Statisztikai és szabály alapú morfológiai elemzők kombinációja beszédfelismerő alkalmazáshoz

Németh Bottyán[1], Mihajlik Péter[1], Tikk Domonkos[1], Trón Viktor[2]

[1] Budapesti Műszaki és Gazdaságtudományi Egyetem, TMIT, Budapest Magyar Tudósok Körútja 2. 1117,
{bottyan, mihajlik, tikk}@tmit.bme.hu
[2] International Graduate College Language Technology and Cognitive Systems
University of Edinburgh and Saarland University,
v.tron@ed.ac.uk

Kivonat: A magyar nyelvű számítógépes beszédfelismerésnél célszerűnek tűnik, hogy ne a szavakat, hanem a morfémákat vegyük alapegységnek a nyelvi modell felépítéséhez. Ehhez viszont szükséges, hogy a szavakat a morfémáknak megfelelő szegmentumokra bontsuk. A cikk egy új szegmentálási technikát ismertet, ami két különböző morfológiai szegmentáló módszer egyesítéséből született, és mindkét ősénél jobban alkalmazható számítógépes beszédfelismeréshez. Ennek a rendszernek az egyik pillére egy szabály alapú morfológiai elemző, a hunmorph, a másik pedig egy statisztikai alapokra épülő morfológiai szegmentáló, a morfessor. A kompozíció során igyekeztünk mindkét rendszer előnyeit megtartani, hátrányos tulajdonságait orvosolni. Ez nagyrészt sikerült is, leszámítva, hogy a morfessor által biztosított nyelvfüggetlenség a hunmorph bevonásával elveszett.

1 Bevezetés

A számítógépes beszédfelismerésben általában szó alapú nyelvi modellt használnak a felismeréshez. Ez a magyar nyelv esetén, ahol egy szónak rengeteg különböző alakja lehet, nem tűnik a legésszerűbb megoldásnak. Azt várnánk, hogy a morfémákra épülő nyelvi modellel pontosabb felismerési eredményeket érhetünk el. Elsőként [7] alkalmazott magyar nyelvű beszédfelismerésnél (diktáló rendszerben) morféma alapegységeket, azonban a felismerés pontosságát nem vetette össze a szó alapú megközelítésével. Később [8] szintén próbálkozott a morféma alapú nyelvi modellezéssel orvosi diktáló rendszerben, de még a morféma felismerési pontossága is lényegesen gyengébbnek adódott a szó alapú felismerésnél mért szófelismerési pontosságnál. Azonos metrikával előttünk nem hasonlították össze a morféma és szó alapú magyar nyelvű beszédfelismerési eredményeket. Más nyelveken számos sikeres kísérletet végeztek morféma alapú nyelvi modellek beszédfelismerési alkalmazásával [4], [5].

Mi először a morfessort [1], egy statisztikai tanuló algoritmus alapján működő szegmentálót próbáltuk ki. A kapott felismerési eredmények jobbak az egyszerű szó alapú felismerésnél, de a kapott szegmentumok nem feltétlenül felelnek meg valós

nyelvtani elemeknek, és ha mégis, akkor sem tudjuk a kapott morfémák nyelvtani jelentését [9], [3].

A további fejlesztések szempontjából viszont az aktuális felismerési pontosság mellett fontos, hogy a szegmentumok jelentéssel bíró egységek legyenek, mert ez teszi lehetővé, hogy további nyelvfeldolgozási szinteket építsünk a beszédfelismerő fölé. Így kipróbáltuk a hunmorph-ot [6] is a szegmentumok előállítására. A hunmorph eredetileg morfológiai elemzésére lett kialakítva, nem pedig szegmentálásra, így ehhez először némi átalakításra volt szükség, hogy a szegmentumokat is előállítsa a program. A hunmorph használatánál további problémát jelentett, hogy a program néha összekötött két külön morfémának megfelelő szegmentumot, illetve egy szóra nagyon sok alternatív elemzési megoldást kínált. Az alternatívák közül valamilyen egyszerű heurisztikával igyekeztünk kiválasztani egyet (pl.: a leghosszabb elemzés). Feltehetően ez utóbbi okok miatt a hunmorph-os szegmentálásra épülő beszédfelismerő eredményei rosszabbak voltak a morfessorra épülőénél, viszont még mindig jobbak, mint a pusztán szó alapú elemző eredményei.

