébe tartozik.

A beszédfelismerési feladat során a legvalószínűbb $\hat{w} \in W$ szót keressük, melyre

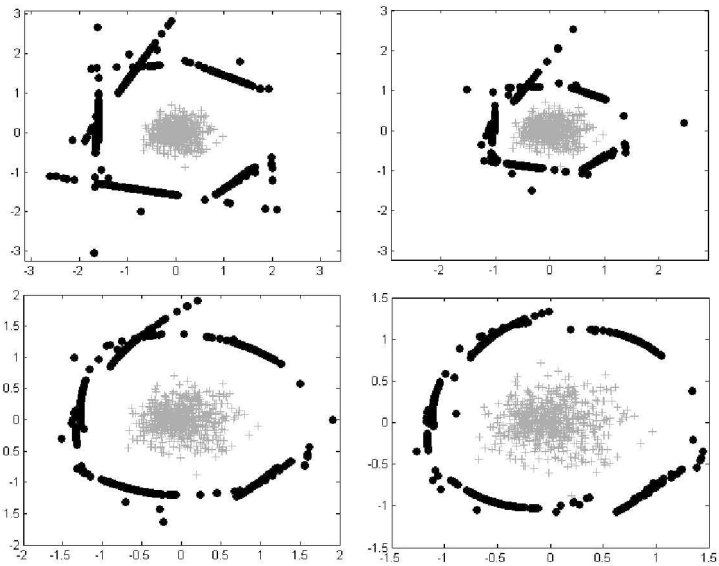
$$\hat{w} = \arg\max_{w \in W} P(A|w)P(w). \tag{20}$$

$P(A|w)$-re koncentrálva, feltesszük, hogy az A beszédjel felbontható nem átfedő szegmensekre. Mivel a jel helyes szegmentálása nem ismert, ezt egy rejtett S változóként írhatjuk le:

$$P(A|w) = \arg\max_{s \in S} P(A, s|w). \tag{21}$$

Modellünk feltevéseitől függően $P(A, s|w)$ számos módon bontható tovább, mivel S ezen a ponton még nem túl jól definiált. Mivel a szegmensekről feltesszük, hogy függetlenek, a lokális valószínűségértékeket egyszerűen összeszorozzuk; mi például a következő képletet használjuk: [30]

$$\prod_{j=1}^{n} \frac{P(o_j|A_j)P(\overline{\alpha}|A_j)}{P(o_j)}, \tag{22}$$



4. ábra. Generált ellenpéldák különböző beállításokkal. Balról jobbra: $(dist, curv) = (1, 0)$, $(dist, curv) = (0.4, 0)$, $(dist, curv) = (1, 0.5)$, $(dist, curv) = (1, 1)$. $curv = 0$ egy hipersík pontjait generálja, míg $curv > 0$ egy jobban illeszkedő hiperfelszín pontjait produkálja; $dist$ szabályozza a határpontok és a generált ellenpéldák távolságát.

ahol α jelöli az antifonémákat, így $P(\overline{\alpha}|A_j)$ annak valószínűsége, hogy az adott szegmens *nem* antifonéma. A következőkben a pontosság növelése érdekében a $P(\overline{\alpha}|A_j)$ valószínűségértékeket becslő módszereket fogjuk megvizsgálni.

Egy efféle egyosztályos osztályozási feladatot alapvetően kétféle módon lehet megoldani: használhatunk egy olyan módszert, amely az összes pozitív példát képes modellezni, vagy valamilyen eljárással ellenpéldákat generálhatunk. Az utóbbi esetben aztán az előálló két osztályhoz tartozó példákat valamilyen statisztikai alapú módszerrel (pl. ANN) választjuk el egymástól. Természetesen az automatikus ellenpélda-generálás módja nem magától értetődő, alapvetően ezt is kétféleképpen tehetjük: feladatspecifikusan, vagy valamilyen általános eljárást használva.

Az összes fonéma leírásának talán legkézenfekvőbb módja Gauss-görbék összegeként írni le eloszlásukat. A Gaussian Mixture Models (GMMs) [5] épp ezt csinálja, és alkalmazása a probléma kezelésének legelterjedtebb módja az irodalomban [7]. Másik alkalmazott módszerünket Tóth vezette be, mely ,,hibás" szegmensek generálására épül [30]. Ez az eljárás (a ,,Hibás szegmensmintavételezés") azon az ötleten alapszik, hogy a negatív példák valószínűleg olyan felvételrészletek, melyek kezdő- vagy végpontján valójában nincs fonémák közti határ. Ha ismerjük a valós fonémahatárokat – ami bármely tanítóadatbázis esetén elmondható –, könnyen generálhatunk negatív példákat a kezdő- vagy végpontok nem valós fonémahatárra helyezésével. A konkrét megvalósítás igen kézenfekvő: minden fonémához számos antifonémát generálunk egyik vagy mindkét fonémahatár δ ezredmásodperces eltolásával.

