Magyar szövegek természetes nyelvi előfeldolgozása

Miháczi András, Németh László, Rácz Miklós

Mihaczi.Andras.Janos@stud.u-szeged.hu
NemethL@gyorsposta.hu
Racz.Miklos@stud.u-szeged.hu
Szegedi Tudományegyetem, Informatikai Tanszékcsoport

Kivonat. A természetes nyelvi szövegek előfeldolgozásának feladata a szöveg mondatokra, szavakra bontása, tokenizálása (tokennek nevezzük a legkisebb önálló jelentéssel bíró szövegegységet). Ehhez szorosan kapcsolódik az úgy nevezett nyílt tokenosztályokba tartozó egyes tokenek felismerése. Ezek olyan tokenek, amelyekben speciális (írás)jelek vagy szóközök vannak. Az előfeldolgozás része a tulajdonnevek felismerése is, hiszen itt nagyméretű, tulajdonneveket tartalmazó, szótárakat kell használni A feladatok megoldására kipróbáltunk reguláris kifejezések alapján generált automatát, valamint döntésifa-tanuló algoritmusok által tanult szabályokat.

1. Bevezetés

A természetes nyelvi szövegek feldolgozásának kiindulópontja egy egyszerű, formázatlan szöveg. Az előfeldolgozás feladata a szöveg mondatokra, szavakra bontása, tokenizálása.

A szöveg mondatokra bontása az esetek nagy részében egyszerű feladat, a probléma a lehetséges mondatzáró írásjelek adott környezetben való értelmezése, ugyanis vannak olyan esetek, ahol nem mondathatárt jelölnek.

A szóhatárok a legtöbb esetben egyértelműek, hiszen a szavakat szóközök választják el. A feladat itt is a különleges esetek kezelése, előfordul, hogy az írásjelek hozzátartoznak egy szóhoz. Ehhez szorosan kapcsolódik az úgy nevezett nyílt tokenosztályokba tartozó egyes tokenek felismerése.

A tulajdonnevek felismerése azért az előfeldolgozás része, mert befolyásolja a szegmentálást, ugyanis egy tulajdonnév több szóból is állhat. Ez a folyamat általában szótárak használatával történik, illetve meghatározhatók bizonyos szabályok, melyek alapján összeállnak a tulajdonnevek.

A feladatok megoldására több módszert is kipróbáltunk. A munkában nagy segítséget jelentett az annotált Szeged Korpusz[1]. A korpusz alapján a mondat- és szószegmentálásra egyaránt kipróbáltunk döntésifa-tanuló algoritmusokat. A tulajdonnevek felismerése nagy méretű szótárak használatával történt. A

[1] Ezúton szeretnénk köszönetet nyilvánítani a Szeged Korpusz készítőinek, az SZTE Informatikai Tanszékcsoport, a MorphoLogic Kft. és az MTA Nyelvtudományi Intézet munkatársainak szakmai segítségükért, valamint az általuk gyűjtött adatbázisokért.

tulajdonnévszótár összeállításában is segítségünkre volt a korpusz. Ezen kívül felhasználtunk célszótárakat. Az eredmények tesztelése ugyancsak a Szeged Korpusz alapján történt [2].

2. Szegmentálás

2.1. Mondat és szószegmentálás

A szövegfeldolgozás első lépése a szöveg mondatokra és szavakra bontása. A szavak és írásjelek jelentik a szöveg alapegységeit, ennél részletesebb felbontással nem foglalkozunk. A feldolgozás során ezekhez rendelünk attribútumokat (például szófajok), vagy vonjuk őket össze nagyobb csoportokba (például főnévi szerkezetek). A szavakon itt nem csak a hétköznapi értelemben vett szavakat értjük, ide tartoznak a nyílt tokenosztályok elemei, illetve a több tagból álló tulajdonnevek is.

A mondatokra bontás szintén fontos, ez határozza meg az összetartozó szavakat. Ez főleg a szöveg szemantikai feldolgozásánál számít, de jelenthet információt például a szófaji egyértelműsítés számára is.

Az is fontos, hogy a két fázis milyen sorrendben követi egymást, hiszen fel lehet használni az előző rész eredményeit. (A mondatokat szavak alapján bonthatjuk, illetve a tokenek meghatározásánál segíthet, hogy nem léphetjük át a mondathatárt.)

2.2. Nyílt tokenosztályok

Egy úgynevezett nyílt tokenosztály elemeit valamilyen meghatározott közös szintaktikai tulajdonság megléte sorolja egy osztályba. Ezek az osztályok formális nyelvtannal definiálható potenciálisan végtelen elemszámú halmazok, azaz a szabályok ismételt alkalmazásával – elvileg – végtelen sok tokent állíthatunk elő. Ilyen lehet a személynevek, az időtartamot jelentő kifejezések osztálya, a webcímek vagy az azonosítók. Az ilyen szavak két csoportba oszthatók. Vannak, amelyek a már meglévő MSD kódokkal jelölhetők, a többi kódolására új MSD kódok kerültek bevezetésre. Ezek rendszerezését Dr. Bibok Károly végezte el.