Szerettük volna a szabály alapú szegmentáló gyengeségeit kiküszöbölni, mivel az volt az alapfeltevésünk, hogy a természetes morfémahatárokat használatával jobb eredményeket érhetünk el. Hogy javítsunk a hunmorph-os szegmentálás eredményein, megpróbáltuk a felajánlott alternatívák közül a statisztikailag legértelmesebbet kiválasztani. Ehhez a választáshoz a morfessorban a szegmentálás tanulására használt módszert alkalmaztuk.

2 Szegmentáló használata a beszédfelismerőben

A beszédfelismerőkben használt nyelvi modellek általában szó alapúak, vagyis a felismerés alapegysége a szó. Az agglutináló nyelvek esetén ez a megközelítés jelentős hátrányokkal jár. Mivel egy szónak rendkívül sok különböző alakja lehet, amit ez esetben mind különböző szónak kell tekintenünk, a beszédfelismerőnek egy igen nagyméretű szótárban kell keresnie. Nagy szótár esetén a felismerés során sok lehetséges megoldás közül kell választanunk, és ráadásuk az egyes szavakra jutó tanítópéldák száma is kicsi lesz.

A problémák leküzdéséhez érdemes a morfémákat választani a felismerés alapegységeként. Azonban ennek a megközelítésnek is megvannak a maga buktatói. Először is a szavakat morfémákra kell bontani, ami korántsem magától értetődő feladat. Vannak olyan szóelemző alkalmazások, amelyek a szótő mellett a szóhoz kapcsolódó morfémákat is meghatározzák, de ezek nem a beszédfelismeréshez lettek fejlesztve, és nem adják meg, hogy a szóban mely betűk (még érdekesebb, mely hangok) felelnek meg az egyes nyelvtani elemeknek. Csak ízelítőnek nézzünk néhány problémásabb esetet.

1. Táblázat: A szótő nem állítható elő szegmentumként

Szó	**Elemzés**
lenniük	van <INF><PERS><PLUR>
hass	hat <SUBJUNC-IMP><PERS<2>>
arannyal	arany <CAS<INS>>
hússzor	húsz [MULTIPL-ITER]/ADV
borókásban	boróka [ATTRIB]/ADJ<CAS<INE>>

Amikor szegmentálni szeretnénk egy szót, sokszor az eredeti szótőt nem kapjuk vissza, néha az egész szótő, gyakran csak a szótő vége módosul. Kérdés az is, hogy ha kötőhangot használunk a toldalékoláskor, azt melyik részhez kapcsoljuk.

Egy szónak gyakran több lehetséges jelentése és ezzel együtt több lehetséges szegmentálása van. Ezt a problémát valamilyen egyértelműsítés alkalmazásával lehet megoldani. Az emberek a szövegkörnyezet alapján könnyen meg tudják határozni, hogy mi az éppen megfelelő elemzés, de a számítógépnek ez bonyolultabb feladat. De nem csak a feladat nehézsége jelent gondot, hanem az is, hogy a beszédfelismeréshez használt (trigram) nyelvi modell nem alkalmas az ilyen jellegű egyértelműsítésre, ezért valamilyen egyszerűbb szabály alapján kell választanunk a különböző lehetőségek közül. A legegyszerűbb ilyen lehetőségek a leghosszabb, legrövidebb vagy a legelső elemzés választása. Ésszerű lenne a leggyakoribb megoldás választása, de nem állt rendelkezésünkre egyértelműsítő, így nem tudtunk gyakoriságot számolni a tanítókorpuszon.

