Hócza András

A magyar nyelv automatikus szintaktikai elemzése szabályalapú gépi tanulási technikák alkalmazásával

Doktori értekezés

Témavezető: *Gyimóthy Tibor, PhD*

Szegedi Tudományegyetem TTIK, Matematika- és Számítástudományok Doktori Iskola

Szeged, 2008

Tartalomjegyzék

Előszó

Jelen értekezés témája a szintaktikai elemzés, melynek gyakorlati megvalósítása és alkalmazása magyar nyelvre történt. A szerző módszereiben szabályalapú gépi tanulási technikákat alkalmazott, melyek segítségével egy elemzett korpuszból kinyerhető információk felhasználásával szintaktikai elemzésre alkalmazható modell építhető. A szabályalapú reprezentáció ember számára olvasható módon tárolja a megszerzett ismereteket így lehetőséget biztosít a tudásbázis karbantartására és így szakértői tudással történő kiegészítésre.

A természetes nyelvek jelenségeinek ábrázolása kihívást jelent a számítógépes nyelvészet számára, ezen belül a magyar nyelv a szabad szórend és a ragozott szóalakok nagy száma miatt a nehezebben elemezhető nyelvek közé sorolható. A generatív nyelvtanok alkalmazásai megjelenésük idején ígéretes lehetőségnek tűntek, mivel ezeket hatékony algoritmusokkal lehet elemezni, különösen a környezetfüggő és a reguláris nyelvtanok esetén. Azonban hamarosan megjelentek cáfolatok, ellenpéldák, melyek azt mutatták, hogy ezek a nyelvosztályok nem alkalmasak a természetes nyelvek bizonyos jelenségeinek ábrázolása. Napjainkra a generatív megközelítés háttérbe szorult, ezeket felváltották olyan nyelvelméletek és formalizmusok, melyekben a nyelvi jelenségek minél pontosabb leírása került előtérbe a nyelv generálása helyett.

A értekezés beszámol a szerző által elért eredményekről, egy új formalizmus, a faminta nyelvtan és egy új mintatanuló algoritmus, az RGLearn kifejlesztéséről és ezek alkalmazásairól. A szerző kidolgozott egy komplex korpusz alapú szintaktikai elemző módszert, ami a korpuszból gépi tanulási technikákkal kinyert famintákat tartalmazó modell segítségével végzi el a szintaktikai elemzést. Az ezzel megvalósított felszíni és teljes szintaktikai elemzőt beépítette információkinyerő és gépi fordító rendszerekbe.

Köszönettel tartozom Gyimóthy Tibornak, témavezetőmnek, Kocsor Andrásnak és Alexin Zoltánnak a dolgozat elkészítésében nyújtott segítségükért és tudományos projektek vezetőiként a kutatási tevékenységemhez adott hasznos tanácsaikért, valamint Csirik Jánosnak az Informatikai Tanszékcsoport vezetőjének, szakmai tanácsaiért és a kutatási feltételek megteremtéséért.

Köszönettel tartozom az SZTE Nyelvtechnológiai Csoport munkatársainak, elsősorban Csendes Dórának, Hatvani Csabának és Kovács Kornélnak, akik tevékenységükkel és ötleteikkel nagymértékben hozzájárultak az elért eredményeimhez.

Hálás vagyok családomnak a belém vetett bizalomért és a dolgozat elkészítése alatt irányomban tanúsított türelmükért és megértésükért.

1. Bevezetés

A szintaxis a görög eredetű szüntakszisz (elrendezés) szóból származik. A szintaxis feladata a mondattan, azaz a mondatok szerkezetének leírása. A mondatok értelmezése során megfigyelhetjük, hogy az egyes szavak, illetve szócsoportok összetartoznak, a mondaton belül önálló egységet alkotnak, míg másokra ez nem igaz. A összetartozó szócsoportok egyik jellemző tulajdonsága a kicserélhetőség, azaz tudunk találni olyan példamondatot melyben az egyező kategóriájú szavak illetve szócsoportok egymás helyébe írhatók, például:

*Az asztalon van **terítő**.*

*Az asztalon van **egy könyv**.*

*Az asztalon van **a vasárnapi ebéd**.*

Az összetartozó szócsoportokat frázisoknak is nevezzük és ezeket kategóriákba soroljuk legfontosabb elemük, a fej alapján. Ezek alapján beszélhetünk például főnévi (NP), igei (VP), melléknévi (ADJP), határozószói (ADVP) és névutós (PP) csoportokról. Ezek egymásba ágyazottak lehetnek, azaz egy szócsoport tartalmazhat másik szócsoportot. Ez alapján a mondatok szócsoportjai egy összefüggő hierarchikus szerkezettel, fával ábrázolhatóak, melynek levelei a szavak, csúcsai pedig a szócsoportok címkéi. A fa gyökerénél a mondat (S) szimbólum szerepel.

Egy természetes nyelv milliós nagyságrendben tartalmaz szavakat. Ragozó nyelvek esetén, mint például a magyar nyelv, a sok rag és jel miatt ez a szám elérheti a 10 milliót is. Az Internet elterjedésével a mindenki által könnyen elérhető írott szöveges információ mennyisége drámai módon növekedett. Ennek a heterogén, soknyelvű anyagnak a bármilyen kis mértékű rendezése, feldolgozása, kivonatolása is nagy fontosságú, ezért került a természetes nyelvi feldolgozás a tudományos érdeklődés középpontjába.

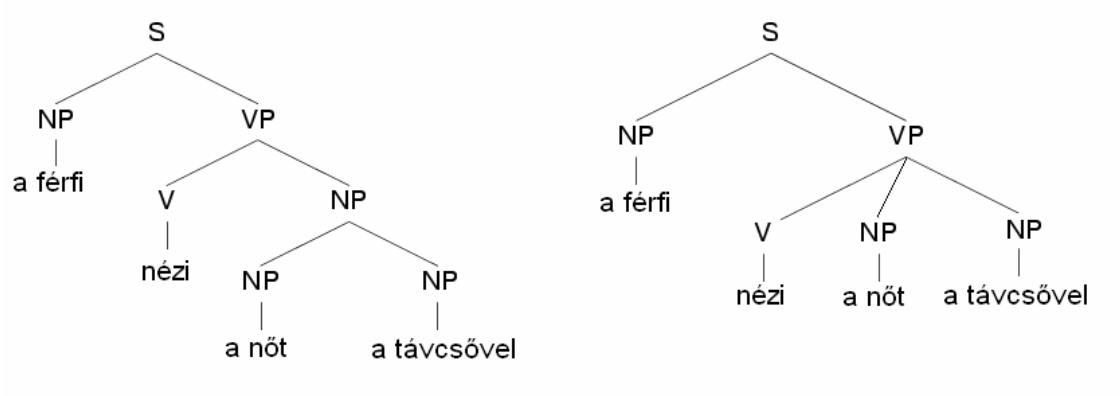
A mesterséges intelligencia hatékony eszköze a tanuló algoritmusok alkalmazása. Az ilyen algoritmusokat működtető rendszerek a kapott információk hatására növelni tudják a hatékonyságukat, jobb, pontosabb döntéseket tudnak hozni, és rendelkeznek általánosítási képességgel, azaz olyan esetekben is döntésképesek, melyekkel működésük során még nem találkoztak. A szabályalapú rendszerek egy fontos osztályt alkotnak a tanuló algoritmusok között. Elterjedésüket előnyösen befolyásolja, hogy a bennük alkalmazott szabályok az ember számára közvetlenül olvashatók és megérthetők, így lehetőséget ad további emberi (szakértői) tudás integrálására.

Egy szöveg megértésének kulcsa, a szavak közötti kapcsolatok és a nyelvi szerkezetek legszemléletesebben szabályok segítségével ábrázolhatóak. A természetes nyelv használata során a közléseinknek különféle nyelvtani szabályok szerint megfelelőnek kell lennie, hogy az általunk átadni kívánt információ mások számára is érthető legyen. Ezeket a szabályokat a közlés során nem kell értenünk, csak alkalmaznunk kell a fejünkben lévő közlési sémákat az átadandó információra. Ennek a folyamatnak a fordítottja játszódik le az élő beszéd vagy az írott szöveg megértése közben, a szövegben felismert sémák összerendezésével a fejünkben megjelenik a kapott információ egy reprezentációja. A mondat értelmezésének számítógépes megvalósítása során is az emberi szövegértés analógiáját tudjuk követni, hiszen célunk annak modellezése, vagy annak támogatása (az feldolgozott információk áttekinthető kivonatolásával). Ezért az tűnik a legalkalmasabbnak, ha a problémát szabályokon alapuló módszerek alkalmazásával oldjuk meg.

A természetes nyelvek szabályalapú modellezésére számos nyelvelmélet, formalizmus született. A korábban megjelent nyelvelméletek a generatív nyelvtanokból indultak ki, melyekre az a jellemző, hogy megpróbáljuk levezetni az elemzendő mondatot és ennek sikere esetén a kialakult levezetési szabályokból felépíthető szintaxisfa adja meg a mondat elemzését. Erre a különböző programozási nyelvek fordítása esetén hatékony algoritmusok léteznek. Azonban míg egy programozási nyelv tervezhető, az elemezhetőséget is figyelembe véve, egy természetes nyelv öntörvényű, ráadásul számos változatban előfordul ami miatt mindig mindenre vannak kivételek. A generatív nyelvtanok bevezetése után nem sokkal megjelentek a természetes nyelvekre vonatkozó cáfolatok, melyek egyre nagyobb bonyolultságú algoritmusokkal elemezhető nyelvosztályokba helyezték a természetes nyelvekre alkalmazható leírásokat. Később megjelentek a generatív szemlélettel többé kevésbé szakító nyelvelméletek és formalizmusok, melyek az elemzendő mondat levezetése helyett inkább modellezni próbálták a természetes nyelv különféle jelenségeit.

A szintaktikai elemzés során a fő problémát az információhiány jelenti. A számítógép nem a mondat értelmezése alapján dönt, hanem a morfológiai jegyekből kiindulva próbál következtetni a szemantikára. Ha nem rendelkezünk megfelelő információval valamiről, akkor általánosabb szempontok szerint kell döntenünk, ami növeli a bizonytalanságot. Ezért növekszik a számításba vehető esetek száma, azaz többértelműségek keletkeznek. Mivel azzal az előfeltételezéssel élünk, hogy a közlő szándéka szerint pontosan egy értelme van a mondatnak, döntéskényszerben vagyunk, valamilyen rangsor alapján ki kell választanunk a legjobb lehetőséget. Ehhez néha nem elegendő a rendelkezésre álló információ, ismernünk kellene a mondat kontextusát, ahhoz, hogy helyesen tudjunk dönteni. Ezt a dilemmát szemlélteti az 1.1 ábra (az eltérő szintaxis más értelmezést jelent). Ugyanakkor ahogy bővítjük az információk körét

nagyságrendekkel nő a felesleges egyéb adatok mennyisége, melyből ki kell szűrni a lényeget. Ez egy bizonyos határon túl lehetetlenné teszi, hogy az adott kérdésben elfogadható időn belül megbízható döntés születhessen.



1.1. Ábra Szintaktikai (szemantikai) többértelműség

A gépi tanulási technikák alkalmazása adatorientált megközelítést jelent, melyben az emberi szakértelem szerepe főleg az annotált adatok előállításában van. Az így előállított szintaktikailag elemzett mondatokat a gépi tanulási modell előállítására, valamint a modellen alapuló automatikus elemző algoritmus finomítására és hatékonyságának kiértékelésére lehet használni. A tapasztalatok azt mutatják, hogy az adatorientált megközelítés járhatóbb út az olyan szakértői rendszerekhez képest, melyekben kizárólag nyelvész szakértők által megkonstruált modell működik, mivel az ember korlátokba ütközik, ha fejben kell nagymennyiségű adatfeldolgozást elvégeznie. A nem adatorientált rendszerek elkészítése során a szakértőknek esetenként több száz összetevős szintaktikai feltételrendszert kell körültekintően megfogalmazniuk a helyes működés érdekében, ami sokkal nehezebb feladat, mint intuitív vagy szemantikai megfontolások alapján helyesen elvégezni mondatok annotációját. A szabályrendszer előállításában az jelenti a legnagyobb gondot, hogy sok esetben szemantikai alapon eldönthető tényeket kell szintaktikai eszközökkel megfogalmazni, ami embernek és gépnek egyaránt nehézséget okoz. Azonban, ha ezt a műveletet adatfeldolgozással helyettesíteni lehet, akkor a gép előnyben van az emberhez képest.

A szintaxis elemzés alapvető fontosságú számos természetes nyelvre megfogalmazott feladatban, mint például az ***információkinyerés*** (***Information Extraction***), a ***beszédfelismerés*** (***Speech Recognition***), a ***beszéd szöveggé konvertálása*** (***Speech-to-Text***) vagy a ***számítógépes fordítás*** (***Machine Translation***), melyek megvalósítása napjainkban még komoly kihívásnak számít. Az ilyen és hasonló természetesnyelvi alapfeladatok jó minőségű megoldására jelenthet fontos építőköveket az emberi intelligenciát megközelítő rendszerek kidolgozására felé haladva.

1.1. Az eredmények összefoglalása

A disszertációban a szerző beszámol az elmúlt években elért tudományos eredményeiről. Ezek két csoportba oszthatók, egyrészt beszélhetünk elméleti konstrukciókról és gyakorlati alkalmazásokról. Az első csoportba sorolhatóak a következő elméleti eredmények:

I/1. 2.7.2. fejezet: A szerző kidolgozott egy új formalizmust, melyet famintáknak nevezett el [Hócza04a]. A famináták mondatokon belül nagyobb, több szintű szintaktikai egységeket különítenek el, ugyanakkor hasonló szerkezetek összevonására is lehetőséget biztosítanak, így hatékony eszközt adnak az olyan ragozó és szabad szórendű nyelvek elemzéséhez, mint például a magyar nyelv.

I/2. 3.2.3. fejezet: A szerző kifejlesztett egy általános mintatanuló algoritmust, mely az RGLearn nevet kapta [Hócza04a]. Az algoritmus megkeresi a minták általánosítása és specializálása közötti optimális arányt, így famintákra alkalmazva azt a faminta halmazt, amely a maximális pontosságú szintaktikai elemzést adja.

I/3. 4.2.1. fejezet: A szerző elkészítette a chart parser szintaktikai elemző algoritmus famintákra alkalmazható változatát, mellyel bottom-up elemzés végezhető [Hócza04a].

I/4. 4. fejezet: A szerző egy komplex faminta alapú szintaktikai elemző módszerbe foglalta össze az egyedi lépéseket: kiinduló mintahalmaz gyűjtése, tanulás, elemzés, kiértékelés, modell optimalizálás [Hócza04a].

A gyakorlati alkalmazások az alábbi pontokba foglalhatók össze:

II/1. 3.2.4. fejezet: A szerző elkészített egy szövegkörnyezeti mintákon alapuló szófaji egyértelműsítőt, melynek alkalmazható mintáit az RGLearn algoritmussal állította elő. A módszer összehasonlításra került a szerző társai által kidolgozott módszerekkel [Kuba04].

II/2. 5.3.1. fejezet: A szerző alkalmazta a komplex faminta alapú módszert magyar nyelvű szövegek főnévi csoportjainak tanulására és felismerésére [Hócza04a]. Az szerző által elért eredmények jelentős javulást mutattak a magyar nyelvre ezt megelőzően közölt eredményekhez viszonyítva.

II/3. 5.3.2. fejezet: A főnévi csoportokra alkalmazott felszíni elemzés beépítésre került a szerző és társai által készített információkinyerő rendszerbe [Hócza03b], mely magyar nyelvű gazdasági rövidhíreken volt kiértékelve.

II/4. 5.4.2. fejezet: A komplex faminta módszert a szerző alkalmazta magyar nyelvű szövegek teljes szintaktikai elemzésére Hócza05b], [Hócza06a].

II/5. 5.4.3. fejezet: A szerző és társai a teljes szintaktikai elemzés faminta tanuló modelljét a Boosting algoritmussal optimalizálták [Hócza05a].

II/6. 5.4.5. fejezet: A szerző a teljes szintaktikai elemzőt beépítette a GenPar gépi fordító rendszerbe és létrehozott egy új, magyar-angol fordításra alkalmas kiegészítést [Hócza06b].

1.2. Az értekezés felépítése

Az értekezés öt fő fejezetre oszlik. Az 1. fejezetben az általános bevezetés, a szerző által elért eredmények részletezése és az értekezés fejezetenkénti áttekintése található.

A 2. fejezetben számba vesszük, hogy a természetes nyelvek szintaktikai elemzése milyen követelményeket támaszt az arra alkalmazott módszerekkel szemben, milyen nehézségek, problémás jelenségek vannak és ezekre milyen irányzatok, megoldási javaslatok születtek. A fejezet végén bemutatásra kerül egy a szerző által kidolgozott módszer, a famintákkal történő szintaktikai elemzés.

A 3. fejezetben a gépi tanulás fogalmának bevezetése után megismerjük azokat a technikákat, melyeket a szerző valamilyen szinten felhasznált a szabályalapú szintaktikus elemzési modellek építésében. Ebben a részben ismertetésre kerül egy a szerző által kifejlesztett mintatanuló algoritmus, az RGLearn, melyet faminták tanulására is alkalmazhatunk.

A 4. fejezetben található a szerző által kidolgozott komplett szintaktikai elemző módszer részleteinek kifejtése: a modellépítés folyamata, az elemző algoritmus megvalósítása, a kiértékelés technikája, valamint az eredmények visszacsatolásával végzett modell-optimalizálás.

Az 5. fejezet a szintaktikai elemzés magyar nyelvre történő alkalmazásának részleteit foglalja össze. Először a problémát írjuk le: milyen specialitások, nehézségek jellemzik a magyar nyelvet szintaktikus elemzési szempontból, és a rendelkezésre álló korpusz milyen információk felhasználását teszi lehetővé. A további részek a szerző szintaktikai elemzésben elért eredményeit mutatja be, valamint a felszíni és teljes szintaktikai elemzés megvalósított alkalmazásaira hoz példákat, információkinyerő illetve gépi fordító rendszerek részeként.

Az utolsó fejezet röviden összefoglalja az értekezést. Az értekezést a függelékben magyar és angol nyelvű összefoglaló, valamint irodalomjegyzék zárja.

2. A természetes nyelvek szintaktikai elemzése

Ebben a fejezetben áttekintjük, hogy a természetes nyelvek szintaktikai elemzése során milyen problémák adódhatnak és ezek kezelésére milyen megközelítési irányok, megoldási lehetőségek vannak, hogy jellemezzük azt, hogy egy nyelvészeti és számítási szempontból alkalmas szintaktikai leírás milyen követelményeknek kell, hogy megfeleljen. A fejezet végén ismertetésre kerül egy saját fejlesztésű szintaxisábrázolási módszer, a faminta formalizmus, mellyel nagyobb szintaktikai szerkezeteket lehet leírni és ezek segítségével az elemzést elvégezni.