Ezen beszédfelismerés-specifikus módszer mellett általános ellenpéldageneráló eljárást is alkalmazhatunk, a pozitív példák X halmazát használva bemenetként, mely a negatív példák egy halmazát adja kimenetként. A következőkben Bánhalmi algoritmusát alkalmazzuk [2]; alapötlete, hogy minden pozitív $x \in X$ példát X-en kívülre vetít. Ehhez először X határpontjainak B halmazát határozza meg, majd minden pozitív példát keresztülvetít a hozzá legközelebb eső határponton. A vetítéshez

Eljárás	Pontosság	Helyesség
GMM 20 Gauss-komponenssel	91.12%	91.80%
Hibás szegmensmintavételezés	91.97%	92.62%
Általános ellenpéldageneráló eljárás, $curv = 0.0$, $dist = 1.5$	95.58%	95.82%
Antifonéma-modellezés nélkül (referenciaérték)	88.17%	88.53%

10. táblázat. A három alkalmazott módszer fontosabb eredményei az Orvosi adatbázison (mondatfelismerés, szegmensalapú eset).

két paramétert használ: $curv$ határozza meg az eredményként kapott felszín görbeségét, míg $dist$ szabályozza a generált ellenpélda és a felhasznált határpont távolságát. Végül ellenőrzi, hogy az újonnan generált pont valóban X-en kívül esik-e, mely a határpontok meghatározásánál használt módon történik. Ha a pont nem valós külső pont, az eljárást meg kell ismételni a második (harmadik, stb.) legközelebbi határponttal. Ezeket a lépéseket X összes elemére végrehajtja, így N negatív példát generál az N pozitívhoz (ld. a 4. ábrát).

Az eljárás nyilvánvaló gyengesége a nagyobb adatbázisok esetén igen magas időigénye. Esetünkben valóban igen nagyméretű adatbázisok fordultak elő, így az eljárás lefuttatása különböző $dist$ és $curv$ paraméterekkel irreálisan sok időt vitt volna el. Ennek – a módszer felhasználásához létfontosságú – orvoslására egy sor módosítást vezettünk be.

- Észrevettük, hogy az eljárás két, élesen elkülönülő részre osztható: az első határozza meg a határpontokat, míg a második végzi el az ellenpéldák tényleges meghatározását. Szerencsére az első felelős a nagy időkomplexitásért, a két paraméter ($dist$ és $curv$) viszont kizárólag a másodikban fordul elő. Így a tesztelt paraméterkombinációk számától függetlenül elég, ha az első – és lényegesen lassabb – részt csak egyszer hajtjuk végre.

- A második rész még tovább bontható: ha már ismerjük is a határpontok B halmazát, először minden x elemre meg kell határoznunk az $x_b \in B$ hozzá legközelebbi határpontot. Ez az eljárás sem hivatkozik a paraméterekre, így leválasztható a tényleges vetítésről, és elegendő, ha ezt is csak egyszer hajtjuk végre (persze szigorúan a határpontok meghatározása után), majd eredményét eltároljuk. Ezután új $dist$ és $curv$ értékek kipróbálásakor elég x_b indexét kiolvasnunk egy tömbből ahelyett, hogy újra meghatároznánk, ahogy az algoritmus eredeti változatában történt.

- Végül a tesztelés során azt találtuk, hogy az algoritmus utolsó lépése (melyben az eredményül kapott pontot vizsgáljuk meg, vajon az eredeti halmaznak valóban külső pontja-e) sem feltétlenül szükséges: nagyobb $dist$ értékek használatakor (esetünkben már ha $dist \geq 1.5$) a kapott pont az eredeti példahalmaztól túl távol esik ahhoz, hogy ez előfordulhasson. Természetesen a $dist$ érték csökkentésével a halmaz belsejébe vetítés esélye nő, de ez a kockázat elviselhető, mivel a vizsgálat (és az esetleges ismétlések) elhagyásával igen nagymértékű gyorsulás érhető el.

Ezen változtatások alkalmazásával módunk nyílt mindhárom módszer tesztelésére, melyet egy mondatfelismerési problémán végeztünk el (ld. a 10. táblázatot). Referenciaértéknek az antifonéma-modellezés elhagyásával elért felismerési pontosságértékeket választottuk. A tesztelt módszerek közül az általános ellenpélda-generáló eljárás alkalmazása vezetett a legjobb teszteredményekhez, azonban alkalmazásához szükség volt az itt bevezetett módosításokra: nélkülük ugyanis a módszer két paraméterét nem lehetett volna megfelelően behangolni az algoritmus futásiidő-igénye miatt.