Tegyük fel, hogy rendelkezésre áll a morféma alapú nyelvi modell és a hozzá megalkotott a beszédfelismerő, és feldolgozunk vele egy felvételt. Ekkor a kimeneten egy morfémasort fogunk kapni, amit még nem elég, mert mi kimenetként szavakat várunk a beszédfelismerőtől. Hogy ezt megtehessük, egy egyszerű trükköt alkalmazunk. Bevezettünk egy új szimbólumot (#), ami a szóhatárokat jelöli. A nyelvi modell építésekor ezt egyszerűen egy új morfémának tekinthetjük. Így a felismerési eredmény tartalmazza a szóhatárokat, és a szavak könnyen összerakhatók. Ennek a módszernek annyi hátránya van, hogy a szóhatároknál csökkenti a modell kontextusérzékenységét. A hatás különösen akkor jelentős, ha a szavak átlagosan kevés morfémából épülnek fel. (A rendelkezésünkre álló korpuszban a szavankénti morféma szám 1,6 körül van.) Ha ellensúlyozni szeretnénk a hatást, akkor hosszabb kontextust kellene figyelembe vennünk, ami jelentősen növeli a szükséges számításokat, ezért ezt nem alkalmaztuk.

3 Az eredeti módszerek

3.1 Hunmorph

A hunmorph egy nyílt forráskódú morfológiai elemző. A morphdb.hu morfológiai szótár segítségével elemzi a szavakat, és előállítja azok morfológiai elemzését, vagyis

meghatározza és annotálja a szótövet és a toldalékokat. Ahhoz, hogy az elemzést használni lehessen a beszédfelismerésben, olyan módon kellett átalakítani a szoftvert, hogy a leválasztott toldalékokat az aktuális szóban szerepelő formában adja vissza. A következő példa szemlélteti a kimeneten véghezvitt változtatást.

Az „odatették” eredeti elemzése:

```
oda/PREV+tesz/VERB<PAST><PLUR><DEF>
```

És az ennek megfelelő szegmentálást is tartalmazó elemzés (szegmentumok a „{}” jelek között):

```
{odate} oda/PREV+tesz/VERB {tték} <PAST><PLUR><DEF>
```

Természetesen az elemző teljes átalakítására nem volt lehetőség, a szegmentumokat olyan formában állnak elő, ahogy az a program belső működésének leginkább megfelel. Így az eredmény néha eltér az intuitív megoldástól, és elég sajátosnak tűnhet.

2. Táblázat: Problémák a szegmentálással

Szó	**Szegmentálás**	**Hunmorph kimenete**
kossal	ko-ssal	{ko} kos /NOUN {ssal} <CAS<INS>>
ontással	ont-á-ssal	{ont} ont {á} /VERB[GERUND]/NOUN {ssal} <CAS<INS>>
állásomban	áll-ás-omban	{áll} áll {ás} /VERB[GERUND]/NOUN {omban} <POSS<1>><CAS<INE>>
elfordította	el-fordít-otta	{el} el/PREV+ {fordít} + fordít {otta} /VERB<PAST><DEF>

A 2. táblázat első két példája a szavak és toldalékok néhol furcsa szétválasztását mutatja be. Ez a probléma elő-előfordul ugyan, de a szegmentálások jelentős része jóval közelebb áll az emberi intuícióhoz (legalábbis valamelyik a felajánlott szegmentálások közül). A második két eset azt mutatja be, hogy a program gyakran nem vág szét olyan toldalékokat, amiket még tovább lehetne darabolni. Ezekre jellemző, hogy ugyan egy szegmentumként szerepelnek a kimeneten, de a szegmentum után megtalálható mindkét részhez tartozó nyelvtani címke. Ezt a megfigyelést kihasználva egy utólagos feldolgozással e címkék jelentős részét tovább tudjuk darabolni. Ennek lényege, hogy eltároljuk az összes olyan szegmentumot, amihez csak egy címke tartozik. Nevezzük ezeket atomi szegmentumoknak. Ezek után az összetett szegmentumokhoz keresünk olyan atomi szegmentumokat, amelyek címkéje megtalálható az összetett szegmentumhoz tartozó címkék között, és a szegmentum egy részét lefedi. Ha sikerül egy szegmentumot teljesen és átlapolódásmentesen lefedni atomi szegmentumokkal, akkor a fedésnek megfelelően feldaraboljuk azt.

3.2 Morfessor

A morfessor egy statisztikai módszerek alapján működő szegmentáló program. Eredetileg finn nyelvre fejlesztették ki, mert a finnben egy szónak annyira sok alakja lehetséges, hogy egy szabály alapú morfológiai elemző elkészítése túl bonyolult lenne. A

program pusztán egy címkézetlen korpuszból képes megtanulni, hogy hogyan kell szegmentálni egy adott szöveget, és semmilyen nyelvspecifikus szabályt nem tartalmaz, ezért alkalmazása igen egyszerű volt számunkra. További előnye a szabály alapú vetélytársával szemben, hogy minden egyes szóra csak egyetlen szegmentálást ad, vagyis az egyértelműsítés problémáját kiküszöböli.