Számos szabályalapú modell létezik természetes nyelvek mondattanához. A különböző típusú modellek leírásához különböző nyelvtani formalizmusokat használhatunk. A nyelvtani formalizmusok olyan metanyelvek – nyelveket leíró nyelvek –, amelyek definiálják azt a rendszert, amellyel egy természetes nyelv szabályai leírhatók. E metanyelvekkel szemben a következő követelményeket támaszthatjuk ([Shieber86]):

- ***Nyelvészeti alkalmasság***: annak mértéke, hogy az adott metanyelven mennyire lehet egyes nyelvi jelenségeket a nyelvészek által alkalmazott elveknek megfelelően kifejezni, valamint, hogy a metanyelvvel a nyelvi jelenségek mekkora körét lehet leírni.
- ***Számítási hatékonyság***: annak mértéke, hogy az adott nyelvtani formalizmus milyen hatékonyan valósítható meg számítógépen. Minél hatékonyabb a nyelvtani formalizmus, annál gyorsabb, illetve annál kisebb erőforrás-igényű a megvalósítására szolgáló számítógépes program.

Egy adott metanyelv kifejezőerejének jellemzésére kétféle megközelítés létezik. A korábban megjelent nyelvelméletek a formális nyelveket tekintették kiindulási alapnak, melyben egy nyelvtant az jellemez, hogy milyen nyelv (szimbólumsorozatok halmaza) vezethető le formális szabályai segítségével. Mivel ez a megközelítés később a természetes nyelvekre való alkalmazás tekintetében különféle korlátokba ütközött, a formalizmus által levezethető nyelv, mint a kifejezőerő jellemzése, a ***gyenge generatív kapacitás (Weak Generative Capacity)*** elnevezést kapta. Ehhez képest az ***erős generatív kapacitás (Strong Generative Capacity)*** arra koncentrál, hogy a formalizmus milyen strukturális leírását képes adni az elemzendő mondatnak.

A gyenge generatív kapacitás szerinti jellemzést használva viszonylag egyszerű a nyelvtanok nyelvészeti alkalmasság és számítási hatékonyság szerinti összehasonlítása, hiszen matematikailag pontosan leírható háttér áll rendelkezésre a generált nyelv és az adott osztályba eső nyelvtant elemző algoritmus típusának valamint bonyolultságának

meghatározására. A nyelvtani formalizmusok erős generatív kapacitás szerinti összehasonlítása ennél jóval nehezebb feladat, mivel a velük történt elemzés végeredménye esetenként igen eltérő formátumú lehet. A közös platformra hozás helyett egyszerűbb úgy feltenni a kérdést, hogy az adott formalizmus képes-e leírni a természetes nyelv bizonyos strukturális jelenségeinek körét és ha igen, ez milyen hatékonyságú algoritmussal végezhető el.

A nyelvtannal szemben támasztott követelmények együttes teljesítéséhez meg kell találnunk az optimális középutat. Minél bonyolultabb és részletesebb egy nyelvtani formalizmus, annál nagyobb az azt működtető elemző algoritmus tárigénye és feldolgozási ideje. Egy számítási szempontból hatékonyabb modell pedig a nyelvtan szűkítésével, különféle információk elhagyásával valósítható meg, ami viszont a nyelvészeti alkalmasságot csökkenti.

2.1. Generatív nyelvelmélet

A klasszikus generatív nyelvelmélet szerint a formális nyelvekkel megvalósított nyelvleírások jellemzője, hogy a nyelvtani szerkezeteket az alkotóelemeik felszíni sorrendje (azaz szövegbeli előfordulási sorrendje) alapján ismeri fel. A generatív nyelvtanok formális leírása a következő:

2.1 Definíció: Egy *(N, T, S, R)* formális négyest ***generatív grammatikának*** nevezzük négyes, ahol T a terminális szimbólumok véges halmaza, N a nemterminális szimbólumok véges halmaza, $N \cap T = \varnothing$, $V = N \cup T$ az összes szimbólum, $S \in N$ egy speciális kezdőszimbólum, R a helyettesítési szabályok véges halmaza, $R \subseteq V^+ \times V^*$.

A helyettesítési szabályok halmaza egy olyan Descartes szorzat, melynek minden eleme egy (α, β) páros, melyre a $\alpha \rightarrow \beta$ jelölés is használható. Adott G nyelvtan és π, $\sigma \in V^*$ szimbólumsorozatotok esetén σ ***egy lépésben levezethető*** π-ből (jelölés: $\pi \Rightarrow_G \sigma$), ha van olyan α_1, $\alpha_2 \in V^*$ és G–beli $\alpha \rightarrow \beta$ szabály, hogy $\pi = \alpha_1\alpha\alpha_2$ és $\sigma = \alpha_1\beta\alpha_2$. Ezt a szabályalkalmazást más néven ***derivációs lépésnek*** is nevezzük. A G nyelvtannal σ ***több lépésben levezethető*** π-ből ha létezik $\pi \Rightarrow_G \ldots \Rightarrow_G \sigma$ lépéssorozat. A G nyelvtan által generált nyelv egy olyan $w \in T^*$ (csak terminálisokból álló) szimbólumsorozatok halmaza, melyek S-ből egy vagy több lépésben levezethetők. Ezt a formalizmust ezért hívjuk generatív grammatikának, mert szabályok segítségével adhatók meg (generálhatóak) a nyelv elemei.

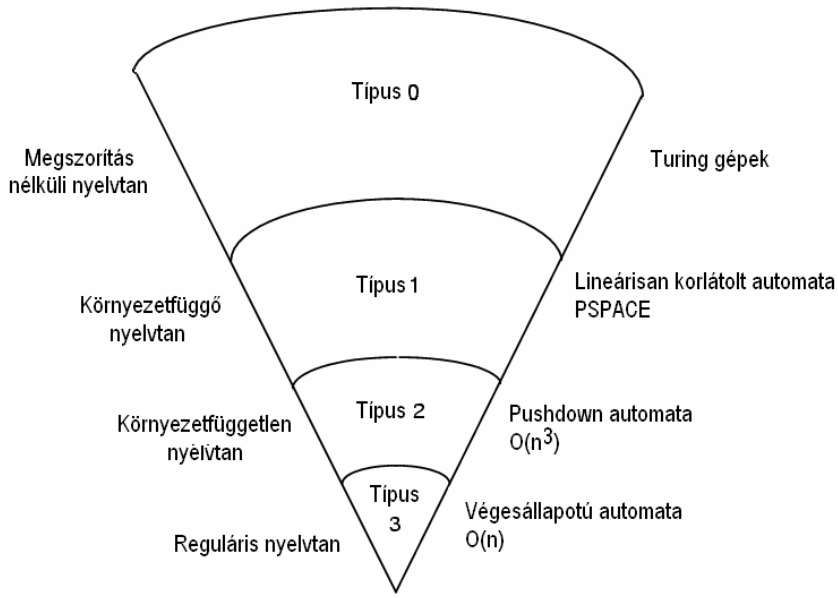
2.1.1. Nyelvtanok Chomsky-féle osztályozása

A generatív nyelvtan szabályaira vonatkozó megkötések alapján különféle típusú nyelvtanokat hozhatunk létre és tanulmányozhatjuk ezek generatív erejét, vagyis, hogy az adott típusú nyelvtannal tudunk-e generálni egy másik nyelvtantípus által generált nyelvet. Ezzel az eljárással meghatározhatunk különféle nyelvosztályokat és azok tartalmazási hierarchiáját azon az elven, hogy az a bővebb nyelvosztály amely képes generálni a szűkebb nyelvosztály által generált összes nyelvet. Ilyen hierarchia létrehozására számos lehetőség kínálkozik, azonban a számítógépes nyelvészetben a Chomsky-hierarchia [Chomsky57] vált elfogadottá.

2.2. Definíció: Legyen adva egy $G(N, T, S, R)$, valamint $\alpha, \beta, \gamma \in V^*$ szimbólumsorozatok, melyek ε értékűek is lehetnek, $\omega \in V^+$ szimbólumsorozatok, melyek nem lehetnek ε értékűek, $A, B \in N$ nemterminális szimbólumok, $a, b \in T$ terminális szimbólumok, ekkor:

- 0-ás típusú: ***megszorítás nélküli nyelvtan***, melyben a helyettesítési szabályokra nincs korlátozás: $\alpha A\beta \rightarrow \gamma$
- 1-es típusú: ***környezetfüggő nyelvtan***, melyben a helyettesítési szabályok alakja $\alpha A\beta \rightarrow \alpha\omega\beta$, valamint megengedett az $S \rightarrow \varepsilon$ szabály.
- 2-es típusú: ***környezetfüggetlen nyelvtan***, melyben a helyettesítési szabályok alakja $A \rightarrow \omega$, valamint az $S \rightarrow \varepsilon$ szabály is megengedett.
- 3-as típusú: ***szabályos (reguláris) nyelvtan***, melyben a helyettesítési szabályok alakja $A \rightarrow aB$ vagy $A \rightarrow a$ alakú lehet.

A definíciók által megadott nyelvosztályokra tartalmazási hierarchia áll fenn, ezen felül meghatározható az is, hogy milyen elven működő és milyen bonyolultságú algoritmus elegendő az adott nyelv elemzéséhez, vagyis a generált nyelv elemeinek felsorolásához (2.1**.** ábra). A nyelvosztályokat bonyolultságelméleti szempontból különböző hatékonyságú algoritmusokkal lehet elemezni. A reguláris modellek és a környezetfüggetlen nyelvtanok programozási szempontból hatékonyak, a reguláris nyelvtanok lineáris bonyolultsággal elemezhetőek véges automatával, a környezetfüggetlen nyelvtanok elemzése veremautomatával valósítható meg, melyek bonyolultsága polinomiális. Az ezután következő nyelvosztályok már elemzési szempontból kevésbé hatékonyak, lévén, hogy exponenciális vagy még annál is nagyobb bonyolultságú algoritmusok alkalmazhatók rájuk. Vannak olyan környezetfüggő grammatikák, melyek a PSPACE osztályba sorolódnak A megszorítás nélküli nyelvtanok Turing géppel ismerhetők fel, így ezek bonyolultsága a még nehezebb EXP kategóriába sorolható.



2.1. ábra Chomsky-hierarchia, és a nyelvosztályok bonyolultsága

2.1.2. Természetes nyelvek leírására alkalmas generatív modellek

Az a kérdés, hogy a milyen nyelvosztály alkalmas a természetes nyelvek leírására azért érdekes, mert már az is komoly gondot jelenthet az alkalmazhatóság tekintetében, ha a feldolgozási időt leíró függvény N mondathossz esetén N^2 (polinomiális) helyett 2^N (exponenciális). Míg rövid mondat elemzése esetén nincs jelentős a különbség a kettő között, ez hosszabb mondatoknál elfogadhatatlanul nagy futási idő különbségeket eredményezhet.

Chomsky művében azt is megmutatta, hogy a reguláris nyelvek nem alkalmasak a természetes nyelvek leírására, mivel a természetes nyelvekben szereplő jelenség, az ***önbeágyazás***, nem jellemezhető reguláris nyelvtannal. Ez a jelenség, mint azt az alábbi példa mutatja a magyar nyelvben is létezik:

„*János telefonált.*”

„*János, aki Éva barátja, telefonált.*”

„*János, aki Éva – akinek kölcsönadtam a könyvem – barátja, telefonált.*”

…

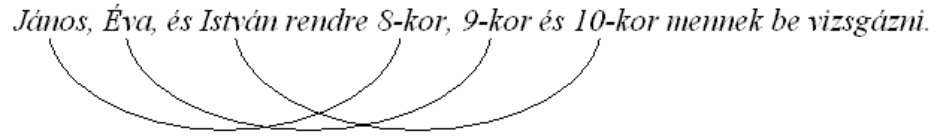
Elvileg a beágyazásokat a végtelenségig folytathatnánk, azonban azt is meg kell jegyeznünk, hogy 3-4 mélységű önbeágyazáson túl már nem igazán követhető a kapott mondatok jelentése, ezért ez az élő nyelvben csak ritkán és korlátozott mélységben

fordul elő. Ha viszont formálisan nézzük a jelenséget, az állítás, hogy a önbeágyazás jelensége nem írható le reguláris nyelvtannal a ***pumping lemmával*** igazolható, mely kimondha, hogy ha L egy végtelen reguláris nyelv, akkor léteznek olyan α, β, γ szimbólumsorozatok, melyek közül $\beta \neq \varepsilon$, hogy $\alpha\beta^n\gamma \in L$, ahol $n \geq 0$.

A pumping lemma bizonyítása azon alapul, hogy egy végesállapotú automatában, amely az L nyelv tetszőlegesen hosszú szavait fogadja el, az automata állapotszámánál hosszabb szavakra az érintett állapotokban kell lenni ismétlődésnek, azaz körnek (ez a β-t elfogadó rész) és egy ilyen kör tetszőleges számban megismételhető. A pumping lemma segítségével azt nem lehet bebizonyítani, hogy mely nyelv reguláris, azt viszont igen, hogy mi nem az.

Az önbeágyazás például nem reguláris, mert ha a szavak függőségeit a példamondatok alapján egy [[[…]]] zárójelezésnek fogjuk fel, melybe az összetartozó új párokat a szerkezet közepére szúrhatjuk be, akkor ezt a jelenséget az $a^m b^m$ nyelvvel lehet modellezni, amiről könnyen belátható, hogy nem hozható $\alpha\beta^n\gamma$ alakra tetszőleges n esetén.

A reguláris nyelvtanok után a környezetfüggetlen nyelvtanokhoz is születtek olyan a természetes nyelvi példákkal való cáfolatok ([Pullum84], [Shieber85]), mint például a ***keresztező függőségek***, mely jelenség magyar nyelvre a „rendre” típusú mondatokban figyelhető meg:



Az ilyen típusú mondatokban a függőségeket egy $\{\alpha\alpha \mid \alpha \in V^*\}$ nyelvvel ábrázolhatjuk, mely önmaga után fűzött szimbólumsorozatokat tartalmaz. Erről belátható, hogy nem írható le környezetfüggetlen nyelvtannal és így megállapítható, hogy a környezetfüggetlen nyelvtanok sem elegendőek a természetes nyelvek minden jelenségének a leírásához.

Ebben az esetben is meg kell jegyeznünk, hogy a keresztező függőségek csak korlátozott hosszban vannak jelen a természetes nyelvekben, ugyanakkor korlátlan méretű és végtelen elemszámú nyelvek felhasználásával bizonyítottuk be, hogy a jelenség környezetfüggetlen nyelvtanokkal nem ábrázolható. Ezért nem túl bölcs döntés ilyen elméleti megfontolások alapján, a természetes nyelvben ritkán és korlátozott méretben előforduló jelenségek miatt a hatékonyan elemezhető nyelvosztályokat kizárni a gyakorlati alkalmazásokból. Például ha változtatunk a definíciókon és bevezetünk egy K korlátot az adott jelenség (önbeágyazás, keresztező függőségek) feltételezett maximális méretére, akkor áthidaló megoldásokkal kezelni tudjuk a jelenségeket az adott nyelvosztály által biztosított eszközökkel is, így a hatékonyság gyakorlati

alkalmazásokban tartható marad. Ezzel a módszerrel viszont új, a generatív megközelítésnek „csak korlátozottan” eleget tevő formalizmusokat kapunk.

2.1.3. Enyhén környezetfüggő nyelvtanok

Annak ellenére, hogy nyelvi példákkal alátámasztott cáfolatok bizonyítják, hogy a környezetfüggetlen nyelvtanok nem elegendőek a természetes nyelvek leírásához, mégis a néhány kísérlettől eltekintve (pl. [Simmons92]) a környezetfüggő nyelvtanok természetes nyelvi alkalmazása lényegében felfedezetlen terület maradt. A modern nyelvészet inkább olyan megoldásokat talált ki, melyekben a jól kezelhető bonyolultság előnyeit megtartva környezetfüggetlen nyelvtanokat alkalmaz alapvázként és az ebből adódó nyelvészeti hiányosságokat valamilyen konstrukcióval, egyéb a kontextusra utaló információk bevezetésével hidalja át. Az ilyen rendszerek az ***enyhén környezetfüggő (mildly context-sensitive)*** elnevezést kapták.

2.3. Definíció ([Joshi85])**:** ***Enyhén környezetfüggő nyelvtan*** az a formalizmus, ami kielégíti az alábbi három kritériumot:

(1) Az elemzési probléma polinom időben végrehajtható.

(2) A nyelvtan által levezetett szavak hossza lineárisan növekszik (például $\{a^{2^n} \mid n>0\}$ nem ilyen).

(3) Le tud vezetni keresztező függőségeket, de ezek maximális hosszára az adott nyelv esetén megadható egy felső korlát.

A definíción túl meghatározták az enyhén környezetfüggő nyelvek családba tartozó ismertebb nyelvtani formalizmusok körét, valamint azt, hogy generatív szempontból mely formalizmusok között áll fenn ekvivalencia vagy tartalmazás ([Joshi91])**.** Az enyhén környezetfüggő nyelvtanok alkalmazása tehát kitüntetett szerepet kap a természetes nyelvek szintaktikai elemzése során, mivel az ebbe a nyelvosztályba tartozó formalizmusok segítségével lehet generatív szempontból hatékonyan megoldani a problémát, ezért formalizmus bevezetése esetén célszerű azt megvizsgálni, hogy az teljesíti-e az enyhén környezetfüggő nyelvtan definícióit.

2.2. Szintatikai elemző algoritmusok természetes nyelvekre

A különféle nyelvtani modellekhez szorosan kötődik, hogy milyen algoritmusok alkalmazásával végezhetjük el egy természetes nyelvi mondat szintaktikai elemzését, hiszen egy ilyen algoritmus végrehajtási módja és bonyolultsága nagyban függ a

nyelvtan jellegétől és az elemzés végeredményének általunk elvárt minőségétől. Mivel egy részletes nyelvtan az elelemzés során a többértelműségek miatt hatalmas keresési teret definiál, célszerű megvizsgálni, hogy milyen lehetőségeink vannak a hatékony végrehajtás eléréséhez. Ehhez több szempontot is össze kell hangolnunk, hiszen viszonylag gyorsan találhatunk egy lehetséges elemzést (amennyiben létezik ilyen), ami viszont nem feltétlenül a legjobb, vagy megkereshetjük a legjobbat az összes lehetséges elemzés felsorolásával és összehasonlításával, de ez az alkalmazott technikától függően nagyon időigényes is lehet.