2.3. Tulajdonnevek

A tulajdonnevek halmaza részhalmaza a főnevek halmazának. Automatikus felismerésük azért nehéz, mert a szövegben előfordulhat akár a világ bármely tájáról vett tulajdonnév is. Ezekre adhatunk bizonyos egyszerű szabályokat, de a döntő szerepet a tulajdonnevek felsorolása jelenti. Ezért a tulajdonnév-felismerésnél döntő szerepet játszik a szótár használata. Nehézséget okoz, hogy a tulajdonnevek száma dinamikusan nő.

3. Szakérői tudást felhasználó feldolgozás

A következő fejezetben olyan módszereket mutatunk be, melyek alapja a szakértői tudás, valamint a szakértők által megfogalmazott szabályok alkalmazása.

3.1. Huntoken

A Huntoken szövegfeldolgozó program a bemeneti szöveget mondatokra és szavakra bontja, illetve a nyílt tokenosztályokba eső szavak egy részét felismeri, és a megfelelő MSD kóddal jelöli. A program bemenete ISO-8859-2 karakterkódolású szöveges állomány, amely a HTML 4 szabvány ISO-8859-1-es karakterentitásait is tartalmazhatja. A program kimenete a Szeged Korpusz készítése során használt XML-alapú formátum.

A Huntoken program csőbe köthető szűrőprogramokból áll, amelyek a GNU Flex lexikaielemző-generátorral készültek. A csővezetéket a huntoken parancs indítja el. A csőbe kötött szűrőprogramok szerepét a következő bekezdések foglalják össze:

A hun_clean szűrő normalizálja a bemenő szöveges állományokat a következő fontosabb műveletek elvégzésével: ismétlődő szóköz értékű (későbbiekben szóköz) karakterek törlése, ismétlődő üres sorok és közbeékelt szóközök törlése, sor eleji és sor végi szóközök törlése, nem törő szóközök szóközzé alakítása, az összes ISO-8859-2-ben szereplő ISO-8859-1-es entitás karakterré alakítása (például: á → á).

A hun_sentence szűrő <s> nyitó- és </s> zárócímke közé zárja a felismert mondatokat, vagyis elvégzi a mondatra bontást. A mondatra bontáshoz viszonylag kevés számú Flex szabály lett megadva (<10), de a szabályok között erősen összetettek is akadnak.

A hun_abbrev program ismert rövidítések, és más beépített szabályok alapján felülbírálja, és szükség esetén módosítja a hun_sentence által megállapított mondathatárokat, valamint a hun_token által megállapított szóhatárokat. Hasonló számú szabályt tartalmaz, mint a hun_sentence szűrő. Összehasonlításul Aberdeen és munkatársai több mint 100 szabályt alkalmaztak mondatra bontó Flex alkalmazásukban [5].

A hun_sentence és hun_abbrev páros a Szeged Korpusz szövegein 1% mondatra bontási hibát követ el. Kifinomultabb módszerek felhasználásával nagyobb mértékű javulás lenne elképzelhető. Angol nyelvű szövegeken, szófaji és egyéb információk felhasználásával a 0,2% hibahatárt sikerült megközelíteni a mondatra bontásban [3].

A hun_token a legnagyobb és a legösszetettebb szűrő. A szóra és írásjelekre bontás mellett a nyitott tokenosztályokba eső szavak jelentős részét felismeri, és a megfelelő MSD kóddal látja el. A következő nyitott tokenek felismerését végzi el: toldalékmorfémák, elektronikus címek, e-mail, webhely, útvonal, fájlkiterjesztés, indexek (trade mark, registered trade mark), számok: (sport)eredmények, előjeles egész számok, időpont, dátum, pontot tartalmazó számok, százalékjelet tartalmazó számok, fokjelet tartalmazó számok, arány (SI mértékegységgel), méret × jellel, képletek, azonosítók (szabvány jelzete, telefonszám, írásmű része, ISBN kód, rendszám, egyéb kötőjellel kezdődő, vagy végződő szavak, számmal és betűvel jelölt számok

A hun_token több mint 200 szabályt tartalmaz, amelyek összehangolt működését csak a beépített tesztrendszerrel lehetett biztosítani a fejlesztés során. Minden egyes új szabály, vagy szabálymódosítás esetén is ellenőrizve lett, hogy a többi szabály tevékenysége nem módosult a változtatások hatására. A Flex nyomkövető üzemmódja pedig lehetőséget biztosított arra, hogy egy-egy bonyolult, nyitott tokenosztályba eső szó felismerését lépésről lépésre tanulmányozni lehessen. A következő példa a 640x480 típusú tokeneket ismeri föl, és az alábbi outputot generálja:

```
[0-9]+("x"|"&times";"?)([0-9]+)?

<w>640x480
<anav>
<msd><lemma>640x480</lemma>
<mscat>[Onm-sn]</mscat></msd>
</anav>
</w>
```

A teszteredmények alapján a Huntoken program szövegfeldolgozó képességei elérik, és pontosságban meg is haladják a kézi szövegfeldolgozását.