Érdemes kicsit közelebbről is megnézni, hogy milyen elvek alapján működik a morfessor, már csak azért is, hogy később jobban megérthessük a két módszer kombinálásának lényegét. A következőkben a [2]-re fogunk támaszkodni. Az „alap" morfessornak több javított változata is készül, de mi csak a legegyszerűbb verzióval fogunk részletesebben foglalkozni. Természetesen a méréseknél az összes verziót kipróbáltuk, és a morfessor eredményeként ezek közül a legjobbat közöljük.

A program feladata, hogy előállítsunk egy nyelvi modellt egy címkézetlen korpuszból. A modell pedig nem más, mint a morfémák egy halmaza és egy rajtuk értelmezett nyelvtan. Tehát egy olyan modellt szeretnénk találni felügyelet nélküli tanulással, aminek segítségével tömören le tudjuk írni a tanulókorpuszt és ráadásul a morfémakészletünk is tömör marad. A probléma megfogalmazható egy maximális a poszteriori paraméterbecslési feladatként (MAP):

$$\begin{gathered}\arg\max_M P(M \mid korp) = \arg\max_M P(korp \mid M)P(M), ahol \\ P(M) = P(szótár, nyelvt)\end{gathered} \tag{1.1}$$

A MAP becslés két részből áll: a nyelvi modell valószínűségéből és a korpusz modellre vetített feltételes valószínűségének maximum likelihood becsléséből. Látható az is, hogy a modell valószínűsége a szótár és a nyelvtan együttes valószínűségével egyezik meg. Érdemes megjegyezni, hogy a képletben szereplő valószínűségek bayesi értelemben vett valószínűségek, tehát nem előfordulások valószínűségei, hanem priori hiedelmek bizonyosságát fejezik ki.

Annak a valószínűsége, hogy egy adott szótárat állítsunk elő, a szótárban szereplő morfémák együttes valószínűségével arányos, pontosabban, ha *M* a különböző morfémák száma, akkor *M!*-szorosa, mert a morfémák ennyiféle különböző sorrendben kerülhetnek a szótárba. A morfémáknak két tulajdonsága van, amelyek befolyásolhatják a szótár valószínűségét: a morfémák előfordulási gyakorisága és az őket felépítő betűsor, ami tartalmazza a morféma hosszát is. Így a szótár valószínűségére a következő képletet kapjuk, ahol $s_{\mu i}$ a μ_i morfémát reprezentáló karaktersor és $f_{\mu i}$ a morféma előfordulási gyakorisága.

$$P(szótár) = M!P(f_{\mu_1}, f_{\mu_2}, f_{\mu_3}, ...)P(s_{\mu_1}, s_{\mu_2}, s_{\mu_3}, ...) \tag{1.2}$$

Az előfordulási gyakoriságokból adódó tagot az egész szótárra globálisan tudjuk számolni. Legyen *N* az összes morféma összes előfordulásának száma.

$$P(f_{\mu_1}, f_{\mu_2}, f_{\mu_3}, ...) = 1/\binom{N-1}{M-1} = \frac{(M-1)!(N-M)!}{(N-1)!} \tag{1.3}$$

A második tag számolásához azzal az egyszerűsítő feltevéssel élünk, hogy a morfémákat alkotó karaktersorozat független a többi morfémát alkotó karaktersorozattól. Így az együttes valószínűség megegyezik a morfémák valószínűségének szorzatával. Feltesszük továbbá azt is, hogy a morfémákat alkotó karakterek valószínűségei is függetlenek egymástól, és így a morféma valószínűsége az őt alkotó karakterek valószínűségének szorzatával azonos.

$$P\left(s_{\mu_1}, s_{\mu_2}, s_{\mu_3}, \ldots\right) = \prod_{i=1}^{M} P\left(s_{\mu_i}\right), \acute{e}s$$
$$P\left(s_{\mu_i}\right) = \prod_{j=1}^{l_{\mu_i}} P\left(c_{ij}\right) \tag{1.4}$$