2.2.1. Elemzés környezetfüggetlen nyelvtannal

Ha például adott egy *G* környezetfüggetlen nyelvtan és ennek szabályaival elemezni szeretnénk egy *M* természetes nyelvi mondatot akkor lényegében egy arra alkalmas algoritmussal azt keressük, hogy *M* eleme-e a *G* által generált nyelvnek és ha igen, milyen derivációs lépésekkel vezethető le az *S* start szimbólumból. Mivel egy derivációa szabályok alkalmazásának helyét és sorrendjét írja le, ezt kétféleképpen is megtehetjük: a gyökértől a levelek felé vagy fordítva. Ha a ***felülről-lefelé (top-down)*** építkezés elvét követjük, akkor az S szimbólumból kiindulva kifejtjük a nemterminális szimbólumokat az összes lehetséges módon, amíg meg nem kapjuk az elemzett mondatra illeszkedő első (vagy összes lehetséges) fát. Az **alulról-felfelé (bottom-up)** építkezés során mondat szavaiból indulunk ki azokat egyelemű részfáknak tekintve és a részfák gyökereire illesztjük az alkalmazható szabályok jobboldalát, amíg az első (vagy az összes lehetséges) olyan *S* gyökerű fát meg nem találjuk ami fedi a teljes mondatot.

```
TOP-DOWN-PARSE(Sentence, Grammar) returns parse
    Parse ← S
    loop
        if COVER(Parse, Sentence) then
            return Parse
        else
            foreach Rule of Grammar
            if APPLYCABLE(Rule, Parse) then
                PUSH(APPLY(Rule, Parse), Agenda)
        if Agenda is empty then
            return empty
        else
            Parse ← POP(Agenda)
```

2.2. Ábra A top-down elemzés leegyszerűsített algoritmusa

A 2.2. ábrán látható top-down elemzés algoritmusát egyszerűen átalakíthatjuk bottom-up elemző algoritmussá. A kétféle elelemzési módnak egymáshoz viszonyítva vannak előnyei és hátrányai. Például a top-down – más néven ***célvezérelt*** – elemzés mindenképp *S*-gyökerű fákat állít elő, a nyelvtan által előállítható egyéb elemzések kifejtésére nem fecsérel számítási kapacitást. A bottom-up – más néven ***adatvezérelt*** – elemzés viszont nem állít elő az input mondatra nem illeszkedő elemzést. Azonban számos közös problémát megállapíthatunk, ami felveti hatékonyabb elemző algoritmusok kidolgozásának szükségességét.

Az egyik ilyen probléma az, hogy semmi sem garantálja, hogy egy adott nyelvtan esetén bármely input mondatra létezzen arra illeszkedő *S* gyökerű elemzési fa, másrészt, ha a nyelvtan tartalmaz rekurzív szabályokat, illetve több lépéses rekurziót, azaz egy alkalmazott szabály feje újra felbukkan a levezetésben, ekkor az ismétlődés miatt a top-down elemzés végtelen ciklusba kerül.

Az alap módszer fő hátránya redundancia. Amikor elemzés során valamilyen megfontolás alapján elvetünk egy félig elemzett fát, akkor valójában egy visszalépés történik a keresésben. A soron következő fa továbbelemzése során sok esetben ugyanazok a derivációs lépések ismétlődnek, ami egyrészt felesleges műveletismétléssel jár, másrészt a memóriában kell tartani azt az esetenként igen nagyszámú fa verziót, melyek még nem kerültek sorra a feldolgozásban. Ez a nyelvtantól és az elemzendő mondattól függően elfogadhatatlanul nagy futási időt és memóriaigényt eredményezhet.

A dinamikus programozás alkalmazása lehetőséget teremt az előzőekben felvetett problémák hatékony megoldására. A dinamikus programozás során a redundancia miatt fellépő fölösleges számításokat úgy spóroljuk meg, hogy a problémát alproblémákra bontjuk és ezek kiszámított eredményét egy arra alkalmas táblázatban tároljuk, melynek minden lehetséges elemét csak egyszer számítunk ki.

2.2.2. Elemzés speciális környezetfüggetlen nyelvtanokkal

A szakirodalomban számos környezetfüggetlen nyelvtanra épülő algoritmus szerepel. Ezek közül legkorábban az ***LR algoritmus*** [Knuth65] és ennek különféle változatai, továbbfejlesztései jelentek meg. Az LR algoritmus lényege, hogy a nyelvtan előelemzésével egy olyan táblázatot generál, ami lényegében egy determinisztikus végesállapotú automata működését reprezentálja, így igen gyors, $O(N)$ bonyolultságú bottom-up elemzés hajtható végre vele. Az egyetlen hátránya, hogy a működése a determinisztikus környezetfüggetlen nyelvtanokra korlátozódik, ami miatt természetes nyelvek szintaktikai elemzésére nem, különféle programozási nyelvek fordítóprogramjaiban viszont kiválóan alkalmazható. Az általánosított LR algoritmus a

GLR [Tomita91] már képes volt többértelmű nyelvtani szabályokkal elemezni, azonban még ez az algoritmus is több kiegészítésre szorult, míg alkalmazni lehetett teljesen általános környezetfüggetlen nyelvtanra. Az LR algoritmus az evolúciója során elvesztette az igen kedvező lineáris bonyolultságát, az általános CFG-re alkalmazható változatnál ez már $O(N^3)$.

Egy másik szintaktikai elemző algoritmus különböző variációit három szerző J. Cocke, D. H. Younger és T. Kasami egymástól függetlenül dolgozta ki, ezért Cocke-Younger-Kasami vagy röviden ***CYK parser*** néven ismert. A leginkább hozzáférhető eredeti leírását Younger adta meg [Younger67]. Az algoritmus megszorítása a nyelvtan felé, hogy annak ***Chomsky normálformában (CNF)*** kell lennie, azaz csak $A \rightarrow BC \mid a$ alakú szabályokat tartalmazhat. Erre a formátumra bármilyen környezetfüggetlen nyelvtan átalakítható. Másrészt a nyelvtanról feltételezzük, hogy ε-mentes. A CNF előnye, hogy a szabályformái egyszerűségének köszönhetően egy nemterminális helyettesítésnek pontosan két tagja van, így ezt egy tömbben ábrázolva pontosan kiszámolható a szabályban szereplő szimbólumok tömbbeli indexe. A CNF hátránya, hogy az alkalmazásával kapott elemzési fa nem feleltethető meg a vele gyengén ekvivalens általános CFG-nek.

A CYK parser bottom-up elemzést alkalmaz. Működése során egy háromszög-mátrixot tölt ki minden lehetséges nemterminálisra úgy, hogy legalulra az alkalmazható $A \rightarrow a$ típusú szabályok baloldalai kerülnek, mivel ezek pontosan egy terminálist generálnak, a további tömbelemek kitöltésénél $A \rightarrow BC$ alakú szabályokkal von össze egymással szomszédos szócsoportokat. Valós értékű tömböt alkalmazva a felismert részfákhoz pontszám (pl. valószínűség) rendelhető, ami további heurisztikák bevezetését teszi lehetővé. Az algoritmus $O(N^3)$ bonyolultságú.

Az algoritmus hátránya, hogy CNF formátumú nyelvtannal működtethető, de egy általános CFG mindig átalakítható erre a formátumra, valamint, ha elegendő információt megőrzünk az eredeti nyelvtanból, az elemzéssel kapott bináris fát utófeldolgozási lépésként visszaalakíthatjuk az eredeti nyelvtannak megfelelően.

2.2.3. Chart parser és változatai

Az ***Earley parser*** néven ismerté vált algoritmus [Earley70], mely top-down elemzést valósít meg, szintén dinamikus programozási eszközöket alkalmaz a redundáns elemzési lépések kiküszöbölésére. Fő eleme egy olyan tömb – a ***chart*** – ami egy N hosszú mondat esetén N+1 bejegyzést tartalmaz, így minden a mondat minden szavára van a szót megelőző és azt követő chart pozíció. A chart minden pozíciója élek listáját tartalmazza melyhez $[A \rightarrow \alpha \bullet B\beta, [i,j]]$ érték párok vannak rendelve, melynek első eleme

a ***pontozott szabály*** azt jelzi, hogy a szabály felismerése során idáig az α szimbólumsorozatot sikerült ráilleszteni az inputra és a hátralévő részben B a soron következő még nem illesztett szimbólum. Az érték pár második eleme $[i,j]$ pedig azt jelzi, hogy a szabály illesztését az i-edik pozíción kezdtük el és jelenleg a j-edik pozíciónál tartunk. Egy felismert szabály $[A \rightarrow \gamma\bullet, [i,k]]$ alakú, melynek jelentése az, hogy az i és k chart pozíciók közötti szavaknak A felismert szintaktikai kategóriája. Az algoritmus három fő műveletet tartalmaz, melyek a következők:

(1) ***Predictor:*** Ha a vizsgált befejezetlen élben a felismerendő szabály $A \rightarrow \alpha\bullet B\beta$ alakú, az adott pozícióra létrehoz új éleket az összes $B \rightarrow \gamma$ alakú nyelvtani szabályra $B \rightarrow \bullet\gamma$ állapottal.

(2) ***Scanner:*** Abban a befejezetlen élben, melyben a felismerendő szabály $A \rightarrow \alpha\bullet B\beta$ alakú, megvizsgálja, hogy a soron következő B szófaji (POS) szimbólum-e, és ha igen és illeszkedik is a neki megfelelő pozíción lévő input szó szófajára, akkor továbblép a B szimbólumon, tehát $A \rightarrow \alpha B\bullet\beta$ lesz a szabályból.

(3) ***Completer:*** A $B \rightarrow \gamma\bullet$ alakú szabályhoz rendelt befejezett élhez megkeresi az összes olyan befejezetlen élet, melyben B a következő felismerendő szimbólum – azaz $A \rightarrow \alpha\bullet B\beta$ állapotban van – és azt továbblépteti a B szimbólumon, belőle tehát $A \rightarrow \alpha B\bullet\beta$ lesz.

Az Earley parser alkalmazott technikáinak továbbfejlesztésével kapott ***chart parser*** algoritmus [Kaplan73], [Kay86] számos olyan új elemet tartalmaz, mellyel flexibilisebbé tehető a gyakorlati alkalmazás. A legfontosabb ezek közül az ún. ***fundamentális szabály***, ami új él beszúrásakor aktivizálódik és az ***aktív*** vagy ***passzív*** élekkel való lehetséges összekapcsolásokat elhelyezi egy ***agenda*** nevű átmeneti tárolóba. Aktív él, melynek felismerése még nem fejeződött be, a passzív él pedig a felismert élt. Az agenda tárolási módjának megválasztása meghatározza az építkezési stratégiát. A ***verem (LIFO)*** feldolgozási módot alkalmazva mélységi, ***sor (FIFO)*** esetén pedig szélességi építkezést végezhetünk. További újítás, hogy nem csak top-down, hanem bottom-up elemzési stratégiát alkalmazhatunk, vagy ez akár ***kétirányú (bidirectional)*** is lehet, azaz menet közben dinamikusan is eldönthetjük, hogy az adott helyzetben melyik az előnyösebb. A algoritmus vázlata a 2.3. ábrán látható.

```
CHART-PARSE(words, grammar, agenda-strategy) returns chart
    INITIALIZE(chart, agenda, words)
    while agenda is-not empty do
          current-edge = POP(agenda)
          ADD-TO-CHART(edge)
          FUNDAMENTAL-RULE(edge)
          MAKE-PREDICTIONS(edge)
    return chart

FUNDAMENTAL-RULE(edge)
    if edge = (A → α•Bβ, [i,j]) then
          foreach (B → γ•, [j,k]) in chart
                ADD-TO-AGENDA((A → αB•β, [i,k]))
    elsif edge = (B → γ•, [j,k]) then
          foreach (A → α•Bβ, [i,j]) in chart
                ADD-TO-AGENDA((A → αB•β, [i,k]))

MAKE-PREDICTIONS(edge)
    if Top-Down and edge = (A → α•Bβ, [i,j]) then
          foreach (B → γ) in grammar
                ADD-TO-AGENDA((B → •γ, [j,j]))
    elsif Bottom-Up and edge = (B → γ•, [i,j]) then
          foreach (A → Bβ) in grammar
                ADD-TO-AGENDA((A → B•β, [i,j]))

ADD-TO-CHART(edge)
    if edge is-not-in chart then
          PUSH(state, chart_entry)

ADD-TO-AGENDA(edge)
    if edge is-not-in agenda then
          PUSH(state, chart_entry)
```

2.3. Ábra. A chart parser algoritmusa ([Jurafsky00])

Bizonyos természetesnyelvi feladatok esetén előnyös lehet, ha a szintaktikai elemzés inkrementálható, azaz menet közben változtatva az inputot, nem kell a teljes szintaktikai elemzést újra elvégezni, mert az elemző algoritmus fel tudja használni az előző input-

állapot szintaktikai elemzését, kiegészítve azt az új rész elemzésével. Erre a feladatra a chart megközelítés alkalmasnak látszik, mivel a szöveg nem változó részeit lefedő chart éleket nem kell újraelemezni. Ez tulajdonképpen egy olyan kihasználható előny, ami a redundanciát kiküszöbölő dinamikus programozás technikának köszönhető. Az ***inkrementális chart parser*** [Wirén89] az eredeti algoritmus kiegészítésével képes a szócsoportok beszúrásának, törlésének és cseréjének különféle eseteit dinamikusan kezelve aktualizálni chart tartalmát az aktuális szöveg szintaxisának megfelelően, teljes újraelemzése nélkül.

Összehasonlítva a CFG-vel történő teljes szintaktikai elemzésre készült algoritmusokat megállapíthatjuk, hogy a chart parser számos szempont szerint felülmúlja a többi elemzőt, mivel nincs megszorítás a benne alkalmazható nyelvtani szabályokra, flexibilis, nagy szabadsági fokot biztosít arra, hogy megfelelő kiegészítésekkel alkalmazni lehessen különféle elemzési feladatokra és formalizmusokra, valamint bonyolultság tekintetében is megfelelő, mivel ez $O(N^3)$ általános CFG nyelvtanra.

2.2.4. Felszíni elemzés

Néhány természetes nyelvvel kapcsolatos feladat nem igényel teljes szintaktikai elemzést. Az input mondat ***felszíni elemzése (Shallow Parsing)*** is elegendő az olyan feladatokhoz, mint például az ***információkinyerés (Information Extraction)*** ahhoz, hogy megfelelő információkkal lássa el a feladat által meghatározott adat-sablont. Ez az egyszerűsítés a szintaktikai elemzésben sebességnövekedéssel jár, ami lehetővé teszi nagymennyiségű adat gyors feldolgozását és ez számos további alkalmazási lehetőséget teremt.

Az ilyen rendszerekben központi szerep jut a főnévi csoportok felismerésének, mivel az igék vonzatait többnyire főnévi csoportok alkotják, ezeket a kapcsolatokat az igének megfelelő szemantikus keretben ábrázolva már meg is kaptuk a feladat végrehajtásához elegendő információt biztosító mondat-struktúrát. A főnévi csoport (NP) legfontosabb eleme a fej (az NP utolsó szava, ami főnév) és vannak még az ezt megelőző módosítószavak (pl. névelő, melléknév, számnév). Egy ilyen szósorozatot például az alábbi módon ábrázolhatjuk:

$$NP \rightarrow (Det)\ \{Adj|Num\}^*\ Noun$$

Ez a CFG szabályok jelölésére emlékeztető forma valójában egy végesállapotú automatát ad meg, melyben több CFG szabály össze lett vonva.

Természetesen vannak más, esetenként összetettebb szerkezetű főnévi csoportok is, melyekben további szófaj (például kötőszó) vagy információ (nem, szám, eset) szerepel, de ezeket össze lehet szerkeszteni egyetlen nagy NP-felismerő automatává. Az automaták alkalmazásának az előnye, hogy lineáris bonyolultságú feldolgozás végezhető el velük, a hátrány – mint azt az LR elemzőknél is láttuk - viszont az, hogy csak korlátozott CFG alakítható át a lineáris bonyolultság megtartásával.

A fő problémát azt automata alapú felismerők esetén a rekurzió jelenti. Az egyik kezelési mód azt ***automata-kaszkád*** alkalmazása [Abney96] melynél véges rekurziós mélységet feltételezve több szintű szócsoport felismerés történik, esetenként egyes szinteken különböző szócsoportokat felismerő automatákkal. Ilyen elven működik például FASTUS [Hobbs96], illetve a CLaRK rendszer [Simov01]. A másik megközelítés a ***Recursive Transition Network*** [Fodor80] alkalmazása, amely valójában már nem is automataként működik, mert a különféle szócsoportok felismerését eljárásként kezeli és a felismerés végezte utáni visszatéréshez szükséges információkat egy veremben tárolja.

Egy másik lehetőség a felszíni elemzés szócsoportjainak meghatározására a ***chunking***. Ebben az esetben a problémát visszavezetjük a szófaji egyértelműsítésnél (POS tagging) használt módszerekre. Egy jellemző probléma lehet például a legbelső főnévi csoport (base-NP) meghatározása és az a feladat, hogy a mondat szavaihoz rendeljünk hozzá öt kategóriát: NP eleje (B), NP belső szava (I), NP vége (E), egy tagú NP (BE) vagy NP-n kívül esik (O).

A chunking algoritmusát vezérlő modell előállítása annotált korpusz felhasználásával jellemzően gépi tanulással történik, melyre használhatunk például szabályalapú gépi tanuló módszert, mint a C4.5 [Quinlan93], vagy HMM taggert [Charniak93]. A gépi tanulás előkészítésekor rendelkezésünkre állhat, így felhasználhatjuk a szófaji egyértelműsítés végeredményét is.

A szabályalapú megközelítés jellemzője, hogy a vizsgált szópozíció előtti-mögötti környezetében egy ablakon belül kérhetünk le információkat a szomszédos szavakról és döntünk ha az adott helyzetre vonatkozó szabály. Általában egy menetben nem lehet feloldani a többértelműségeket, ezért visszalépéses keresést (backtracking) kell alkalmaznunk.

A chunking módszernél is lehetőség van többszintű (kaszkád) szócsoport-felismerésre, de ebben az esetben is kezelni kell a felismerendő CFG szabályok rekurziójának lehetőségét, például az automata megközelítésnél megemlített módszerek valamelyikével.

2.3. Erős generatív kapacitást igénylő nyelvi jelenségek

A természetes nyelvekben felbukkanó speciális jelenségek – mint a korábban tárgyalt önbeágyazás és keresztező függőségek – miatt a reguláris és környezetfüggetlen nyelvek a generatív megközelítés alapján nem alkalmasak a természetes nyelvek minden jelenségének a leírásához. Mivel ezek a jelenségek az élő nyelvben csak ritkán és korlátozott mértékben fordulnak elő, ezek miatt még nem lenne indokolt, hogy lemondjunk a polinomiális időben elemezhető környezetfüggetlen nyelvtanok alkalmazásáról. Azonban a természetes nyelvek tartalmaznak olyan jelenségeket is, melyek ténylegesen gyakran előfordulnak, meghatározó szerepük van a mondatok szintaxisában, ugyanakkor a generatív nyelvtanok nem alkalmasak ezek hatékony ábrázolására. Ebben a fejezetben ezeket a jelenségeket tekintjük át, valamint felvetett problémákra megadjuk a szakirodalom által kínált megoldásokat is.