3.2. Tulajdonnév felismerés

A tulajdonnevek szövegben való megtalálásához szükséges a felismerő nagyfokú lexikai tudása. Tehát egy ilyen programnak nagy méretű, tulajdonneveket tartalmazó szótárat kell kezelnie.

A mi esetünkben kézenfekvőnek látszott, hogy jelentős mennyiségű tulajdonnevet nyerhetünk az annotált Szeged Korpuszból. Első megközelítésben kigyűjtöttük a korpuszból a tulajdonnevek összes előforduló szóalakját. Mindegyikhez hozzárendeltük a megfelelő szótövet, valamint a szófaját meghatározó MSD kódot. Ez a kiindulási szótár jó alapot jelent a természetes nyelvi szövegekben való tulajdonnév felismeréshez. Az így kapott szótár 18 286 különböző szóalakot tartalmaz. A különböző szótövek száma 14 027 volt. [2]

A tulajdonnevek szótárban való tárolására két lehetséges megoldás mutatkozik. Az egyik szerint a szótárba eltárolásra kerülnének a tulajdonnevek ragozás szempontjából vett összes szóalakja. A magyar nyelv erősen agglutináló jellegű, vagyis a különböző nyelvtani funkciókat legtöbbször toldalékokkal (ragokkal, jelekkel, képzőkkel) fejezi ki, azaz egy adott tulajdonnévnek akár több száz ragozott szóalakja is lehet. Ez nagymértékben megnöveli a szótár méretét, és így a szótárban való keresés is hosszabb időt vesz igénybe. A másik lehetőség szerint a szótárba csak a tulajdonnevek alanyi esetű, azaz ragozatlan alakját vesszük fel. Ekkor a szótárban való keresés a szótár kisebb mérete miatt gyorsabban megvalósítható, viszont ekkor a ragozás problémáját bizonyos toldalékolási szabályok megadásával kell kiküszöbölnünk. Ezen szabályokhoz fel kell venni a főnevekhez kapcsolódó lehetséges toldalékokat (például: -ban/ben, -on/en/ön, -nak/nek). A toldalékok vizsgálatának problémája nagyon összetett, nézzük néhány összetevőjét: A toldalékok szótőhöz való kapcsolásakor figyelembe kell venni például a hangrend szerinti

illeszkedést, a mássalhangzók hasonulását, kötőhangok közbeékelését. Ilyen és ehhez hasonló szabályok sokasága fogalmazható meg a toldalék vizsgálatával kapcsolatban, különösen a nem magyar tulajdonnevek esetén.

A tulajdonnév felismerési fázist a természetes nyelvi szöveg feldolgozási folyamatában a szöveg mondatokra és szavakra bontása után valósítottuk meg. Tehát a feladat az, hogy az egymás után következő szavakról megállapítsuk, azok tulajdonnevek, vagy sem. Beszélhetünk egy-, illetve többtagú tulajdonnevekről. Az egytagúak egy szóból állnak, a többtagúak több, szóközzel elválasztott szóból állnak. Többtagú tulajdonnevek esetén előfordulhat, hogy a szavak között írásjelek is vannak. A tulajdonnevek jelölésénél a többtagúakat összevonjuk, azaz a feldolgozás későbbi fázisai számára azok már egy szóként jelennek meg. A tulajdonnevekhez a felismeréskor szótövet és szófajának, valamint toldalékolásának meghatározásához MSD kódot rendelünk. Egy tulajdonnév TEI XML jelölése:

```
<w>Magyar Hanglemezkiadók Szövetségét
<ana>
<msd><lemma>Magyar Hanglemezkiadók Szövetsége
</lemma><mscat>[Np-sa]</mscat></msd>
</ana>
</w>
```

Tekintve a lehetséges tulajdonnevek nagy számát, nincs lehetőségünk minden tulajdonnevet felvenni a szótárba. Többtagú tulajdonnevek esetén az egyes tagok kombinációjával rengeteg tulajdonnevet ismerhetünk fel. Például egy személynév a legtöbb esetben egy vezetéknévből és egy keresztnévből áll. Az összes személynév eltárolására nincs módunk, de fölvehetjük egy listába a keresztneveket és a vezetékneveket. Így egy szabály alkalmazásával, miszerint egy személynév vezetéknévből és keresztnévből áll, már a legtöbb személynevet föl tudjuk ismerni.