A morféma hosszát implicit modellezzük a már említett morfémavég szimbólum (#) bevezetésével, amit minden morféma végére odaírunk a szótárban. A „#” valószínűségéből könnyen számolható egy *l* hosszú morféma valószínűsége, mivel a morféma először tartalmaz *l* „#”-től különböző szimbólumot végül pedig egy „#”-t. Ennek valószínűsége egy szimpla exponenciális eloszlásból adódik.

$$P(l) = \left[1 - P(\#)\right]^{l} P(\#) \tag{1.5}$$

A nyelvtan azt határozza meg, hogyan lehet az egyes nyelvi elemeket kombinálni. A legegyszerűbb morfessor egyáltalán nem veszi figyelembe a kontextus az elemek kombinációjánál, ezért nem is igazán lehet nyelvtanról beszélni, vagyis a szótár és a nyelvtan együttes valószínűsége a szótár valószínűségére redukálódik. Ez azt is jelenti, hogy egy morféma ugyanolyan valószínűséggel fordulhat elő bármilyen morféma után, vagy a szó elején és végén. A morféma gyakorisága tehát egy egyszerű maximum likelihood becslés. Ha f_μ a μ morféma előfordulási gyakorisága, akkor ez így írható le:

$$P(\mu_i) = \frac{f_{\mu_i}}{N} = \frac{f_{\mu_i}}{\sum_{j=1}^{M} f_{\mu_j}}. \tag{1.6}$$

A korpusz összes szava felbontható a szótárban található morfémákra, gyakran több felbontás is lehetséges. A MAP modellt használva mindig a legvalószínűbb felbontást fogjuk választani. A korpusz valószínűsége egy adott nyelvi modell esetén a következő, ahol *W* a szavak száma és n_j a *j.* szóban található morfémák száma, a morfémák előfordulási valószínűsége pedig az (1.6) képlet alapján számolandó:

$$P(korp \mid M) = \prod_{j=1}^{W} \prod_{k=1}^{n_j} P(\mu_{jk}) \,. \tag{1.7}$$

Az eddigi képletek meghatározzák a szegmentálás tanulása során maximalizálandó függvényt. A maximum megtalálásához egy mohó algoritmust javasoltak. Induláskor a szótár a korpuszban található szavak halmaza. A tanulás során az algoritmus sorban veszi a szavakat, és megpróbálja őket különféle módon szegmentálni, majd az alternatívák közül a legnagyobb valószínűségűt tartja meg. A módszert mindaddig folytatjuk, amíg szignifikáns javulást tapasztalunk.

A keresés során nem közvetlenül a valószínűségeket számoljuk, hanem azok (kódhosszként is értelmezhető) negatív logaritmusát, mert így a szorzás helyett összeadást tudunk alkalmazni.

Az algoritmus egy speciális adatstruktúrát használ, ahol a korpusz minden egyes szavának megvan a saját bináris vágási fája. A fában a levelek, vagyis azok az elemek, amelyek nincsenek tovább darabolva, felelnek meg a szótárban szereplő morfémáknak, és csak ők számítanak bele a kódhosszba. Minden egyes csomópontnál tároljuk az adott elem előfordulási gyakoriságát, ami megegyezik a szülők előfordulási gyakoriságának összegével. Továbbá minden csomópont csak egyszer szerepelhet ebben az adatstruktúrában, tehát ha két szó vágási fájában egy részfa átlapolódik, akkor azt a részfát csak egyszer tároljuk el.