2.3.1. Szabályok alkalmazásának statisztikai valószínűsége

A környezetfüggetlen nyelvtanok egy hátránya, hogy minden szabály ugyanolyan súllyal szerepel benne. Ha rendelkezésünkre áll egy szintaktikai elemzéseket tartalmazó korpusz, akkor ezen összeszámolhatjuk a különféle szabályok előfordulási gyakoriságát és azt fogjuk tapasztalni, hogy a természetes nyelvre való alkalmazhatóság tekintetében jelentős különbségek vannak a szabályok között, mint például az alábbi főnévi csoportok, melyek közül legnagyobb számban a legelső szabályalkalmazásra találhatunk példát:

$NP \rightarrow Det\ Noun$	(*a kutya*)
$NP \rightarrow Noun$	(*kutya*)
$NP \rightarrow Det\ Num\ Adj\ Noun$	(*a két fekete kutya*)

Feltételezhetjük, hogy egy ismeretlen szövegen is nagyjából ugyanazt a szabályeloszlást kapnánk, mint amit a rendelkezésünkre álló korpuszon mértünk, ezért előfordulási gyakoriság különbséget tesz a szabályaink között, melyet egy alkalmas mérőszám bevezetésével célszerű figyelembe vennünk. Mindennek akkor lesz jelentősége, amikor elég nagy nyelvtanunk van ahhoz, hogy elemzés során többértelműségeket kapjunk és valamilyen szempont szerint választanunk kell a lehetséges elemzési fák közül.

A ***valószínűségi környezetfüggetlen nyelvtan*** [Booth69] vagy röviden ***PCFG (Probabilistic Context Free Grammar)*** valószínűség fogalmának bevezetésével természetes kiterjesztését adja a CFG-nek.

2.4. Definíció: A ***PCFG*** egy olyan *(N, T, S, R, P)* ötös, ahol *N* a nemterminális szimbólumok véges halmaza, *T* a terminális szimbólumok véges halmaza, $N \cap T = \varnothing$, $V = N \cup T$ az összes szimbólum, *S* a *start* szimbólum, $S \in N$, *R* levezetési szabályok véges halmaza, $R \subseteq N \times V^*$, $P \subseteq R \times [0..1]$ valószínűségi függvény egy 0 és 1 közötti valós számot rendel minden levezetési szabályhoz.

Minden nemterminális szimbólumra igaz, hogy lehetséges levezetési szabályaihoz rendelt valószínűségek teljes eseményrendszert alkotnak:

$$\forall A \in N : \left(\sum_{B \to \beta \in R} P(A \to \beta \mid A) \right) = 1 \tag{2.2}$$

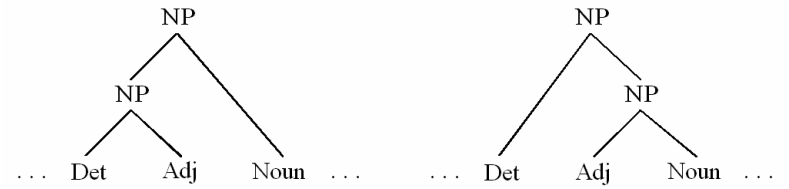
Ha adott egy *S* mondat, melyre az adott nyelvtan alapján lehetséges elemzési fáinak halmazát $\mathfrak{I}(S)$ jelöli, akkor keressük azt a $T_{max}(S)$ elemzési fát ami legvalószínűbb lesz *S*-re nézve. Ennek az elvnek a formális levezetése a következő:

$$\begin{aligned} T_{\max}(S) &= \arg\max_{T \in \mathfrak{I}(S)} P(T \mid S) \\ &= \arg\max_{T \in \mathfrak{I}(S)} \frac{P(S \mid T) P(T)}{P(S)} \\ &= \arg\max_{T \in \mathfrak{I}(S)} P(T) \end{aligned} \tag{2.3}$$

A levezetésben a Bayes szabály alkalmazása után kihasználtuk azt, hogy csak olyan elemzési fákat veszünk figyelembe melyek tartalmazzák a mondat összes szavát, így $P(S \mid T) = 1$, valamint, hogy a $P(S)$ valószínűség minden elemzési fára nézve ugyanaz a konstans érték, így a maximumkeresést nem befolyásolja. Ha egy adott *T* elemzési fa a $A_i \to \beta_i$ ($i=1\ldots n$) ismert szabályok alkalmazásának sorozatával állt elő, melyeket független eseményeknek tekintünk, így *T* valószínűségét a szabályalkalmazások valószínűségeinek szorzataként kapjuk meg:

$$P(T) = \prod_{i=1}^{n} P(A_i \to \beta_i \mid A_i) \tag{2.4}$$

A részelemzések valószínűségét figyelembe véve az elemző algoritmus működése gyorsítható heurisztikák alkalmazásával. Például a Viterbi algoritmus [Viterbi67] alapötletét felhasználva, miszerint az azonos mondatrészt lefedő és azonos gyökerű részfák (2.4. ábra) közül a legvalószínűbbet érdemes megtartani, mivel a maximumkeresés miatt biztos, hogy a legvalószínűbb részfa részszorzata lesz benne a keresett legvalószínűbb derivációs fa valószínűségi szorzatában.



2.4. Ábra. A Viterbi módszer alkalmazása

A 2.2 fejezetben ismertetett elemző algoritmusokat (LR-, CYK-, bottom-up chart parser) bottom-up stratégia esetén könnyű úgy módosítani, hogy már elemzés közben egyúttal ki is számítsák a valószínűségeket. Amikor az elemző befejezte egy új csúcs felismerését, akkor a felismert szabály és az általa lefedett szimbólumok valószínűségei alapján ki lehet számolni az új csúcs valószínűségét, melyet ezek szorzataként kapunk meg. A Viterbi módszer beépítésével, megfelelő feltételek mellett elhagyhatjuk a kisseb valószínűségű csúcsokat és ezzel növelhetjük az elemzés sebességét.

A valószínűség bevezetésének az előnyök mellett vannak hátrányai is. Például az, hogy sok érték kiszámítása és tárolása kell hozzá és új szabályok bevezetése esetén újra kell számolni az összes valószínűséget.

2.3.2. Lexikális és strukturális függőségek

Amikor egy természetes nyelv elemzésére készült környezetfüggetlen nyelvtan a szavak esetén csak azok szófaját veszi figyelembe, akkor a szabályok alkalmazásánál egyforma jelentőségűnek tételezi fel az összes azonos szófajú szót. Bizonyos többértelmű esetekben azonban a szavak jelentése erősen befolyásolhatja azt, hogy pontosan melyik szabály alkalmazása valószínűsíthető a lehetőségek közül. Erre példa az alábbi két mondat:

A vadász nézte a szarvast a távcsővel.

A szarvas nézte a vadászt a távcsővel.

A két példamondatra ugyanazok a szabályok alkalmazhatók, de a mondat értelmezése alapján kizárhatóak azok a részelemzések melyekben a távcső a szarvashoz tartozik, azaz a $[_{NP}\ [_{NP}$ *a szarvast* $]\ [_{NP}$ *a távcsővel* $]\ ]$ kizárható az első mondat elemzésében, míg a második mondatban a $[_{NP}\ [_{NP}$ *a vadászt* $]\ [_{NP}$ *a távcsővel* $]\ ]$ szerkezet valószínűsíthető, ugyanakkor lexikális információk nélkül a két eset egyenrangúnak tekinthető a $[_{NP}\ [_{NP}$ *Det Noun* $]\ [_{NP}$ *Det Noun* $]\ ]$ részfát előállító szabályok alkalmazása során. Erre a lexikális eltérésre következtethetünk abban az esetben is, ha rendelkezünk elegendően nagy méretű elemzett korpusszal arra, hogy megszámoljuk a <vadász, távcső> illetve a <szarvas, távcső> összerendelések

előfordulásait, melyeket normalizálva megbecsülhetjük az egyik, illetve másik lehetőség valószínűségét.

A ***lexikális függőségek*** okozta problémák kezelése ***lexikalizált valószínűségi nyelvtan*** [Charniak97] alkalmazásával oldható meg, melynek szabályaiban minden nemterminális szimbólumhoz hozzá van rendelve egy lexikális fej, azaz egy olyan szó amelyhez hozzárendelhető az azt körülvevő szócsoport többi szava. Ezáltal az eredeti nyelvtan minden szabályának annyi másolata van ahány féle a szabállyal lefedhető <szimbólum, fej> párokból álló sorozat előfordul a korpuszban. Az így kapott lexikalizált szabályokhoz rendelt valószínűségek becslése a korpuszbeli előfordulások alapján történik. A módszer hátránya, hogy nagyon sok szabály keletkezik, valamint nagyon nagy annotált korpusz kell a valószínűségi modell elkészítéséhez, ezért simításra van szükség, hogy a 0 előfordulások esetén is becsülni lehessen a valószínűséget.

A PCFG elemzés esetén a szabályalkalmazásokat egymástól független eseményeknek tekintjük, ezáltal leegyszerűsödik az elemzési fa valószínűségének kiszámítása, így az a (2.4) képlet alapján a szabályalkalmazások valószínűségeinek a szorzata lesz. Bizonyos esetekben azonban egy szabály alkalmazhatósága nem független attól, hogy az alkalmazásával kapott csúcs hol helyezkedik el az elemzési fában. Ezért PCFG elemzés során felléphet ***túlgenerálás***, vagyis az a jelenség, hogy a nyelvtan szabályainak alkalmazásával olyan szerkezetek is relatíve nagy valószínűséget kapnak, melyek az annotált korpuszban semmiképp sem fordulhatnának elő.

Például ha vesszük a $NP \rightarrow NP\ Noun$ rekurzív szabályt, ennek p valószínűsége relatíve nagy lesz, mivel a szabály gyakran előfordul a korpuszban. Az n-szeres rekurzió becsült valószínűsége p^n a viszonylag nagy p miatt nagyobb n-ek esetén is relatíve nagy lesz, ugyanakkor a 3-nál nagyobb rekurziós mélységű szövegrészek már annyira erőltetettek, hogy egyszer sem fordulnak elő „normális” szövegben, például:

Pista főnöke

Pista főnökének kalapja

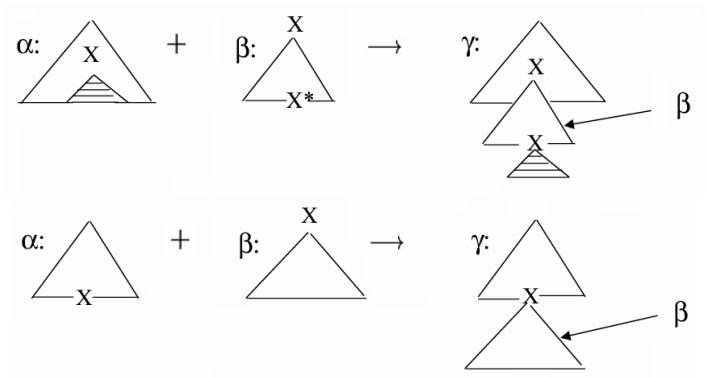
Pista főnöke kalapja bojtjának …

A ***strukturális függőségek*** okozta problémák úgy kerülhetők el, ha a korpuszból vett nagyobb szerkezeteket alkalmazunk az elemzés során. A korpuszban ténylegesen létező többszintű szerkezetek, részfák kontextust adnak a belső leírásuk derivációs lépéseihez, így nem fordulhat elő túlgenerálás. Egy ilyen, a részfák összekapcsolásával elemző formalizmus formális definíciója a következő:

2.5 Definíció [Joshi92]**:** A ***faösszekapcsoló nyelvtan (Tree Adjoining Grammar, TAG)*** egy olyan *(T, N, I, A, S)* ötös, ahol *T* terminális szimbólumok véges halmaza, *N*

nemterminális szimbólumok véges halmaza, $N \cap T = \emptyset$. Az I a ***kezdeti fák (initial tree)*** véges halmaza, melyek nem tartalmaznak rekurziót, azaz a fa által levezetett szimbólum nem tér vissza később a fa belsejében. Az A olyan ***segédfák (auxiliary tree)*** véges halmaza, melyek alapcsúcsának és gyökerének címkéje megegyezik, azaz rekurziót tartalmaz. Az S pedig a ***start*** szimbólum, $S \in N$

A TAG elemzésekben a derivációs fa a kezdeti és segédfákon végzett faösszekapcsoló műveletek sorozatával áll elő, ehhez kétféle művelet áll rendelkezésre, melyet a 2.5. ábrán szemléltetjük. Az ***adjunkció*** során veszünk egy β segédfát és egy α kezdeti vagy derivációs fát, úgy, hogy β alapcsúcsának címkéje megegyezik α egy N belső csúcsának címkéjével, ahol N gyökércsúcsa a T részfának α-ban. Az N gyökerű T részfát eltávolítjuk el α-ból, majd β-t α-ba illesszük T helyére, T-t pedig β alapcsúcsa helyére. A ***behelyettesítés*** pedig úgy történik, hogy veszünk egy β kezdeti fát és egy α kezdeti vagy derivációs fát, úgy, hogy β gyökércsúcsának címkéje megegyezik α egy szabad N csúcs címkéjével és β-t N helyére illesztjük.



2.5. Ábra. Az adjunkció és a behelyettesítés végrehajtása a TAG formalizmusban

A lexikális és strukturális függőségek kombinálása is megoldható, ezért a TAG formalizmusnak létezik lexikalizált változata is a ***lexikalizált TAG (LTAG)*** [Joshi92], melyek minden kezdeti és segédfája tartalmaz legalább egy szó terminálist, amit ***horgonynak (anchor)*** nevezünk.

2.3.3. Távoli függőségek és szabad szórend

A mondat szavait elemezve megfigyelhetjük, hogy bizonyos szavak valamilyen szempont szerint összepárosíthatók egymással. Egy szó jelentését egy hozzákapcsolt másik szó meg is változtathatja (specializálja az általános jelentést), ezért a kapcsolatokat egy irányított nyíl segítségével is ábrázolhatjuk. Ez a reláció azt is kifejezi, hogy az egyik szó mondatbeli szerepe valamilyen szempont (például jelentés) alapján megváltozik, azaz ***függ*** attól, hogy milyen szó van hozzákapcsolva.

Környezetfüggetlen nyelvtanok függőségekre való alkalmazásában az jelenti a problémát, hogy a függőségek nem feltétlenül szomszédos szavak között vannak, hanem lehetnek távoli, az érdektelen mondatrészen átívelő kapcsolódások is. Mivel nem tudunk olyan szabályt megadni amit nem összefüggő mondatrészekre alkalmazni lehetne, szükség van egy függőségek leírására alkalmas módszerre.

A ***függőségi nyelvtan (dependency grammar)*** irányított bináris kapcsolatot teremt a mondat szavai között. Minden bináris kapcsolatnak van egy ***feje*** és egy ***módosítója***, a kapcsolat jellegét a fejből a módosítóba mutató irányított él jelzi. Az élekhez lehetnek címkék hozzárendelve, melyek a függőségi relációk típusát adják meg. A függőségek kiinduló pontja a főige, de a mondat minden szava be van vonva a függőségek láncolatába, mely egy összefüggő szerkezetet alkot a teljes mondatra. Ennek a megközelítésnek az elmélete szerint mondat egy olyan szósorozat, amire az igaz, hogy létezik egy olyan kapcsolati gráf, amely megfelel az alábbi feltételeknek:

- ***Kereszteződés***: kirajzolva a kötéseket, azok nem keresztezhetik egymást.
- ***Összeköttetés***: az összes szó összeköttetésben áll valamelyik másik szóval.
- ***Feltétel***: az összes összeköttetés megfelel a nyelvtan követelményeinek.

A függőségi nyelvtanok további jellemzője, hogy nem tesz fel semmit a szórendről (a generatív nyelvtanokkal ellentétben), valamint nem csak egymást követő szavak között lehetnek függőségek. A ***szabad szórend*** nem jelenti azt, hogy bármelyik szót bárhova tehetjük a mondatban, például magyar nyelvben a főnévi csoportok szavai kötöttek (*a piros autó ≠> autó piros a*). A szabad szórend például az igék vonzatainak átrendezhetőségét jelenti:

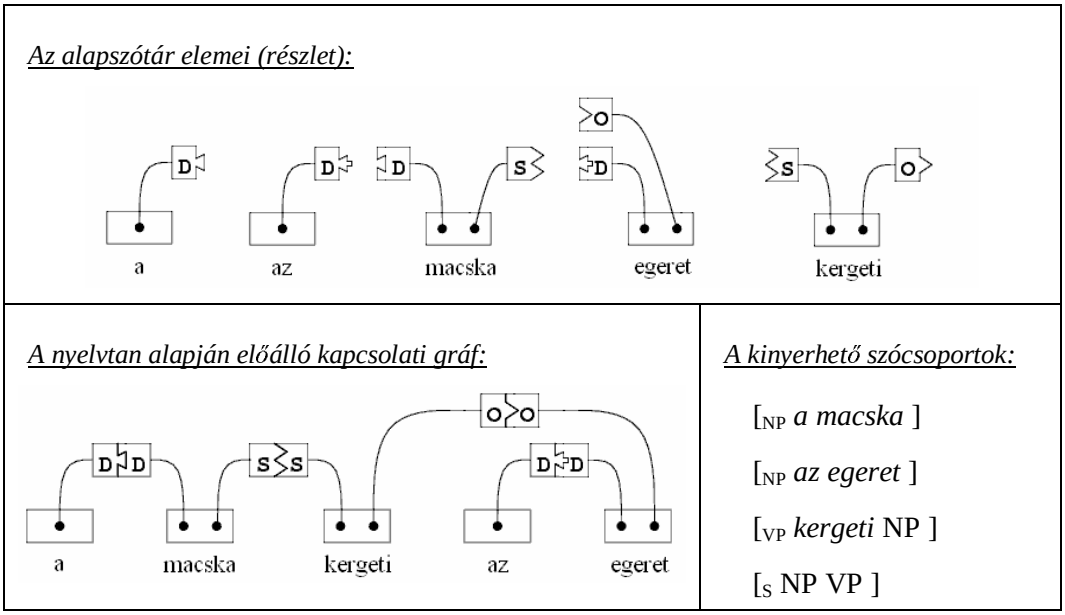
$$[Péter]_{Nom}\ elküldte\ [a\ levelet]_{Acc}\ [a\ barátjának]_{Dat}$$

$$elküldte\ [a\ levelet]_{Acc}\ [a\ barátjának]_{Dat}\ [Péter]_{Nom}$$

$$[a\ levelet]_{Acc}\ [a\ barátjának]_{Dat}\ [Péter]_{Nom}\ elküldte$$

Azonban ez is elegendő ahhoz, hogy a szabályaink számát drasztikusan megnövelje, ha az igék vonzatait környezetfüggetlen szabályokkal akarnánk leírni, mivel minden lehetséges sorrendre fel kellene vennünk egy új szabályt.