	[13]	[18]	[16]	[11]	[12]	[14]	[25]	[15]	[24]	[10]	[17]	[9]
I/1.	•					•	•					
I/2.		•										
I/3.			•	•	•	•	•					
II/1.					•	•	•	•	•	•		
II/2.											•	
II/3.												•

11. táblázat. A tézispontok és a szerző publikációinak viszonya

A fejezet eredményeinek tézisszerű összefoglalása

II/1. A beszédfelismerési probléma keresési feladatában az előálló hipotézisek általában *költségük* alapján vannak rangsorolva, mely jellemzően két lépésben számítódik: először *fonémaszintű* költségeket aggregálnak a *keretszintűekből*, majd a hipotézisek költségeit határozzák meg a fonémaszintűekből. A szerző fuzzy függvényeket alkalmazott ezen a két szinten: *közép aggregációs operátorokat* és *háromszögnormákat* használt a beszédfelismerés pontosságának növelésére. A korábbi megfigyelések fontosságának csökkentésére bevezette a hatványközép operátor egy kétparaméteres változatát is. Egy felhasznált általánosított fuzzy metszet operátor mindkét paraméterének beállításához az alkalmazás feladatát átfogalmazta egy kétdimenziós minimalizálási problémává, majd egy globális optimalizáló eljárást használt annak megoldására. A beszédfelismerés pontosságában elért javítások azt jelzik, hogy az alkalmazott operátorok jobban illeszkednek a problémához, mint az eredeti szorzás művelet (publikálva: [10; 12; 14; 15; 24; 25]).

II/2. A szerző kidolgozott egy alkalmazásorientált eljárást a fuzzy metszet operátorok modellezésére a logaritmikus generátorfüggvény bevezetésével. Feltéve, hogy ez a függvény szakaszonként lineáris, bevezetett egy technikát a függvény aktuális problémához illesztésére az előforduló valószínűségek hisztogramjának felhasználásával. Ezt az eljárást a II/1-es tézispontban bevezetett minimalizálási problémában alkalmazta, és megmutatta, hogy nagyobb mértékben képes növelni a beszédfelismerési pontosságot, mint a tesztelt klasszikus normák (publikálva: [17]).

II/3. A szerző alkalmazott egy általános ellenpélda-generáló eljárást a szegmensalapú beszédfelismerési feladat antifonéma-problémájában. A felhasználáshoz szükséges volt az eljárás futási idejének radikális csökkentése, aminek elérésére a szerző számos módosítást is bevezetett. Így jelentősen csökkenteni tudta a beszédfelismerési hibát az antifonéma-problémára használt standard eljárásokhoz képest (publikálva: [9]).

Összegzés

A szerző szerint a beszédfelismerés legfontosabb szempontjai a pontosság és a sebesség, mivel a bemondott hangot a lehető legpontosabban kell felismerni, és lehetőleg valós időben. Mindkét területen egy sor technikát vezetett be, amelyek a kísérletek eredményei alapján sikeresnek tekinthetők. Legtöbb módszer rendelkezik paraméterekkel, melyeket egy új környezetben valószínűleg újra be kell állítani, azonban az ismertetett teszteredmények alátámasztják az eljárások hatékonyságát. Néhány igen általános módszert is sikerült bevezetni (logaritmikus generátorfüggvény, az általános ellenpélda-generáló algoritmus javításai), melyek a beszédfelismerési területen kívül is alkalmazhatóak.

Hivatkozások

[1] L. R. Bahl, P. Gopalakrishnan, and R. L. Mercer. Search issues in large vocabulary speech recognition. In *Proceedings of the 1993 IEEE Workshop on Automatic Speech Recognition*, Snowbird, UT, 1993.

[2] A. Bánhalmi, A. Kocsor, and R. Busa-Fekete. Counter-example generation-based one-class classification. In *Proceedings of ECML*, pages 543–550, 2007.

[3] J. Dombi. Towards a general class of operators for fuzzy systems. *IEEE Transaction on Fuzzy Systems*, 16(2):477–484, 2008.

[4] D. Dubois and H. Prade. *Fundamentals of Fuzzy Sets*. Kluwer Academic Publisher, 2000.

[5] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley & Sons, New York, 1973.

[6] W. Gerstner and W. M. Kistler. *Spiking Neuron Models*. Cambridge University Press, 2002.