Az algoritmus fő művelete a csomópont újradarabolása. Ez egy rekurzív művelet, amely először megkeresi, hogy az adott csomópontot hol kell kettévágni ahhoz, hogy a legjobb értékeket kapjuk, majd a kapott két csomópontot tovább darabolja. Ezt mindaddig folytatja, amíg lehet olyan vágást találni, amely csökkenti a szótár teljes kódhosszát. Tehát ahogy említettük, induláskor maguk a szavak a morfémák, majd véletlen sorrendben végigmegyünk az összes szón és újradaraboljuk őket. Ha végeztünk az összes szóval, akkor kezdjük elölről, mindaddig, amíg egy megadott korlátnál jobban tudjuk csökkenteni a globális kódhosszt.

```
újravágás(csp)
// EGY CSOMÓPONT EGY SZÓNAK VAGY EGY DARABJÁNAK FELEL MEG
// 1. ELTÁVOLÍTJUK AZ AKTUÁLIS REPREZENTÁCIÓJÁT A CSP.-NEK
if csp megtalálható a struktúrában then
   for all a csp-ben gyökerez• részfa elemeire (m) do
      csökkentsd számláló(m) számláló(csp)-vel
      if m levél vagyis egy morféma then
         csökkentsd az L(korp | M) -et és L(f_μ1, f_μ2, ...) -et
      if számláló(m) = 0 then
        m eltávolítása
        if m levél then
           csökkentsd L(s_μ1, s_μ2, ...) -et
```

```
// EL•SZÖR MEGPRÓBÁLJUK AZ EGÉSZ SZÓT EGY MORFÉMÁNAK TEKINTENI
csp-t levélként visszarakjuk a struktúrába számló(csp)-vel
növeld L(korp|M)-et és L(f_μ1, f_μ2, ...)-et
növeld L(s_μ1, s_μ2, ...)-et
legjobb megoldás ← [L(korp|M); csp]

// PRÓBÁLJUK KI A CSP. ÖSSZES LEHETSÉGES KÉT RÉSZRE VÁGÁSÁT
Távolítsd el csp-t L(M|korp)-ból, de hagyjuk az adatstruktúrában
Mentsük el az állapotot X-be
for all pre + suf = csp do
   for m in [pre; suf] do
      if az adatstruktúra tartalmazza m-et then
         for all n, m-ben gyökerez• részfa csomópontra do
            növeld számláló(n) számláló(m)-mel
            if n levél then
               növeld L(korp|M)-et és L(f_μ1, f_μ2, ...)-et
      else
         add hozzá m-et az adatstruktúrához számláló(m)-mel
         növeld L(korp|M)-et és L(f_μ1, f_μ2, ...)-et
         növeld L(s_μ1, s_μ2, ...)-et
   if L(M|korp) < legjobb megoldás then
      legjobb megoldás ← [L(korp|M); pre; suf]
   állítsuk vissza az X-be elmentett állapotot

// VÁLASSZUK KI A LEGJOBB VÁGÁST VAGY „NEM VÁGÁST"
Állítsuk az adatstruktúrát és az L(M|korp)-t a „legjobb megoldás"-nak megfelel•en
if pre + suf = m vágás történt then
   pre és suf szül•jének állítsuk be m-et

// FOLYTASSUK A VÁGÁST REKURZÍVAN
újravágás(pre)
újravágás(suf)
```

3 A hibrid megoldás

Amint az első kísérletek után kiderült, hogy a beszédfelismerésben a morfessorral készített szegmentálás jobb eredményt ad, elkezdtünk gondolkodni, hogyan lehet egyesíteni a két módszert. Arra gyanakodtunk, hogy a hunmorph gyengesége abból adódik, hogy nincs megfelelő módszerünk az általa generált lehetőségek közül a legjobbat kiválasztani. De valójában már azzal is megelégedtünk volna, ha a hunmorph segítségével hasonló eredményeket tudunk elérni, mint a morfessorral, mert a szabály alapú rendszer esetén, mintegy melléktermékként megkapjuk a szó nyelvtani elemzését is, amiből reményeink szerint későbbiekben felhasználhatunk. Ezért egy hibrid megoldás elkészítése mellett döntöttünk. Az alapötlet ehhez igen egyszerű: használjuk a morfessor jól bevált statisztikai modelljét a hunmorph által felkínált szegmentálások közül a legmegfelelőbb kiválasztására.