A legismertebb függőségi nyelvtant alkalmazó elemző a ***Link parser*** [Sleator91], ami egy tudományos célra ingyenesen letölthető programcsomag. Az elemző alapját egy ún. ***link nyelvtan*** képezi, amely azt írja le, hogy a szavak, valamint szintaktikai egységek hogyan kapcsolódhatnak egymáshoz. Az elemző program ez alapján felépíti a kapcsolati, kinyerhetőek a szintaktikai elemzés szócsoportjai (NP, VP, PP, …). A 2.6. ábrán bemutatott rövid példa szemlélteti az elemző működését.



2.6. Ábra. Egy rövid példamondat elemzése link nyelvtannal.

2.3.4. Egyeztetés és alkategórizálás

Természetes nyelvekben gyakran előforduló jelenség az ***egyeztetés***, azaz a szavak rendelkezhetnek olyan nyelvtani tulajdonságokkal (eset, szám, nem) melyeknek az összetartozó szócsoportok szavaira nézve összhangban kell lenniük. Különösen jellemző ez a ragozó nyelvekre (például szláv nyelvek, német, magyar), de még a nem ezek közé csoportosítható angol nyelvben megfigyelhető ez a jelenség (például az ige -s ragot kap egyes szám harmadik személyű alany esetén).

Angol nyelvnél maradva a problémát környezetfüggetlen nyelvtan esetén kezelhetjük úgy, hogy új terminális és nemterminális szimbólumokat vezetünk be melyekben jelöljük, hogy a főnevek és igék egyes vagy többes számban vannak-e és az egyes szám harmadik esetét is megkülönböztetjük. Az új szimbólumokkal elő kell állítanunk a meglévő szabályaink minden olyan lehetséges változatát, illetve azok helyett, ami megfelelően leírja az egyeztetés feltételeit. Például az alábbi szabályok lehetnének az $NP \rightarrow Noun \mid Det\ Noun$ és $VP \rightarrow Verb\ NP$ eredeti szabályok egyeztetett változatai:

$$NP_{sing\_n3} \rightarrow Noun_{sing\_n3}$$

$$NP_{sing\_3} \rightarrow Det_{sing\_3}\ Noun_{sing\_3}$$

$$NP_{plural} \rightarrow Det_{plural}\ Noun_{plural}$$

$$VP \rightarrow Verb_{n3}\ NP_{sing\_n3} \mid Verb_{3}\ NP_{sing\_3} \mid Verb_{n3}\ NP_{plural}$$

Az új szabályok bevezetésének módjából látható, hogy a nyelvtan mérete az átalakítás után az eredeti többszöröse lesz. Ez angol nyelv esetén még elfogadhatónak tűnik, a módszer azonban nem vihető át a ragozó nyelvekre (így magyarra sem), melyben figyelembe kellene vennünk az egyes és többes szám 3 esetét, a különféle igeragozásokat, az főnevek eseteit (alany, tárgy, birtokos, stb.) és egyes nyelvekben a nemét (hímnem, nőnem, semleges sem), valamint bizonyos nyelvekben (például orosz) még a mellékneveket is egyeztetni kell.

Az egyeztetéshez hasonló jelenség az ***alkategórizálás***. Az igéknek különböző vonzatai lehetnek, például a *megy* igének van alanya, de nincs tárgya, a *csinál* igének van tárgya is (*csinál valamit*), a *elcserél* igének pedig az alanyon kívül két vonzata van (*elcserél valamit valamire*). Ez az információ fontos lehet az elemzés során, mert ennek ismeretében az ige vonzatainak megfelelő szabályt kell alkalmaznunk. Úgy is eljárhatunk, hogy az igéket a vonzataik lehetséges száma alapján (al)kategóriákba soroljuk (például *intranzitív*, *tranzitív* és *ditranzitív*) és ezekre új szimbólumokat vezetünk be. Ez az eljárás újabb szabályváltozatok létrehozásával jár, ami azt mutatja, hogy alkategórizálás generatív nyelvtanok esetén a nyelvtan méretének növekedését eredményezi. Ez a hatás még inkább jelentkezik, ha a pontos alkalmazás érdekében vonzatok esetét (például *alany*, *tárgy*, *eszköz-*, *hely-*, *időhatározó*, stb.) is jelöljük.

A főneveknek is vannak alkategóriái (*köznevek, tulajdonnevek, személyes névmások*, stb.) melyeknek szintaktikai szempontból is eltérő tulajdonságaik lehetnek. Például magyar nyelvben a köznevek esetén többnyire használunk névelőt, a tulajdonnevek előtt pedig nem (például [$_{\text{NP}}$ *a fiú*], illetve [$_{\text{NP}}$ *Pista*]) valamint más névelőt használunk a mássalhangzóval és magánhangzóval kezdődő köznevek előtt (például [$_{\text{NP}}$ *a fiú*], illetve [$_{\text{NP}}$ *az ember*]).

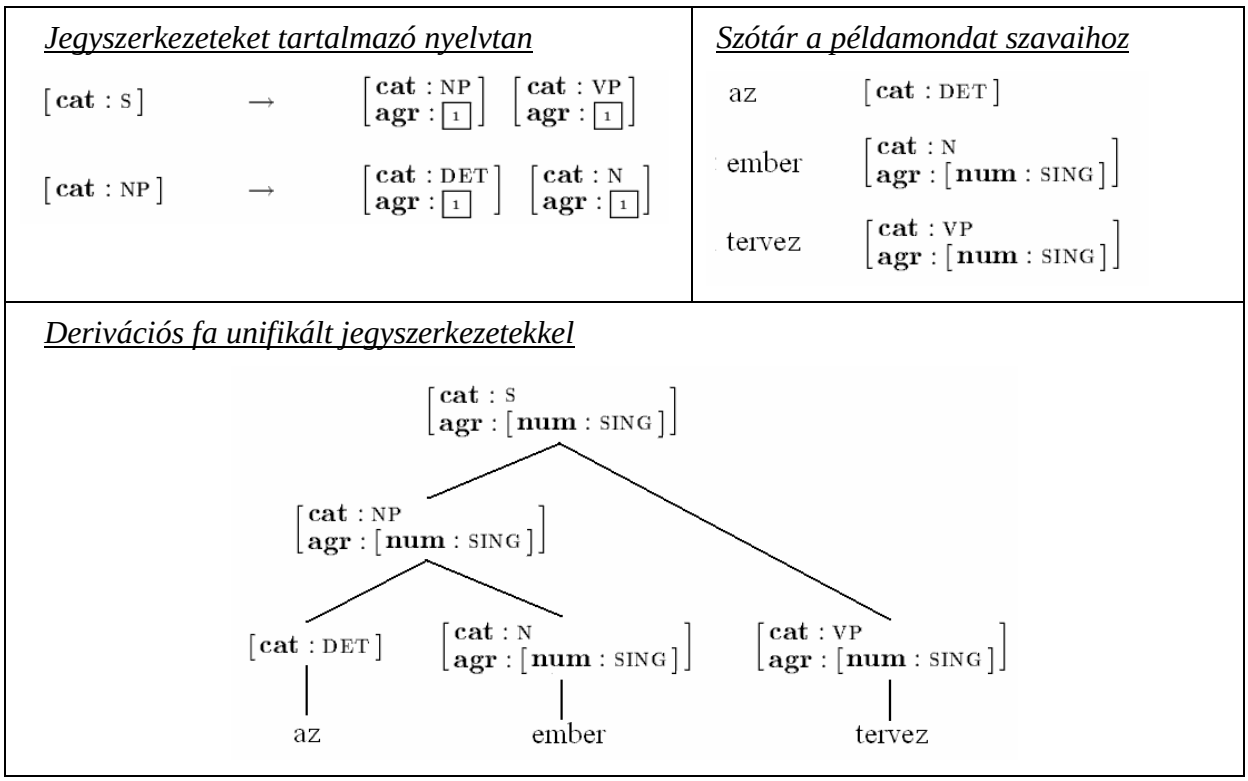
Az alkategórizálás problémája az egyeztetéshez hasonlóan nem oldható meg hatékonyan új szabályváltozatok bevezetésével, mert az a nyelvtan méretét a sokszorosára növelheti. Olyan megoldásra van szükség, amely nemcsak a nyelvi elemek sorrendje alapján azonosítja a nyelvi szerkezeteket, hanem ábrázolja az egyes elemekhez kapcsolódó bonyolultabb nyelvi információkat is és a nagyobb szerkezethez tartozó leírást az alkotóelemekhez tartozó leírásokból származtatja. Egy ilyen ábrázolási modell a ***jegyszerkezet (feature structure)***, mely az adatokat *<jegynév, érték>* párokba rendezi. Minden jegyszerkezet ilyen párok rendezett halmaza, mint például:

$$\begin{bmatrix} \textbf{major} : \text{NP} \\ \textbf{number} : \text{SINGULAR} \\ \textbf{person} : \text{THIRD} \\ \textbf{case} : \text{OBJECT} \end{bmatrix}$$

Ennél bonyolultabb jegyszerkezeteket is megadhatunk, melyben a jegyek leírását egy többszintű struktúra tartalmazza, sőt ez a struktúra lehet rekurzív is, az érték helyén újabb jegyszerkezet is állhat. A jegyszerkezet formális definíciója a következő:

2.6. Definíció: A ***jegyszerkezet*** egy olyan (Q, q_0, θ, δ) négyes, ahol Q a csúcsok véges halmaza, $q_0 \in Q$ a gyökér, $\theta : Q \rightarrow TYPE$ a csúcsok hozzárendelése a *TYPE* jegynév halmaz elemeihez, $\delta : FEAT \times Q \rightarrow Q$ az *FEAT* jegyérték halmaz elemeinek hozzárendelése a csúcsok közti átmenetekhez; $f \in FEAT$, q_1 , $q_2 \in Q$ esetén az átmenetet jelölése: $q_1 \rightarrow^f q_2$.

A jegyszerkezetekre épülő elemzési modellek jellemző művelete az ***unifikáció (egyesítés)***, mellyel az elemzett szócsoport címkéjéhez tartozó jegyszerkezetet a szócsoporthoz tartozó nyelvi elemek jegyszerkezeteiből számítjuk ki. Az unifikáció eredménye egy olyan jegyszerkezet, amelyben minden olyan jegy szerepel, amely a szócsoporthoz tartozó jegyszerkezetekben is megvolt. A konzisztencia az egyesítés művelete esetén fontos követelmény. Például, ha az egyesítendő jegyszerkezetek tartalmazzák ugyanazt az elemi jegyet (ugyanazzal a névvel és elérési úttal), de a különböző értékkel, akkor az unifikáció nem hajtható végre. A 2.7. ábrán egy jegyszerkezetekkel leírt CFG nyelvtan alkalmazásával történik az elemzés.



2.7. Ábra. Egy rövid példamondat elemzése unifikációs nyelvtannal.

A jegyszerkezetek unifikációjával kapott információk birtokában sok nyelvtanilag helytelen eset kizárható, ami csökkenti az elemzési erdő méretét és így növeli az elemzés hatékonyságát. Ugyanis ha a feltételezett nyelvi szerkezet elemeinek jegyszerkezetei nem unifikálhatók, a mondatelemző program a feltételezett szerkezetet nem ismeri fel a nyelv helyes mondatának (vagy részmondatának).

Az egyeztetés unifikációs nyelvtanokkal történő elemzés során az *AGREEMENT* jegy (a 2.7. ábrán *agr* a rövidítése) bevezetésével valósul meg, mely az unifikáció során konzisztencia vizsgálaton esik át. Ha valamely szócsoport tagjainak egyeztetendő jegyei értékben különböznek (például a *NUMBER* jegyek értékei *SINGULAR* ↔ *PLURAL*) akkor az unifikáció nem hajtható végre, így az adott szócsoportra vonatkozó szabály ebben az esetben nem alkalmazható.

Az alkategórizálás pedig például a *SUBCAT* jegy bevezetésével érhető el a HEAD jegyben, melynek a *TRANS*, *INTRANS*, … értékeivel korlátozzuk azt, hogy az adott ige csak a vonzatainak megfelelő *VP* szabályban szerepelhet, például:

Verb → *megy*
<*Verb* HEAD SUBCAT> = INTRANS

Verb → *csinál*
<*Verb* HEAD SUBCAT> = TRANS

VP → *Verb NP*
<*VP* HEAD SUBCAT> = INTRANS
<*VP* HEAD> = <*Verb* HEAD>

VP → *Verb NP NP*
<*VP* HEAD SUBCAT> = TRANS
<*VP* HEAD> = <*Verb* HEAD>

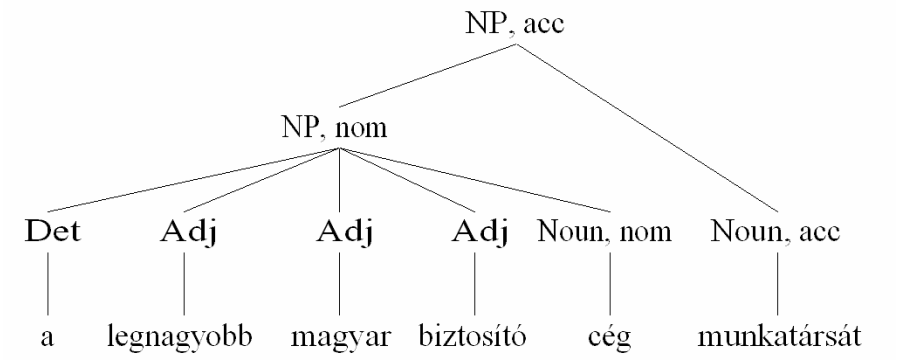
2.4. Faminta nyelvtanok

Ebben a fejezetben bemutatásra kerül a szerző által kidolgozott faminta nyelvtan, melynek bevezetését az indokolja, hogy a természetes nyelvek pontosabb szintaktikai elemzéséhez szükség nagyobb összefüggő szerkezetekre (részfákra). Ha a korpuszból vett nagyobb szerkezeteket alkalmazunk az elemzés során, a strukturális függőségek okozta problémák elkerülhetők. A korpuszban ténylegesen létező többszintű szerkezetek, megadják a kontextust a belső leírásukhoz, így nem fordul elő túlgenerálás.

Jellegénél fogva a faminta formalizmus a fa-összekapcsoló nyelvtanokhoz köthető abból a szempontból, hogy nagyobb, többszintű szerkezeteket tartalmaz. A különbség azonban az, hogy a fák összekapcsolása másképpen történik és a lexikalizált TAG formalizmussal ellentétben a faminták akár több szót is tartalmazhatnak.

2.4.1. Hasonló szerkezetek összevonása

Tegyük fel, hogy adott egy többszintű fa (2.8. ábra), továbbá a szavakhoz illetve szócsoportokhoz hozzá van rendelve azok esete is (nom - alany, acc - tárgy).



2.8. Ábra. Egy többszintű főnévi csoport

A fa által lefedett szócsoportokon mindenféle transzformációkat hajthatunk végre, elhagyhatunk, beszúrhatunk, átrendezhetünk és kicserélhetünk szavakat. Így az eredeti szócsoporthoz nagyon hasonló szócsoportokat kapunk, valamint látjuk azt, hogy hol vannak azok a pontok ahol variálhatjuk a leveleket, anélkül, hogy a magasabb szintek szerkezetén változtatni kellene. További hasonló esetek lehetnek például:

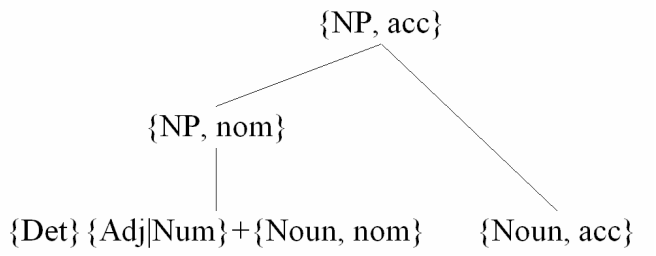
$a_{\{Det\}}$ legnagyobb$_{\{Adj\}}$ biztosító$_{\{Adj\}}$ cég$_{\{Noun,nom\}}$ munkatársát$_{\{Noun,acc\}}$

$a_{\{Det\}}$ $2_{\{Num\}}$ legnagyobb$_{\{Adj\}}$ biztosító$_{\{Adj\}}$ cég$_{\{Noun,nom\}}$ munkatársát$_{\{Noun,acc\}}$

$a_{\{Det\}}$ $2_{\{Num\}}$ cég$_{\{Noun,nom\}}$ munkatársát$_{\{Noun,acc\}}$

$az_{\{Det\}}$ első$_{\{Num\}}$ $2_{\{Num\}}$ cég$_{\{Noun,nom\}}$ munkatársát$_{\{Noun,acc\}}$

Az előző pontokban felsorolt eseteket lefedhetjük egyetlen famintával, ami még ráadásul általánosít is, mert lefedi az előzőekben fel nem sorolt eseteket is (2.9. ábra).



2.9. Ábra. A hasonló szerkezeteket lefedő faminta

A famintát zárójelezett formában is megadhatjuk:

$[_{NP,acc}\ [_{NP,nom}\ \{Det\}\ \{Adj|Num\}+\ \{Noun,nom\}]\ \{Noun,acc\}\ ]$

2.4.2. A faminta nyelvtan definíciója

A faminta nyelvtan mintaillesztési problémaként közelíti meg a mondatszintaxis felismerését. A faminta egy olyan derivációs részfa, ami két fő részre bontható: egy adott szimbólumsorozatra illeszkedő minta és az arra felépíthető egyértelműen meghatározott szintaktikai címkékből álló fa. A faminta nyelvtan formális definíciója a következő:

2.7. Definíció: A ***faminta nyelvtan*** egy olyan *(T, N, S, G, F)* ötös, ahol *T* a terminális szimbólumok véges halmaza, *N* a nemterminális szimbólumok véges halmaza, $N \cap T = \emptyset$, *S* a *start* szimbólum, $S \in N$, *G (T, N, S, …)* egy nyelvtan és *F* a *G* nyelvtannal előállítható elemzési szerkezetek véges halmaza, az alábbi tulajdonságokkal:

- A fa gyökérszimbóluma rögzített.
- A fa leveleit reguláris kifejezéssel írjuk le, melynek speciális operátorai, ’*’ (ismétlődés), ’+’ (pozitív számú ismétlődés), ’|’ (elágazás) és ’(’, ’)’ (zárójelezés). A reguláris kifejezésből kifejtett szimbólumsorozat maximális hosszára megadható egy előre definiált *K* korlát, például a ’+’ esetén is *K* lesz az ismétlődések maximális száma.
- A fa belső csúcsait a fa gyökérszimbólumából és a leveleire adott reguláris kifejezéssel leírt minta alapján számíthatjuk ki a *G* nyelvtan segítségével.
- Egy adott famintával felismert részfát önálló egységnek tekintjük, ezért csak annak gyökerére illeszthetünk újabb famintát, a belső csúcsaira nem. A belső csúcsokra elemzés közben nem hivatkozunk és nem végzünk velük műveletet.