Tulajdonnevek felismerésére alkalmaztunk, reguláris kifejezésekkel megadott szabályokat, valamint célszótárakat. Ilyen célszótárban vannak például a keresztnevek, vezetéknevek, helységnevek, nagyvárosok nevei, cég- és intézményutótagok (kft., rt., Iskola, Tudományegyetem, stb.).

4. Gépi tanulás alkalmazása szegmentálásra

A mondat és szószegmentálásra egyaránt kipróbáltunk döntésifa-tanulást. A tréning és tanulóadatok előállításában fontos szerepe volt a Szeged Korpusznak. Mindkét problémánál az (írás)jelek környezetének vizsgálata a kritikus feladat. A mondat- és szószegmentálásnál is azt a döntést vizsgáltuk, hogy bizonyos karaktereknél illetve tokeneknél van-e mondat-, vagy szóhatár.

A legnehezebb a tréning adatok megfelelő kiválogatása volt. Az esetek túlnyomó többségében nagyon kevés, nagyon egyszerű szabály érvényesül (például mondat végi pont, és utána nagybetű), ezekkel általában el lehet már érni a 90%-os pontosságot. A tanulás során azonban annyira elnyomják a speciális eseteket, hogy azokat az algoritmus zajnak gondolja. Ezen problémákat azzal próbáltuk kivédeni, hogy a

triviális eseteket kihagytuk (például szószegmentálásnál a csupa betűből álló szavakat, ahol nem jelent problémát a szegmentálás).

A csoportosításra két módszert is kipróbáltunk, az egyiknél minden egyes karaktert külön vizsgáltunk, a másiknál próbáltunk őket csoportosítani. Utóbbi esetben betűsorozatokat, számsorozatokat egy egységként kezeltük, figyelembe véve például, hogy a token kezdőbetűje kis vagy nagybetű. Az írásjelek minden esetben darabonként külön csoportot képeztek. Az attribútumok minden esetben a kérdéses és az azt körülvevő tokenek voltak. A környezet mérete nem volt számottevő befolyással az eredményre, a döntési fákban és a belőlük képzett szabályokban csak ritkán fordultak elő a vizsgált, az azt követő, illetve az azt megelőző tokenektől különböző token.

A döntési-fa tanulásra Ross Quinlan C 4.5 algoritmusát használtunk [4], kipróbálva a paraméterek különböző variációit (attribútumok csoportosítása, iterált faépítés, fa vágása, szabályok, stb.).

Az eredmények első megközelítésben jók lettek, a hiba összességében 0,5 és 2% között mozgott. Ez jobb, mint ha csak a triviális szabályokat alkalmaznánk. A nagy döntési fákból csak kevés használható szabály született, ezek száma 7 és 15 között mozgott a tanulási paraméterektől függően. Egy részük triviális (például mondatzáró írásjel és nagybetűs szó esetén van mondathatár), a többi nyelvészetileg nem értelmezhető, a speciális eseteket kezelő szabály.

5. Jövőbeni tervek

A szegmentálással kapcsolatos jövőbeli feladatok között szerepel a tudásalapú rendszerek továbbfejlesztése és a tanuló rendszereknél leírt problémák (tréning adatbázis megfelelő összeválogatása) további vizsgálata.

A tulajdonnév felismerés kapcsán a ragozott alakok pontos felismerése a cél. Ebben segítséget jelent a Szeged Korpuszból kigyűjtött lehetséges toldalékok listája.

6. Irodalomjegyzék

[1] Alexin, Z., Csirik, J., Gyimóthy, T., Bibok K., Hatvani, Cs., Prószéky, G., Tihanyi, L.: Manually Annotated Hungarian Corpus in Proc. of the Research Note Sessions of the 10th Conference of the European Chapter of the Association for Computational Linguistics EACL'03, Budapest, Hungary 15-17 April, pp. 53-56 (2003)

[2] http://www.inf.u-szeged.hu/lll

[3] Mikheev: Periods, Capitalized Words, etc., Computational Linguistics, Vol. 28., 3., 2002, 289 o.

[4] J.R. Quinlan: C4.5: Programs for Machine Learning, Morgan Kaufmann, 1993.

[5] Aberdeen, John S., John D. Burger, David S. Day, Lynette Hirschman, Patricia Robinson, and Marc Vilain. 1995. Mitre: Description of the alembic system used for MUC-6. In Proceedings of the Sixth Message Understanding Conference (MUC-6), Columbia, Maryland, November. Morgan Kaufmann.