A cél érdekében némileg változtattunk a morfessor kódján. Módosításunk lényege, hogy a szavak újravágásakor a költségfüggvényt kiszámoljuk az összes hunmorph által javasolt alternatívára, de nem az összes lehetséges vágásra, és ez alapján választjuk ki a „legjobbat”. Ehhez természetesen először le kell futtatnunk a hunmorphot a korpuszon, és az eredményt meg kell adnunk a morfessornak. Ezek után egy szó újravágásánál már csak a meglévő variációkat próbáljuk ki, és ezek közül választjuk a legjobbat. Ennek érdekében a programot tulajdonképpen csak a pszeudokódban szürkével megjelölt helyeken kellett módosítani, valamint a rekurziónál nyilván kell tartani, hogy éppen melyik hunmorph variációt vizsgáljuk. Tehát az első szürke résznél annyit változtatunk, hogy nem az összes lehetséges vágást vizsgáljuk, hanem csak azokat a vágásokat, amelyek megfelelnek az adott hunmorph elemzésnek. A második szürke soron a módosítás lényege, hogy nem engedjük meg azt, hogy ne vágjunk szét egy morfémát, ha a hunmorph még tovább bontaná azt. Sőt a vágást akkor is elvégezzük, ha ez növeli a kódhosszt..

Fontosnak tartjuk még jegyezni a hibrid megoldással kapcsolatban, hogy ez a morfessor használatát is módosítja. A morfessor működése ugyanis eredetileg két ciklusra bontható. Az első ciklusban egy nagyméretű korpusz alapján a program megtanulja, hogyan kell elemezni a szavakat, és elmenti az elemzéshez használt modellt. Ezek után a program működéséhez nincs szükség a korpuszra, hanem az elmentett modell alapján akár egyes szavakat külön-külön is képes elemezni. Ezzel szemben a hibrid modell csak tanuló üzemmódban használható, mivel a tanuló algoritmust használjuk fel a hunmorphos variációk szűrésére. Persze a morfessor továbbra is elmenti a modellt, és ez alapján működőképes lesz, de nincs garancia rá, hogy az ismeretlen szavakhoz olyan szegmentálást rendel, amit a hunmorph is felkínálna, illetve ilyen esetekben nem adható a szegmentálás mellé morfológiai elemzés. Ez a megkötés számunkra szerencsére nem akadály, mivel csak a beszédfelismerésben használt nyelvi modell építéséhez szeretnénk használni, ami egy offline folyamat, így taníthatjuk az egész korpuszon a morfessort.

4 Kísérleti eredmények

A felismerési teszteket a magyar MALACH korpuszon végeztük [3]. A MALACH (Multilingual Access to Large Spoken Archives) projekt célja, hogy hatékony hozzáférést biztosítson a Holokauszt túlélőinek beszámolóihoz. Az interjúkat 32 nyelven vették fel, és egy jelentős részük, mintegy 2000 órányi hanganyag magyar nyelvű. Ebből a 2000 órából eddig mindössze 31 órányit rögzítettek írásban is. A kísérleteket a lejegyzett részen végeztük. A tanításhoz 26 órányi, a teszteléshez a maradék 5 órányi anyagot használtuk fel (további részletek az akusztikus modell-tanításról a [3]-ban találhatók). A trigram nyelvi modellt a tanítókészlet 200 ezer szava alapján építettük a SRILM eszköz segítségével [10].
Röviden a következő eredményeket kaptuk. Ezek alapján látható, hogy az új megközelítés nem csak a nyelvtani információkat őrzi meg, hanem a felismerési pontosságban is a legjobb (3. táblázat).

3. Táblázat: Szó- és betűhiba arány (WER és LER) a MALACH korpuszon különböző szegmentálási módszereket használva. A feltüntetett hibák két különböző tesztalanyon mért hibák átlagai.

Nyelvi modell	**WER**	**LER**
Szó alapú	50,48	22,72
Szó alapú + beszélő adaptáció	45,1	18,42
Morfessor	47,58	21,26
Morfessor + beszélő adaptáció	39,77	16,11
Hunmorph + Morfessor	47,04	21
Hunmorph + Morfessor + beszélő adaptáció	**39,22**	**16,09**

Összefoglalás

A leírt kísérletek körüljárták a problémát, hogy hogyan lehet előállítani olyan szegmentálást, ami eredményesen használható a morféma alapú beszédfelismerés során. Kipróbáltunk két különböző elven működő, már meglévő módszert. A morféma alapú beszédfelismerési eredmények jobbnak bizonyultak a szó alapúnál, igazolva feltevésünket, hogy magyar nyelven a szó alapú beszédfelismerés nem optimális. Az első pozitív eredmények után megpróbáltuk finomítani a szegmentáló módszerünket. Ehhez a két különböző módszer kombinációjával próbálkoztunk. A megalkotott kombinációnk sikeresnek bizonyult, mert a hibrid módszer segítségével kaptuk a legjobb felismerési eredményeket, és a szabály alapú szegmentáló által előállított nyelvtani elemzés is a rendelkezésünkre áll a szegmentálás mellett.