2.4.3. Faminták változatai

A faminták tulajdonságai a definícióban szereplő *G* nyelvtan reprezentációjától függnek. Legegyszerűbb esetben G egy környezetfüggetlen nyelvtan. Ekkor a faminták egyszerűen terminális és nemterminális szimbólumokból álló reguláris kifejezések, például, a 2.9. ábrán látható szerkezetnek a következő felel meg:

$$[_{NP}\ [_{NP}\ \text{Det}\ \{\text{Adj}|\text{Num}\}+\ \text{Noun}\ ]\ \text{Noun}\ ]$$

Ha a szimbólumokat attribútumok halmazával jellemezzük, tehát a szófajon és szintaktikai címkén kívül a szavaknak illetve szócsoportoknak lehet például esetük és számuk, akkor a 2.9. ábrán láthatóhoz hasonló famintákat kapunk. A gyökérszimbólum az attribútumait a fej attribútumaiból örökli. A fej pozíciója többszintű szerkezet esetén

is pontosan meghatározható, ez például főnévi csoport esetén többszintű szerkezetben is az utolsó szó:

$$[_{\text{NP, eset=fej.eset, szám=fej.szám}} \ldots \text{Fej}_{\text{Noun, eset, szám}}]$$

A faminták esetén is értelmezni lehet a valószínűséget, melyet a PCFG-nél leírtaknak famintákra történő kiterjesztésével lehet bevezetni. Ha a gyökér valószínűségét a faminta belső elemzéséből számítanánk, akkor ugyanazt a valószínűséget kapnánk mintha PCFG-vel elemeznénk, de erről megmutattuk, hogy a strukturális függőséget nem kezeli, mivel a szabályalkalmazásokat független eseményeknek tekinti.

A feltételes valószínűség megállapítására alkalmazott relatív gyakorisággal történő becslés úgy módosul egy adott *F* faminta esetén, hogy egy alkalmas annotált korpuszon az *F* felismert előfordulásainak számát osztjuk az összes olyan faminta-előfordulás számával melyek gyökérszimbóluma *F*-évl megegyezik:

$$P(F \mid Root(F)) = \frac{Count(F)}{Count(Root(F))} \tag{2.5}$$

Egy *T* elemzés valószínűsége, amely a F_i (*i=1…n*) ismert valószínűségű faminták alkalmazásának sorozatával állt elő, a következő lesz:

$$P(T) = \prod_{i=1\ldots n} P(F_i \mid Root(F_i)) \tag{2.6}$$

Egy *S* mondat, melynek elemzéseinek halmazát $\mathfrak{I}(S)$ jelöli, akkor S legvalószínűbb elemzése $T_{max}(S)$ a következő lesz:

$$T_{\max}(S) = \arg\max_{T \in \mathfrak{I}(S)} P(T) \tag{2.7}$$

Ha lexikalizált famintákat szeretnénk alkalmazni akkor az attribútumos famintákhoz hasonlóan a fej pozícióját kell megadnunk a többszintű szerkezetekben ami a faminta gyökeréhez tartozó szót meghatározza. Ez többszintű szerkezetben is NP esetén az utolsó szó, VP esetén az ige, stb.:

$$[_{\text{NP, fej=munkatársát}} \ldots \text{munkatársát}_{\text{Noun}}]$$

$$[_{\text{VP, fej=meglátta}} \text{meglátta}_{\text{Verb}} [_{\text{NP, fej=munkatársát}} \ldots]]$$

A problémát a lexikalizálás során az jelenti, hogy a faminták még ritkábban fordulnak elő, mint az egyszintű szerkezetek ezért nehézséget jelent a valószínűségek hozzárendelése a lexikalizált famintákhoz. Ezért célszerű a szavakat csoportosítani szemantikai és morfológiai jellemzők alapján.

Jegyszerkezetek alkalmazása esetén az jelenti a gondot, hogy unifikáció végrehajtása lépésről lépésre halad, ugyanakkor a faminta belső csúcsaival nem végzünk műveleteket, hanem a mintaillesztést és az ezekhez tartozó unifikációs műveleteket a levelek szintjén szeretnénk végrehajtani. Ezért a belső unifikációs műveleteket ki kell

vezetni a levelek szintjére és el kell végezni a lehetséges összevonásokat. Akkor kerülhet sor összevonásra, ha például az egyeztetést több lépésben hajtottuk végre amíg elértük a gyökeret. Tekintsük például az alábbi szabályokat:

CP → *NP VP*
<*CP* AGREEMENT> = <*NP* AGREEMENT>
<*NP* AGREEMENT> = <*VP* AGREEMENT>

NP → *Det N*
<*NP* AGREEMENT> = <*N* AGREEMENT>

VP → *Verb*
<*VP* AGREEMENT> = <*Verb* AGREEMENT>

Az ezekből a szabályokból előállítható részfát szeretnénk összevonni egyetlen famintává, ekkor az összevont faminta az egyszerűsítések elvégzése után a következő lesz:

$[_{CP}\ [_{NP}$ *Det N* $]\ [_{VP}$ *Verb* $]\ ]$
<*CP* AGREEMENT> = <*N* AGREEMENT>
<*N* AGREEMENT> = < *Verb* AGREEMENT>

Végezetül megadhatók vegyes megoldások is, melyek az igei vonzatok szabad szórendjét függőségi nyelvtan írja le, míg azok alkotóelemeit a famináták valamelyik változatával ismerjük fel.

2.4.4. Chart parser alkalmazása famintákra

A 2.2.3 fejezetben ismertetett Chart parser egy eredményesen alkalmazható dinamikus programozási eszköz és szinte minden tulajdonságában felülmúlja a többi szintaktikai elemző algoritmust. Az alap algoritmus megszorítás nélkül alkalmazható általános környezetfüggetlen nyelvtanra, azon kívül rugalmasan kiegészíthető új, speciális tulajdonságokkal, melynek köszönhetően már számos nyelvtani formalizmusra implementálták.

Ezen szempontok miatt a famintákkal történő elemzésre a bottom-up chart parsert alkalmazhatjuk, ami az eredeti algoritmushoz néhány kisebb módosításával jár. Az egyik ilyen különbség, hogy a famináták mintaillesztése során a reprezentációtól függően lehetnek olyan technikai jelzések, melyek nem vesznek részt az illesztésben, ezeket egyszerűen csak át kell olvasni. Például egy pontozott faminta a következő lehet:

$[_{NP}\ [_{NP}$ Det • Noun] Noun]

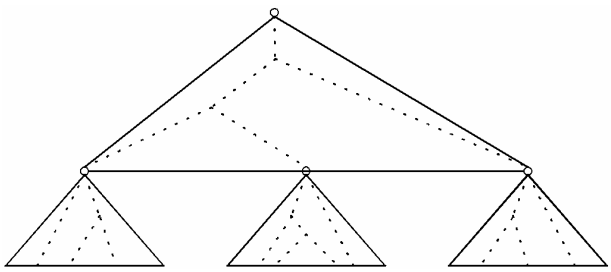
A famintákban levélpozíciókon nem csak egy szimbólum (szófaj vagy szintaktikai címke) szerepelhet, hanem összetett illesztési feltétel is, például többféle morfológiai jellemző (eset, szám, stb.). Másrészt a famináták reguláris kifejezésekkel leírt elemeket is tartalmazhatnak, például ismétlés ($[_{NP}$ Det Adj* Noun]) vagy választás ($[_{NP}$ Det Adj|Num

Noun]). Mivel ezek az illesztési műveletek a mintában és a mondatban pozíciónként előre haladva lineárisan végrehajthatók (analóg módon egy automata működtetésével), a chart parser mintaillesztési eljárása implementálható úgy, hogy megfeleljen ezeknek a szempontoknak.

Amikor a faminta leveleinek illesztése megtörtént, következhet egy új befejezett él beillesztése a chartba a faminta gyökérszimbólumával megcímkézve. Mivel bottom-up elemzés történik, minden levélelemhez rendelkezésre áll az összes szükséges információ, így ezen a ponton az új él beillesztése előtt különféle műveleteket végezhetünk:

- ***Egyeztetés:*** Lehetnek olyan szabályaink melyek azt írják elő, hogy a faminta egyes leveleinek például nemben, számban egyezniük kell. Ezek olyan feltételek, melyek a mintaillesztés során nem biztos, hogy feloldhatóak egyszeri átolvasással, a mintaillesztés végén azonban minden információ rendelkezésre áll a döntéshez. Amennyiben az illesztett faminta nem felel meg az egyeztetésnek, nem hozzuk létre az új befejezett élet.
- ***Öröklés:*** Lehetnek olyan szabályaink is, melyek a levélelemekből információkat hoznak fel és ezeket az új élhez kapcsolódóan szintén el kell tárolni. Ezek az öröklési szabályok szinte kizárólag a fej szimbólumra vonatkoznak, mivel az egy olyan kitüntetett elem a szócsoportban, melynek morfológiai jellemzői meghatározzák az egész szócsoport mondattani szerepét.
- ***Valószínűség:*** Az új élben szereplő gyökérszimbólum valószínűsége a levélelemek valószínűségének szorzata lesz. Miután kiszámítottuk a valószínűséget, a Viterbi módszert követve megvizsgálhatjuk, hogy van-e már olyan él a chartban ami ugyanazt a szócsoportot fedi le mint az új él ugyanazzal a szócsoporttal és a kisebb valószínűségűt elhagyjuk (illetve, ha ez az új él akkor nem vesszük fel).

A derivációs fában a famintákkal felismert részfák önálló egységek csak azok gyökerei és levelei kapcsolódhatnak egymáshoz, a belső csúcsok nem (2.10. ábra).



2.10. Ábra. A famintákkal felismert részfák egymáshoz kapcsolódása

A derivációs fában a faminták általában egy nagyobb összefüggő szintaktikai szerkezetet tartalmaznak, ezért ezek belső csúcsait nincs értelme más szerkezetekbe bevonni. Más szempontból ennek az elvnek köszönhetően az elemzési idő is csökkenthető.

2.4.5. Nyelvészeti alkalmasság és számítási hatékonyság

Kérdés lehet, hogy a faminta-formalizmus alkalmas-e a természetes nyelvek problémás jelenségének leírására, valamint, hogy bonyolultságelméleti szempontból milyen hatékonyságú algoritmusokkal lehet az elemzést faminták segítségével végrehajtani.

A faminták alkalmazásának előnye abban van, hogy az annotált korpuszban létező többszintű szerkezetekkel végzi az elemzést, de változatainak köszönhetően lehetőség van statisztikai információk hozzáadására, lexikális és távoli függőségek, valamint az egyeztetés és alkategórizálás ábrázolására.

A faminták többféle változatban is előfordulhatnak, melynek információtartalmát, jellegét, az is befolyásolja, hogy az annotált korpusz milyen lehetőségeket biztosít.

A faminták változatban is előfordulhatnak, melynek információtartalmát, jellegét, az is befolyásolja, hogy az annotált korpusz milyen lehetőségeket biztosít. A faminták alkalmazásának előnye abban van, hogy az annotált korpuszban létező többszintű szerkezetekkel végzi az elemzést, de változatainak köszönhetően lehetőség van statisztikai információk hozzáadására, lexikális és távoli függőségek, valamint az egyeztetés és alkategórizálás ábrázolására.

Kérdés lehet, hogy a faminta-formalizmus alkalmas-e a természetes nyelvek problémás jelenségének leírására, valamint, hogy bonyolultságelméleti szempontból milyen hatékonyságú algoritmusokkal lehet az elemzést faminták segítségével végrehajtani.

A famintákból álló nyelvtanokról belátható, hogy az enyhén környezetfüggő nyelvtanok közé sorolható. Mivel a reguláris kifejezések kifejtésének maximális hossza legfeljebb *K* lehet, a faminta nyelvtan átírható egy vele ekvivalens (például környezetfüggetlen) nyelvtanra. Így a kifejtett nyelvtan generatív váza segítségével az elemzés polinom időben elvégezhető, valamint a levezethető szavak hossza legfeljebb lineárisan növekszik. A faminták segítségével nagyobb részfákat is meg tudunk adni. Mivel a keresztező függőségeket maximális hosszára minden nyelv esetén megadható egy felső korlát, a keresztező függőségeket faminták segítségével be lehet építeni a nyelvtanba.

2.5. Konklúzió

Ebben a fejezetben áttekintettük, hogy a természetes nyelvek szintaktikai elemzése során milyen problémák jelentkeznek és ezeket hogyan lehet nyelvészeti és számítási szempontból hatékonyan megoldani. A fejezet végén a szerző ismertetett egy saját fejlesztésű szintaxisábrázolási módszer, a faminta formalizmust, mellyel nagyobb szintaktikai szerkezeteket lehet leírni és ezek segítségével a szintaktikai elemzést elvégezni.

3. Nyelvtani modellek előállítása gépi tanulással

Ebben a fejezetben áttekintjük azokat a gépi tanulási alapelveket és technikákat, melyeket szintaktikai elemző módszereinkben vagy azok alkalmazásaiban valamilyen szinten felhasználtunk. A fejezetben bemutatásra kerül egy saját fejlesztésű mintatanuló algoritmus az RGLearn [Hócza04a], melyet a komplex szintaktikai elemző módszerünkben faminták tanulására lehet alkalmazni.

3.1. A gépi tanulás általános módszerei

A számítógépek megjelenése óta az egyik legérdekesebb kérdés, hogy ezek képesek-e tanulni, azaz tudunk-e olyan rendszereket készíteni, amelyek működésük során képesek a rendelkezésre álló információk alapján automatikusan javítani a hatékonyságukat. Ha a gépi tanulást pontosabban akarjuk megfogalmazni, akkor az alábbi definíciót használjuk:

3.1. Definíció [Mitchel97]: Adottak a következők: *Z* egy számítógépes program, *E* tapasztalati tények (tréning adatok) egy halmaza, *T* elvégzendő (teszt) feladatok egy halmaza és *P* egy végrehajtási teljesítmény mérték. A *Z* program tanul az *E* gyakorlati tapasztalatokból, ha a *Z* programot az *E* gyakorlati példák feldolgozása után ismételten lefuttatva a *T* teszt feladatokon, a *Z* program *P* mérték szerinti teljesítménye javul.

A gyakorlati tapasztalatok, vagy más néven tanulási példák olyan (x_i, y_i) értékpárok halmaza, melyben az x_i értékek valamilyen objektum vagy esemény leírására szolgálnak, az y_i értékek pedig a következtetést adják meg. A tanulóprogram feladata egy olyan *f* függvény megkeresése, melyre $f(x_i) = y_i$ teljesül az x_i tanuló példákra. Azt a tanulást ***felügyelt (supervised)*** tanulásnak nevezzük, melyben feltesszük, hogy a tanulási példákat egy tanár (bölcs) biztosítja, pontosan meghatározva, hogy adott x_i esetén mi a helyes y_i. Ezzel szemben a ***nem-felügyelt (unsupervised)*** tanulás esetén a tanuló program csak x_i értékeket kap és neki kell keresnie ezekben a szabályosságokat. Van egy köztes tanulási módszer is, a ***megerősítéses (reinforcement)*** tanulás, melyben tanítás során nem adjuk meg az y_i értékeket, hanem a tanuló rendszer döntését utólag értékeljük, hogy az helyes volt vagy sem.

3.1.1. Felügyelt tanulás

A felügyelt tanulás esetén azzal a feltételezéssel élünk, hogy az *f* függvény egyedi példákból általánosítással történő előállítására egyszerű számítógépes modellt tudunk adni bizonyos pontosság mellett, valamint a tanító példahalmaz eléggé informatív az előbbi általánosítás elvégzéséhez. Egy adott *h* függvényt, ami x_i értékekhez y_i értékeket rendel ***hipotézisnek*** vagy ***modellnek*** nevezzük. A tanulás során általában egy keresési (optimalizálási) problémát kell megoldanunk: a példák általánosításával a lehetséges hipotézisek közül keressük azt a hipotézist, ami legjobban illeszkedik a tanulási példákra, azaz:

$$f = \arg\min_{h} \frac{\sum_{i=1..n} \left| h(x_i) = O(x_i) \right|}{n} \tag{3.1}$$

melyben *O* a bölcs (*oracle*) által megadott helyes hozzárendelés. Egy adott hipotézis megtanulásakor feltételezzük, hogy *f* függvény alkalmas lesz előre nem látott *x* értékek esetén is az *y* értékek helyes meghatározására. Ezt az elvet induktív tanulásnak nevezzük, melynek pontos definíciója a következő:

3.2. Definíció - *Induktív tanulási feltevés:* Ha a tanulási folyamat során találunk egy olyan *h* hipotézist ami jól megközelíti a keresett célfogalmat (célfüggvényt), akkor ez a *h* hipotézist előre nem ismert példákon is jól fogja közelíteni az adott fogalmat.

Abban az esetben, ha tanulási feladatban a következtetés (y_i) diszkrét értékű, akkor egy ***osztályozási*** problémáról van szó. Az ilyen problémák megoldására szolgáló statisztikai tanuló algoritmusokat **diszkriminatív** vagy ***generatív*** tanulóknak nevezik, attól függően, hogy mit modelleznek. A diszkriminatív modellek minden osztályt közös jellemző csoporttal írnak le, és arra törekednek, hogy egy-egy osztályt rendre elkülönítsenek a többitől. Ezt egy elválasztó függvény paramétereinek megfelelő beállításával érik el, vagy az osztályokat elemekkel és távolságfüggvény definiálásával reprezentálják. Ilyen diszkriminatív tanuló algoritmusok például a ***C4.5*** [Quinlan93] vagy a ***Mesterséges Neuronhálózatok (Artificial Neural Network, ANN)*** [Bishop95]. A generatív megközelítésben ahelyett, hogy közvetlenül az osztályok $P(y_j|x_i)$ valószínűséget modelleznénk, mint ahogy azt a diszkriminatív tanulók teszik, másik lehetőségként maximalizálhatjuk a megfigyelések $P(x_i|y_j)$ valószínűségét minden egyes osztályra külön-külön. Egy ilyen generatív tanuló algoritmus a ***Rejtett Markov Modell (Hidden Markov Model, HMM)*** [Rabiner89].