[7] J. R. Glass. A probabilistic framework for segment-based speech recognition. *Computer Speech and Language*, 17(2):137–152, 2003.

[8] G. Gordos and G. Takács. *Digital Speech Processing (in Hungarian)*. Műszaki Könyvkiadó, 1983.

[9] G. Gosztolya, A. Bánhalmi, and L. Tóth. Using one-class classification techniques in the anti-phoneme problem. In *Proceedings of the 2009 Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, volume LNCS 5524, pages 433–440, Porto, Portugal, 2009.

[10] G. Gosztolya, J. Dombi, and A. Kocsor. Applying the Generalized Dombi Operator family to the speech recognition task. *Journal of Computing and Information Technology*, 17(3):285–293, 2009.

[11] G. Gosztolya and A. Kocsor. Improving the multi-stack decoding algorithm in a segment-based speech recognizer. In *Proceedings of the 16th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE)*, volume LNCS 2718, pages 744–749, Loughborough, England, UK, 2003.

[12] G. Gosztolya and A. Kocsor. Aggregation operators and hypothesis space reductions in speech recognition. In *Proceedings of the 2004 Conference on Text, Speech and Dialogue (TSD)*, volume LNCS 3206, pages 315–322, Brno, Czech Republic, 2004.

[13] G. Gosztolya and A. Kocsor. A hierarchical evaluation methodology in speech recognition. *Acta Cybernetica*, 17(2):213–224, 2005.

[14] G. Gosztolya and A. Kocsor. Speeding up dynamic search methods in speech recognition. In *Proceedings of the 18th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE)*, volume LNCS 3533, pages 98–100, Bari, Italy, 2005.

[15] G. Gosztolya and A. Kocsor. Using triangular norms in a segment-based automatic speech recognition system. *International Journal of Information Technology and Intelligent Computing (IT & IC) (IEEE)*, 1(3):487–498, 2006.

[16] G. Gosztolya, A. Kocsor, L. Tóth, and L. Felföldi. Various robust search methods in a Hungarian speech recognition system. *Acta Cybernetica*, 16(2):229–240, 2003.

[17] G. Gosztolya and L. L. Stachó. Aiming for best fit t-norms in speech recognition. In *Proceedings of the 2008 International Symposium on Intelligent Systems and Informatics (SISY) (IEEE)*, pages 1–5, Subotica, Serbia, Sept 2008.

[18] G. Gosztolya and L. Tóth. Detection of phoneme boundaries using spiking neurons. In *Proceedings of the 2008 International Conference on Artificial Intelligence and Soft Computing (ICAISC)*, volume LNCS 5097, pages 782–793, Zakopane, Poland, 2008.

[19] D. J. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. The MIT Press, 2001.

[20] G. H. Hardy, J. E. Littlewood, and G. Pólya. *Inequalities*. Cambridge University Press, 1968.

[21] W. Huyer and A. Neumaier. Snobfit – stable noisy optimization by branch and fit. *ACM Transactions on Mathematical Software*, 35(2):1–25, 2008.

[22] F. Jelinek. *Statistical Methods for Speech Recognition*. The MIT Press, 1997.

[23] E. P. Klement, R. Mesiar, and E. Pap. *Triangular Norms*. Kluwer Academic Publisher, 2000.

[24] A. Kocsor and G. Gosztolya. Application of full reinforcement aggregation operators in speech recognition. In *Proceedings of the 2006 Conference of Recent Advances in Soft Computing (RASC)*, Canterbury, UK, 2006.

[25] A. Kocsor and G. Gosztolya. The use of speed-up techniques for a speech recognizer system. *The International Journal of Speech Technology*, 9(3-4):95–107, 2006.

[26] T. N. Sainath. Acoustic landmark detection and segmentation using the McAulay-Quatieri sinusoidal model. Master's thesis, MIT, 2005.

[27] R. Schwartz, L. Nguyen, and J. Makhoul. *Multiple-pass Search Strategies*, chapter 18, pages 429–456. Kluwer Academic Publisher, Philadelphia, PA, 1996.

[28] D. M. Tax. *One-class classification; Concept-learning in the absence of counter-examples*. PhD thesis, Delft University of Technology, 2001.

[29] L. Tóth, A. Kocsor, and J. Csirik. On Naive Bayes in speech recognition. *International Journal of Applied Mathematics and Computer Science*, 15(2):287–294, 2005.

[30] L. Tóth, A. Kocsor, and G. Gosztolya. Telephone speech recognition via the combination of knowledge sources in a segmental speech model. *Acta Cybernetica*, 16(4):643–657, 2004.

[31] S. Young. Statistical modelling in continuous speech recognition. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, Seattle, 2001.