A jövőben több érdekes folytatása is lehetséges a munkának. Egyrészt ha rendelkezésünkre állna egy egyértelműsítő, ami kiválasztja a korpuszban egy szónak az adott helyen legmegfelelőbb morfológiai elemzését, akkor megpróbálhatjuk a szabály alapú szegmentálásból mindig a legvalószínűbbet kiválasztani és ebből állítani elő a

nyelvi modellt. Egy másik kutatási irány annak vizsgálata, hogy a meglévő morfológiai elemzés segítségével hogyan javíthatóak a beszédfelismerési eredmények.

Köszönetnyilvánítás

Köszönet szeretnénk mondani a MOKK munkatársainak, különösen Halácsy Péternek a hunmorph eszközzel kapcsolatos technikai segítségért, valamint a beszédfelismerős – szövegfeldolgozós – számítógépes nyelvész kutatók eszmecseréjének aktív előmozdításáért, amelynek révén - reményeink szerint – nem csak e cikk társszerzői gazdagodtak.
A kutatást – részben – az NKFP-2/034/2004-es projekt keretében az NKTH támogatta.

Bibliográfia

1. Hirsimäki, T., Creutz, M., Siivola, V., Kurimo, M., Pylkkönen, J., and Virpioja, S., "Unlimited vocabulary speech recognition with morph language models applied to Finnish.", Computer, Speech and Language, Vol. 20, Issue 4 (2006) 515–541
2. Creutz, M. and Lagus, K., "Unsupervised Morpheme Segmentation and Morphology Induction from Text Corpora Using Morfessor 1.0.", Publications in Computer and Information Science, Report A81, Helsinki University of Technology, March, (2005)
3. Mihajlik, P., Fegyó, T., Nemeth B., Tüske Z., and Trón V., " Towards Automatic Transcription of Large Spoken Archives in Agglutinating Languages – Hungarian ASR for the MALACH Project TSD 2007, Pilsen
4. Kwon, O.–W. and Park, J., "Korean large vocabulary continuous speech recognition with morpheme-based recognition units," Speech Communication, Vol. 39, Nos. 3–4 (2003) 287–300
5. Hirsimäki, T., Creutz, M., Siivola, V., Kurimo, M., Pylkkönen, J., and Virpioja, S., "Unlimited vocabulary speech recognition with morph language models applied to Finnish.", Computer, Speech and Language, Vol. 20, Issue 4 (2006) 515–541
6. Trón Viktor, Halácsy Péter, Rebrus Péter, Rung András, Simon Eszter, és Vajda Péter: morphdb.hu: magyar morfológiai nyelvtan és szótári adatbázis. III. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY-05), pp. 169–179, Szeged, 2005.
7. Szarvas, Máté: "Efficient large vocabulary continuous speech recognition using weighted finite-state transducers - The development of a Hungarian dictation system" PhD. Thesis, TITECH, Tokyo, 2003
8. Vicsi Klára at al.„Középszótáras, folyamatos beszédfelismerő rendszer fejelsztési tapasztalatai" III. Magyar Számítógépes Nyelvészeti Konferencia, Szeged, (p. 348-359), 2005
9. Péter Mihajlik, Tibor Fegyó, Zoltán Tüske, and Pavel Ircing, „A Morpho-graphemic Approach for the Recognition of Spontaneous Speech in Agglutinative Languages – like Hungarian" In Proc. of INTERSPEECH-2007, pp.: 1497 – 1500, Antwerpen, Belgium, August 27-31, 2007
10. Stolcke, A., "SRILM – an extensible language modeling toolkit", In Proc. Intl. Conf. on Spoken Language Processing, Denver (2002) 901–904
11. Viktor Trón, László Németh, Péter Halácsy, András Kornai, György Gyepesi, and Dániel Varga, „Hunmorph: open source word analysis", In: Proceeding of ACL., 2005