3.1.2. Nem-felügyelt tanulás

Jóllehet a felügyelt tanulást alkalmazó módszerek nagy hatékonyságra képesek, alkalmazásuk eléggé költséges és időigényes (pl. informatív tanító és teszt példák előállítsa, módosítása). Bizonyos feladatokra egyszerűbb módszerek is elegendőek, amelyek működése eléggé generikus tud lenni ahhoz, hogy automatikusan alkalmazhatók legyenek. A gépi tanulás ilyen egyszerűbb, automatikusan alkalmazható módszerei a nem-felügyelt tanuló technológiák. Itt a fő cél az, hogy a rendelkezésre álló adatbázisban korábban nem ismert mintákat, összefüggéseket találjanak.

A nem-felügyelt tanulásnak két jellemző módszere az ***csoportosítás*** és az ***asszociáció***. A csoportosítás vagy ***klaszterezés (clustering)*** feladata az, hogy az elemzendő adathalmazokat homogén diszjunkt részhalmazokra bontsuk. A homogenitást ebben az esetben a csoport elemeinek hasonlóságára alapozzuk. Az asszociáció – egyesítés vagy összekapcsoló elemzés – módszer célja általában az, hogy a tanuló példák közül kikeressék az összes olyan hasonlóságot, melyek nagy valószínűséggel ismétlődnek.

Az ***optimális partícionálási feladat*** formálisan úgy fogalmazható meg, hogy keressük valamely H halmaz megfelelő diszjunkt H_i részhalmazokra való bontását, a lehetséges diszjunkt felosztásokat Ω tartalmazza, a H_i részhalmaz súlyát valamely $F(H_j)$ halmazfüggvény segítségével definiáljuk, ekkor az optimális partícionálás $F(H)$ a következő:

$$F(H) = \arg\min_{D \in \Omega} \sum_{H_i \in D} F(H_i) \tag{3.2}$$

A klaszterezésnek alkalmazása különösen akkor ajánlható, ha a vizsgált adathalmaz nagy és áttekinthetetlenül bonyolult. A hasonló karakterisztikájú csoportokon belül esetleg már rá lehet érezni valamilyen heurisztikára, így alkalmazható lehet ott a felügyelt tanulás valamely módszere.

3.2. Szabályhalmazok tanulása

A tanuló algoritmusok alkalmazása során fontos szempont lehet, hogy a kialakult modell az adott terület szakértői (például nyelvészek) számára értelmezhető, karbantartható legyen. Ez a tény megnöveli a gépi tanulás során kialakult modell felhasználhatóságát, mivel következtetések vonhatók le belőle és lehetőség nyílik a modell továbbfejlesztésére szakértői háttértudás bevonásával. Ezért a szabályalapú rendszerek egy fontos osztályt alkotnak a tanuló algoritmusok között, mivel a bennük

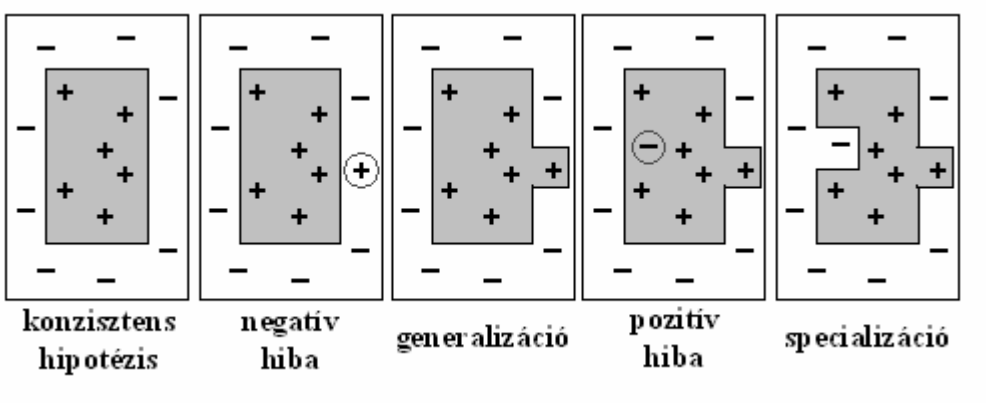
alkalmazott szabályok az ember számára közvetlenül olvashatók és megérthetők. Ezzel ellentétben, például a neuronhálózatokkal vagy más numerikus tanuló módszerekkel kapott modell csak a számítógép segítségével feldolgozható formában tárolható, ami az ember számára lényegében értelmezhetetlen számhalmaz. A szabályalapú rendszerek hátránya az, hogy a találati pontosság esetenként rosszabb, mint a numerikus tanuló módszerekkel elért eredmények. Ez a pontatlanság abból adódhat, hogy a szabályoktól általában azt várjuk el, hogy az átláthatóság érdekében néhány tulajdonság alapján vonjanak le következtetést, holott lehet, hogy a helyes következtetés valójában egy igen bonyolult függvény segítségével számítható ki, amire esetleg minden lehetséges tulajdonság hatással van kisebb vagy nagyobb súllyal.

3.2.1. Fogalom tanulása pozitív és negatív példák alapján

3.2. Definíció [Mitchel97]: ***Fogalom tanulásnak (concept learning)*** nevezzük az olyan tanulási feladatot, amelyben a feladat egy logikai értékű függvény megtanulása az input és output értékeket tartalmazó példák alapján.

Ebben az esetben a tanító példákat két diszjunkt részhalmazra lehet bontani: a ***pozitív*** és a ***negatív*** példák halmazára, attól függően, hogy az adott fogalommal kapcsolatos példáról vagy ellenpéldáról van szó. A pozitív és negatív példák együtt egy U halmazt (univerzum) alkotnak és ennek egy H^+ ($\subseteq$ U) részhalmaza reprezentálja az adott fogalomra vonatkozó H hipotézist leírása által lefedett példákat.

Egy H hipotézis akkor lesz konzisztens, ha pontosan el tudja különíteni egymástól a pozitív és negatív példákat, vagyis a leírása minden pozitív példát tartalmaz és nem tartalmaz negatív példát. Ha a konzisztencia nem teljesül, az két féle hibát jelenthet: a rendszer helytelenül állítja, hogy egy u elem nincs benne H^+-ban, illetve a rendszer helytelenül állítja azt, hogy benne van. Az első esetben a H hipotézisben megfogalmazott feltételrendszert úgy kell általánosítani, hogy ez után a hiányzó pozitív példa is beletartozzon H^+-ba, anélkül, hogy negatív példák is bekerülnének. A mennyiben hibásan lefedett negatív példa kizárása a cél, a feltételrendszert úgy kell specializálni (szűkíteni), hogy ezzel egy időben ne essenek ki korábban már jól lefedett pozitív példák.



3.1. Ábra. Hipotézisek hibalehetőségei és azok javítása

Abban az esetben, ha a H_1 hipotézis kevesebb korlátozást tartalmaz, mint a H_2 hipotézis és tudjuk, hogy minden olyan példa, ami teljesíti H_2 (szigorúbb) feltételeit az H_1 feltételeinek is eleget tesz, akkor azt mondhatjuk, hogy a H_1 hipotézis ***általánosabb*** mint a H_2, illetve, hogy a H_2 hipotézis ***speciálisabb*** mint a H_1. Ezek alapján a hipotézisek között bevezethetünk egy ≤ relációt:

3.3. Definíció: $H_1 \leq H_2 \;\Leftrightarrow\; \forall x \in U \;:\; H_2(x) = \uparrow \;\Rightarrow\; H_1(x) = \uparrow$

A ≤ reláció részleges rendezést ad meg a hipotézisek között, azaz lehetnek olyan hipotézisek melyek között egyik irányban sem teljesül a ≤ reláció. Amennyiben a hipotézisek leírását az $<a_1, a_2, \ldots, a_n>$ vektor adja meg, hipotézistér ≤ reláció szerinti legkisebb eleme (∅, ∅, …, ∅) a ***legspecifikusabb hipotézis*** (mivel leírásának egyetlen példa sem felel meg), legnagyobb eleme <∀, ∀, …, ∀> pedig a ***legáltalánosabb hipotézis*** (nem tartalmaz megszorítást, így annak minden példa megfelel).

3.2.2. Szintaktikai szabályok tanulása felügyelt gépi tanulással

Egy nyelvtani modell előállítása során úgy is definiálhatjuk a célfogalmat, hogy az erre megtanult hipotézist, az adott típusú szócsoportok, szintaktikai egységek felismerésére lehessen felhasználni. Ennek érdekében a tulajdonságvektor egy adott szócsoport belsejének és környezetének lehető legrészletesebb leírását tartalmazza, főleg morfoszintaktikai információkat felhasználva. A tulajdonságvektor segítségével megadhatunk pozitív és negatív példákat, azaz olyan eseteket melyekben a tulajdonságvektor a megtanulandó szócsoportot írja le és olyan eseteket is melyekben nem. A tanulás elvégzésére alkalmazhatunk egy tetszőleges felügyelt tanulásra alkalmas algoritmust, melynek az általunk megadott osztályozás két lehetséges esetére kell modellt készítenie.

Az egyik legismertebb osztályozást tanuló algoritmus az ID3 [Quinlan86] valamint a később ennek felhasználásával készült C4.5 tanuló rendszer [Quinlan93]. Az ID3

algoritmus döntési fák tanulására szolgál. A döntési fák tekinthetők a diszkrét változókon értelmezett diszkrét értékű függvények egy reprezentációjának. A döntési fákban három lényeges objektum különböztethető meg: ***levelek*** adják meg a formula kiértékelési értékét, a ***belső pontok*** a formula egy változója, csúcsok között ***élek*** pedig a változó lehetséges értékei. Az ID3 algoritmus legnagyobb előnye az, hogy az input adatok feldolgozásával egy lépésben elő tudja állítani a döntési fát.

Az algoritmus szempontjából a legfontosabb lépés a döntésbe bevonni kívánt attribútum kiválasztása. A kiválasztási stratégia az ***entrópia*** kiszámítására épül. A teljes példahalmaz (*S*) entrópiáját a lehetséges függvényértékek vizsgálata alapján tudjuk kiszámítani.

$$Entrópia(S) = \sum_{i=1}^{c} - p_i \log_2 p_i \tag{3.3}$$

Az entrópia akkor maximális (1), ha *S*-ben a pozitív és negatív példák száma megegyezik és akkor minimális (0) ha csak pozitív vagy negatív példák vannak *S*-ben. Az entrópia azt a szükséges információ mennyiségét (a bitek számát) jelöli, ami ahhoz kell, hogy a véletlenszerűen érkező példákhoz hozzárendeljük a függvényértékeket. Az ID3 algoritmus fontos eleme még az ***információs nyereség***. Ez azt mutatja meg, hogy egy *A* attribútum mennyire hatékony a példák szétválogatásakor.

$$Nyereség(S, A) = Entrópia(S) - \sum_{v \in Érték(A)} \frac{|S_v|}{|S|} Entrópia(S_v) \tag{3.4}$$

A fenti képlet azt mutatja meg, hogy mennyi az A attribútum-értékei szerint való osztályozás után az entrópiák különbsége. Az a cél, hogy ez a különbség minél nagyobb legyen, mivel ekkor a keletkező részhalmazok jobban szétválasztják a pozitív és negatív példákat.

A döntési fák ekvivalens "ha-akkor jellegű" szabályokká is könnyen átalakíthatók a következőképpen: ahány különböző levél van a döntési fában, annyi feltételes utasítást készítünk, amelynek a feltétel részéhez a levélbe vezető útnak megfelelő logikai formulát rendeljük (a szintenkénti relációkat az és logikai művelettel kapcsoljuk össze).

A tanuló adatok sokszor nem konzisztensek, ezért ha az egyedi példáknak túlzott jelentőséget tulajdonítunk akkor fellép a ***túlilleszkedés*** problémája. Egy $h \in H$ hipotézis túlilleszkedik a tréning adatokra, ha létezik egy olyan $h' \in H$ hipotézis, hogy a h hipotézisnek kisebb a hibája a tréning adatokon mint h'-nek, viszont a h' hipotézisnek kisebb a hibája a tesztpéldákon, mint h-nak. Az ID3 esetében a túlilleszkedés elkerülésére több módszert fejlesztettek ki, amelyek a megtanult döntési fa egyszerűsítésén, ***nyesésén (pruning)*** alapulnak. Az algoritmus először mindenképpen elkészíti (az esetleg túlspecializált) teljes döntési fát. A következő lépésben a döntési fát

szabályokká rendszerévé alakítja át. Ez után a szabályok fejében előforduló relációk közül egyet-egyet töröl, amennyiben az így kapott új szabály pontossága növekszik egy ún. validációs halmazon. A validációs halmaz lehet a tréning példák egy elkülönített része, vagy egy külön erre a célra megadott példahalmaz. Amikor már nem lehet újabb relációkat elhagyni a feltételes utasítások fejéből, akkor az algoritmus véget ér.

Az ID3 algoritmus hátránya, hogy mivel nincs visszalépésre lehetőség döntési fa nem feltétlenül optimális, másrészt a döntési fa elkészítésénél nem a kinyert szabályrendszer pontosságának maximalizálása a fő szempont, hanem az entrópia alapján a feltételek számának minimalizálása, azaz, hogy minél kevesebb csúcs érintésével lehessen eljutni a döntési fában a gyökértől a levelekig. Az attribútumok között lehetnek függőségek, relációk, de ennek megadására az algoritmus nem biztosít lehetőséget, így lehet, hogy az általunk elvárt attribútumok összefüggéseinek szempontjából véletlenszerű, nem értelmesen megfogalmazott szabályok keletkeznek.

3.2.3. Az RGLearn mintatanuló algoritmus

Ebben a részben bemutatásra kerül a szerző által kidolgozott RGLearn mintatanuló algoritmus, mellyel egy adott felismerési feladatra lehet egy összetett mintahalmazt előállítani. Az algoritmus alkalmazva volt faminták tanulására főnévi csoportok [Hócza04a], valamint teljes szintaxis [Hócza06a] esetén. A szófaji egyértelműsítés is visszavezethető a szavak környezetéből vett minták alkalmazására

Minden tanulási feladat tartalmazhat olyan sajátosságokat, melyek kezelése egy általános tanuló algoritmus számára túlságosan speciális, így a feladat megoldása során kompromisszumokat kell kötni, alkalmazkodva a tanuló algoritmus lehetőségeihez. Ilyen esetben az eredmények javítása érdekében szükség van a meglévő tanuló algoritmus kiegészítésére vagy a céloknak megfelelő új algoritmus elkészítésére. Például a faminta nyelvtan, amihez szükség lenne egy megfelelő tanuló algoritmushoz, az alábbi specialitásokat tartalmazza:

- Különböző hosszúságú minták vannak, a minták hosszát a részfákban előforduló levelek (szópozíciók) száma határozza meg.
- Egy adott szimbólumhoz jellemzői (attribútum-értékei) elkülönülnek a faminta többi szimbólumának jellemzőitől. Ezeket a tulajdonságvektorban különálló egységként kellene kezelni minden szópozícióra.
- Az attribútumok között prioritási sorrend van: $A > B$ azt jelenti, hogy egy adott szimbólumok általánosításánál *A* attribútumot csak *B* után szabad elhagyni. Egy attribútum elhagyása után általánosabb lesz a szimbólum (több esetet

lefed). Például szavak esetén a szófaj legnagyobb prioritású, a ragozott szóalak pedig a legkisebb prioritású attribútum, egy szó szimbólum legáltalánosabb alakja csak a szófajt tartalmazza. A prioritás a gyakoriságból is származhat: statisztikailag a szófaj gyakrabban előfordul mint a ragozott szóalak.

- A minta osztálya az adott szócsoportra felépíthető szerkezet. A minta legáltalánosabb alakja csak a szófajt vagy szócsoport címkét tartalmaz. A minta legáltalánosabb alakja által lefedett példa pozitív, ha a minta és a példa szerkezete megegyezik és negatív ha nem, a lefedetlen példákat nem kell figyelembe venni.

Az előzőekben vázolt funkciókat megvalósító algoritmus kifejlesztése után az RGLearn nevet kapta, melynek vázlata a 3.5. ábrán látható.

```
RGLEARN(Példák) returns faminták
   Példák rendezése a példák legáltalánosabb alakja alapján
   Minták = {}
   Példa = legelső példa
   Példacsoport = LEGÁLTALÁNOSABB-MINTA(Példa) szerint egyező példák
   while Példacsoport ≠ {} do
      Pozitív = Példacsoport helyes szintaxist tartalmazó példái
      Negatív = Példacsoport hibás szintaxist tartalmazó példái
      Fok = 0
      Új-minták = LEGÁLTALÁNOSABB-MINTA(Pozitív)
      while EREDMÉNYJAVULÁS(Fok, Új-minták, Pozitív) do
         Specializált-minták = SPECIALIZÁLÁS(Fok, Új-minták, Pozitív)
         Új-minták = LEGJOBB-FEDÉS(Specializált-minták, Pozitív, Negatív)
         Fok = Fok + 1
      Minták ← Új-minták
      Példa = következő LEGÁLTALÁNOSABB-MINTA(Példa)-val nem egyező példa
      Példacsoport = LEGÁLTALÁNOSABB-MINTA(Példa) szerint egyező példák
   return Minták
```

3.5. Ábra. Az RGLearn algoritmus vázlata

Egy minta foka azt jelenti, hogy a minta a pozitív példák legáltalánosabb alakjához viszonyítva hány darab behozott attribútumot tartalmaz. A *SPECIALIZÁLÁS (Fok, Új-minták, Pozitív)* eljárás megvizsgálja, hogy *Új-minták* halmaz elemei milyen *Pozitív* példákat fednek le és ezután kiválasztja azt összes prioritási sorban következő attribútum értéket, melyek segítségével előállítja az összes lehetséges specializált,

Fok+1 fokú mintát. A *LEGJOBB-FEDÉS(Specializált-minták, Pozitív, Negatív)* először statisztikát készít a minták fedéséről a *Pozitív* és *Negatív* példákon és ezek alapján pontszámot (s*core*) rendel minden mintához és visszatér azzal a mintahalmazzal, melyben minden elem valamely *Pozitív* példának a legmagasabb pontszámú fedője. Az *EREDMÉNYJAVULÁS (Fok, Új-minták, Pozitív)* értéke akkor igaz, ha az *Új-minták* halmazban van legalább egy *Pozitív* példa amit a korábbinál magasabb pontszámú új minta fed le.

Egy adott minta pontszáma azt a mértéket adja meg, hogy a tréning példákon a minta alkalmazásával mennyire pontosan lehet elválasztani a pozitív példákat a negatívaktól, a „jó" minta csak pozitív példát fed le. A mérték pedig egy olyan 0 és 1 közötti folytonos értékű függvény, ami akkor éri el a maximumát (1-et) amikor csak pozitív példákat fed le a minta. Különféle mértékekből 1 összegű súlyok lineáris kombinációjával újabb összetett mértékeket lehet előállítani, mint például az alábbi:

$$score = \lambda_1 * (pos - neg) / pos + \lambda_2 * pos \ / \ (pos + neg) \qquad (3.5)$$

ahol a *pos* a lefedett pozitív példák száma, *neg* a lefedett negatív példák száma, valamint $\lambda_1 + \lambda_2 = 1$. Különböző λ_i értékekkel az algoritmus különböző szempontoknak megfelelő mintahalmazt állít elő, így ezek olyan paraméterei lehetnek az algoritmusnak, melyet a tanulási feladat jellegének megfelelően lehet beállítani vagy optimalizálni.

Az RGLearn algoritmus általánosító és specializáló operátorainak megváltoztatásával reguláris kifejezések tanulására is alkalmas eljárás készíthető. Az általánosítás művelete úgy is definiálható, hogy csak a minta szerkezete maradjon meg (még a szavak szófaja is eldobható). Ezek után a specializálás a minta szerkezetének egy adott pozíciójára akár többféle levélelemet is behozhat, azaz lehetővé válik a "|" (vagy) reguláris operátor alkalmazása. Továbbá a specializálás során egy új levélelem behozása az első lépésben történhet a "+" (ismétlés) operátorral, azt feltételezve, hogy az adott típusú levélelem akár többször is ismétlődhet egymás után, majd ezek után lehet egy következő specializáló lépés a "+" elvétele. Mindezek mellett alkalmazva a specializálás alapműveletét (az attribútumok behozatalát pozitív példák alapján), a mintapopuláció tartalmazni fog reguláris kifejezésekkel leírt elemeket is, melyek közül a legjobb hibapontszámot elért minták lesznek kigyűjtve.

3.2.4. Mintaalapú POS-tagger felkészítése az RGLearn algoritmussal

Az IKTA 27/2000 K+F projekt célja egy hatékony szófaji egyértelműsítő módszer és egy ezt implementáló program prototípus kifejlesztése volt magyar nyelvre gépi tanulási algoritmusok felhasználásával. Ennek a projektnek a keretében az RGLearn algoritmus

egy alkalmazásaként a szerző elkészített egy mintaalapú szófaji egyértelműsítőt (POS-tagger), mely összehasonlításra került a szerző társai által alkalmazott más módszerekkel [Kuba03] és [Kuba04].

A mintaalapú POS-tagger az adott szópozícióhoz rendelhető szófaji címkéket a szópozícióra és annak környezetére illeszthető minták alapján választja ki. A mintahalmaz előállítását végző RGLearn pozitív és negatív példák alapján tanul. A pozitív példákat a korpusz annotált mondataiból kinyerhető tulajdonságvektorok adták, melyek a szavak és azok előre megadott ablakméreten belüli szókörnyezetének morfoszintaktikai információit (szótő, szófaj, szám, eset, stb.) tartalmazták, valamint az ezekhez rendel szófaji címkét. Egy adott pozitív példához tartozó negatív példák azok az esetek, melyeket a pozitív példa legáltalánosabb alakja (melyben a környezet szavainak csak a szófaja szerepel) lefed, de a hozzájuk rendelt szójai címke eltérő.

A [Kuba04] cikkben szereplő további módszerek közül az egyik egy HMM alapú statisztikai módszer, a TnT [Brants00] volt, a másik pedig a modell pontosságát transzformációs szabályokkal finomító módszer a TBL [Brill95] volt. A legjobb eredményt a szerző társai által alkalmazott többségi szavazásos módszer érte el, melyben mind a három tagger jóslata részt vett. Az kiértékelések eredményei a 3.1 táblázatban szerepelnek.

Szövegtípus	TnT	TBL	RGLearn	Szavazás
Általános szövegek	96.18%	96.52%	94.54%	96.78%
Üzleti hírek	98.83%	98.26%	97.32%	98.50%
Vegyes szövegek	97.31%	95.79%	96.81%	97.81%

3.1. Táblázat: Az NP-felismerés kiértékelése általános szövegeken és üzleti híreken

3.3. Modellek optimalizálása korpusz alapú módszerekkel

Amikor a modellünk elegendően nagy, a vele történő elemzés során többértelműség léphet fel, melyek közül valamilyen módszerrel választanunk kell. Ehhez megfelelő matematikai hátteret biztosít a statisztikai valószínűség alkalmazása. A statisztikai elkészítéséhez szükség van egy elegendően nagy korpuszra, melyben megfelelő számú példát találhatunk a modellünk elemeinek alkalmazására. A cél egy korpuszra legjobban illeszkedő valószínűségi modell előállítása, azaz amellyel végzett automatikus elemzést kiértékelve a korpuszon maximális értéket kapunk.

3.3.1. Nyelvtanok valószínűségi modelljének tanulása

Miután valamilyen módszerrel előállítottunk egy szabályhalmazt és rendelkezünk elemzett korpusszal a szabályaink előfordulási statisztikája alapján meghatározhatjuk a valószínűségi modellt, mellyel kiszámíthatjuk egy annotáció valószínűségét. Legyen C az X annotációk előfordulási gyakoriságát leíró függvény, azaz C egy korpuszt definiál, továbbá M egy valószínűségi modell X-en melynek, $P \in M$ egy lehetséges példánya. Ha a korpusz annotációja egy adott G nyelvtan segítségével történt, akkor egy $x \in X$ ***annotáció valószínűsége*** a következő:

$$P(x) := \prod_{r \in G} P(r)^{C_r(x)} \tag{3.6}$$

Mivel ugyanazt a P valószínűségi modellt alkalmazzuk a korpusz összes annotációjára, ezek felhasználásával definiálhatjuk a ***korpusz valószínűségét*** is:

$$L(C,P) := \prod_{x \in X} P(x)^{C(x)} \tag{3.7}$$

Az L függvényt ***likelihood*** függvénynek is nevezzük. Olyan valószínűségi modellt szeretnénk találni, amely alapján a korpuszunk leginkább megfelel az elvárásainknak, azaz maximális valószínűségű. Az ennek a szempontnak megfelelő $\tilde{P}$ valószínűségi modellt ***maximum likelihood*** becslésnek nevezzük:

$$L(C,\tilde{P}) := \arg\max_{P \in M} L(C,P) \tag{3.8}$$

Megmutatható (részletes bizonyítás: [Prescher03]), hogy a korpusz valószínűsége akkor lesz maximális, ha a valószínűségi modellt a korpuszbeli ***szabály-előfordulások relatív gyakoriságával*** becsüljük, azaz:

$$\forall A \to \beta \in G \ : \ \tilde{P}(A \to \beta) := \frac{C(A \to \beta)}{\sum_{A \to \alpha \in G} C(A \to \alpha)} = \frac{C(A \to \beta)}{C(A)} \tag{3.9}$$

Amennyiben nem rendelkezünk szintaktikai annotációt tartalmazó korpusszal, de rendelkezünk egy rögzített nyelvtannal (pl. szakértői szabályok) és egy szöveges korpusszal, akkor a korpuszt elemezve a szabályok alkalmazhatóságának gyakorisága alapján tudunk valószínűségi nyelvtant készíteni. Ez úgy történik, hogy leelemezzük a korpusz mondatait a nyelvtanunk segítségével, miközben megszámoljuk a szabályaink előfordulási gyakoriságát és az így kapott statisztika alapján valószínűséget rendelünk a szabályainkhoz. Ez azonban nem megy olyan egyszerűen, mint az annotált korpusz esetén alkalmazható maximum likelihood becslés esetén, mert míg egy annotált korpusz egyúttal szintaktikailag is egyértelműsített, egy elegendően általános nyelvtannal történő elemzés tartalmazhat többértelműséget. Ebben az esetben több iterációs lépésben lehet meghatározni a szabályok valószínűségét úgy, hogy minden lépésben

egyértelműsítjük az aktuális szabályvalószínűségek alapján a szöveges korpusz mondatainak elemzését, elkészítjük az új szabály előfordulási statisztikást és ez alapján újraszámoljuk a valószínűségeket. Ez az eljárás egy olyan valószínűségi modellhez konvergál, mellyel a (3.7) formula, azaz a szöveges korpusz becsült valószínűsége maximális lesz.

Az előbb vázolt számítási módszert az ***Inside-Outside algoritmus*** [Baker79] valósítja meg, ami a ***Forward-Backward algoritmus*** [Baum72] egy valószínűségi nyelvtanokra alkalmazott változata. Mindkét módszer speciális esete az ***EM algoritmus*** [Dempster77] néven ismert nem-felügyelt tanítási módszernek.

3.3.2. A HMM tagger működése és modelljének előállítása

A természetes nyelvek elemzésében vannak olyan feladatok, melyek visszavezethetők ***címkézési (tagging)*** problémára. Ilyen lehet például a szófaji egyértelműsítés (POS tagging), melyre számos természetes nyelvi feladat épül, például a szintaktikai elemzés is. Egy másik alkalmazási terület lehet maga a szintaktikai elemzés abban az esetben, ha egyszerűsíteni tudunk a feladatokon, például a felszíni elemzés (shallow parsing) során elegendő ha a szócsoportok határait meg tudjuk jósolni.

A ***Rejtett Markov Modellekben*** **(HMM)** [Rabiner89] a megfigyelés az állapotok valamilyen sztochasztikus függvénye, így a belső állapotok rejtve maradnak. Egy ilyen modellre épül a címkézési feladatot megoldó ***HMM tagger*** [Charniak93], ami egy adott mondat $W = w_1 \ldots w_n$ szavaihoz hozzárendel egy $T = t_1 \ldots t_n$ címkesorozatot, melyre:

$$\tilde{T} := \arg\max_{T \in \mathrm{T}} P(T \mid W) \tag{3.10}$$

A Bayes szabály alkalmazásával és $P(W)$ konstans valószínűség elhagyásával a maximalizálandó kifejezés $P(W \mid T)\ P(T)$ lesz, melynek a kifejtése és egyszerűsítése a következő:

$$\begin{aligned} \mathrm{P(W \mid T)\ P(T)} &= \prod_{i=1}^{n} P(t_i \mid w_1 t_1 \ldots w_{i-} t_{i-1}) P(w_i \mid w_1 t_1 \ldots w_{i-} t_{i-1} t_i) \\ &\approx \prod_{i=1}^{n} P(t_i \mid t_{i-N+1} \ldots t_{i-1}) P(w_i \mid t_i) \end{aligned} \tag{3.11}$$

Melyben N (=1,2,3,..) értéke alapján unigram, bigram, trigram modellről beszélünk. Ez például trigram esetén azt jelenti, hogy minden szópozíción 3 címke (az aktuális és az előző kettő) alapján képezzük a valószínűséget (A mondat elején valamint unigram esetén értelemszerűen elhagyjuk a felesleges tagokat a szorzatból.) A szorzat akkor lesz

maximális, ha szópozíciónként (láncszerűen) maximalizálunk, így egy tagra az alábbi kifejezést kapjuk:

$$t_i := \arg\max_j P(t_j \mid t_{i-N+1}...t_{i-1})P(w_i \mid t_j) \tag{3.12}$$

A valószínűségeket a korpusz alapú módszereknél alkalmazható relatív gyakoriságok alapján becsüljük:

$$P(t_i \mid t_{i-N+1}...t_{i-1}) \approx \frac{C(t_{i-N+1}...t_{i-1}t_i)}{C(t_{i-N+1}...t_{i-1})} \tag{3.13}$$

$$P(w_i \mid t_1) \approx \frac{C(w_i, t_1)}{C(t_1)} \tag{3.14}$$

3.3.3. Paraméterek optimalizálása szimulált hűtéssel

A mesterséges intelligenciában a problémák egyik jellegzetes tulajdonsága, hogy a megoldásukhoz próbálkozásokon, kereséseken keresztül juthatunk el. Ezekben az esetekben a keresési tér egy adott modell input paramétereinek a lehetséges értékeiből áll, a feladat pedig egy olyan paraméterbeállítás megkeresése, amellyel az előállított modell egy alkalmas kiértékelő függvénnyel mért jósága maximális. Az optimalizáló probléma reprezentálható egy irányított gráffal, melyben a gráf csomópontjai a különböző állapotokat jelölik, a gráf éleinek pedig az állapotok változását előidéző operációk felelnek meg. Kiemelt fontossága van a kiinduló állapotnak és a célállapotnak, melyekből több is lehetséges. Számos kereső algoritmus született az ilyen jellegű feladatok megoldására, ezek egy része az adott állapot környezetéből kiválasztott legjobb megoldások mentén halad. Ez a stratégia gyorsan találatot ad, de a feladat összetettségének függvényében lokális optimumban ragadhat.

A globális optimum megtalálására széleskörűen alkalmazzák a ***szimulált hűtés (simulated annealing)*** [Aarts89] algoritmust, mely az anyagi részecskét kristályállapotba rendezésének analógiáján alapszik. Egy adott, kristályba rendezhető testet felmelegítve az egyes részecskéknek megnő a mozgási energiája. A test fokozatos hűtésével minimalizálni tudjuk a test energiafüggvényét, mivel az, hogy a részecskék az elméleti kristályszerkezet csúcspontjaitól milyen távol lévő pontokban „ragadnak be", attól függ, hogy milyen ütemben végeztük a hűtést. Az ennek analógiájára kifejlesztett algoritmus a 3.6. ábrán található.

```
SZIMULÁLT-HŰTÉS() returns megoldás
   T, M, M_best = INICIALIZÁLÁS()
   while not LEÁLLÁS (T, M_best) do
      M = ÚJ-MEGOLDÁS(M)
      ΔE = E(M_best) - E(M)
      if (ΔE < 0) or
         (ΔE ≥ 0 and e^(-ΔE/λT) > random[0,1]) then M_best = M
      T = L(T)
   return M_best
```

3.6. Ábra. A szimulált hűtés algoritmusa

Az algoritmus a keresési tér diszkrét pontjaiban mozogva keresi egy alkalmasan választott E energiafüggvény minimumát. Ez az energiafüggvény általában a modellt kiértékelő függvény által megállapított hiba, de egyéb információkat is tartalmazhat. Ahhoz azonban, hogy véges lépésen belül is egy elég jó megoldást találjunk, megfelelő hűtési ütemtervet kell találni. Ha a kezdeti hőmérsékletet T_0 jelöli, a $T_{k+1} = L(T_k)$ értékadás határozza meg a következő iterációs lépésben alkalmazott hőmérsékletet, melyben L adja meg a hűtési ütemtervet. A $\Delta T = T_{k+1} - T_k$ hőmérséklet-csökkenések lehetnek lineárisak vagy egyre csökkenő mértékűek, ez utóbbi gyorsíthatja a folyamatot. Bebizonyítható, hogy elegendően kis ΔT-k melletti hűtési ütemtervvel az algoritmus 1 valószínűséggel megtalálja a globális optimumot. Ha az új megoldás jobb, akkor elfogadjuk, ha nem akkor csak bizonyos valószínűséggel fogadjuk el melybe T egy λ skála-konstanssal megszorozva veszünk figyelembe.

3.3.4. Módszerek kombinációja

A módszerek kombinációja egy hatékony eszköz arra, hogy feljavítsuk az egyedi módszerek eredményeit. Jelöljön x egy példát $\{v_1 \ldots v_n\}$ pedig a példákhoz rendelhető osztály címkéket az $f_i(x, j)$ pedig az i-edik osztályozó x példa esetén a j-edik osztályra adott kimenetét. Az i-edik osztályozó – mivel más információ nem áll rendelkezésére – azt az osztályt választja, ahol a legnagyobb kimenetet kapta:

$$v_k \ : \ k := \arg\max_j f_i(x, j) \tag{3.15}$$

Legyen $F(x, j)$ egy kombinációs módszer ami N darab osztályozó esetén határozza meg az x példa esetén a j-edik osztályra adott kimenetét, ekkor szintén a legnagyobb kimenetet választjuk:

$$v_k \ : \ k := \arg\max_j F(x, j) \tag{3.16}$$

A szakirodalomban [Jain00] számos kombinációs módszer (szabály) szerepel, ezek közül néhány:

Összeg szabály: $$F(x, j) := \sum_{i=1}^{N} f_i(x, j) \tag{3.17}$$

Szorzat szabály: $$F(x, j) := \prod_{i=1}^{N} f_i(x, j) \tag{3.18}$$

Max szabály: $$F(x, j) := \max_{i=1}^{N} f_i(x, j) \tag{3.19}$$

Min szabály: $$F(x, j) := \min_{i=1}^{N} f_i(x, j) \tag{3.20}$$

A felsorolt módszerek nem tesznek különbséget az osztályozók között, mindegyik egyforma súllyal szerepel. Ha az osztályozókhoz hozzárendelünk λ_i normalizált súlyokat ($\lambda_1 + ... + \lambda_n=1$), akkor például az összeg szabályból az alábbi új szabályt kapjuk:

Súlyozott összeg: $$F(x, j) := \sum_{i=1}^{N} \lambda_i f_i(x, j) \tag{3.21}$$

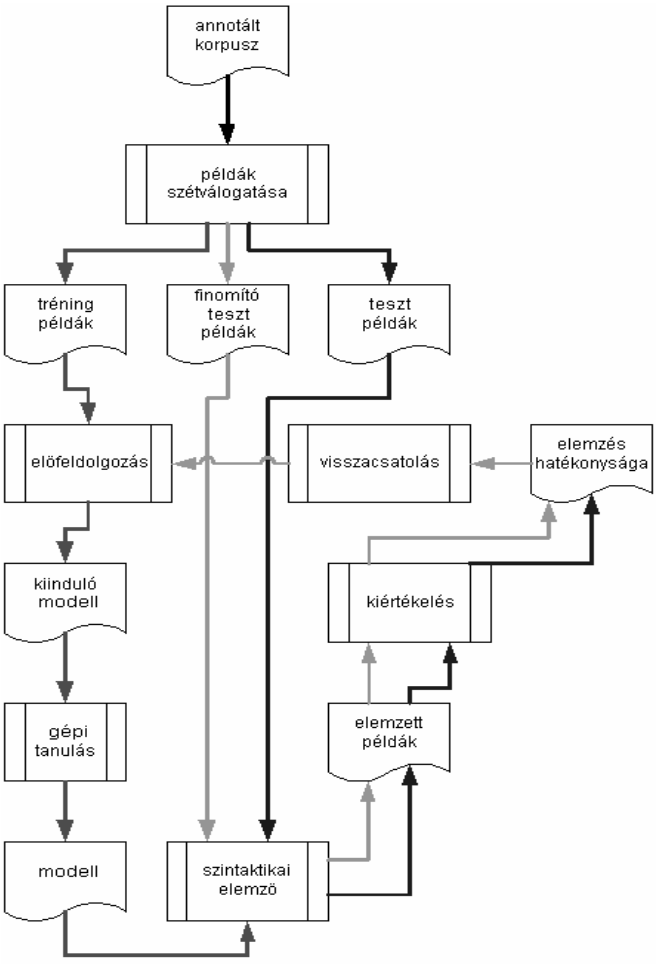
További javulás érhető el az eredményekben, ha súlyozott összeg súlyait a példák egy részén kiértékelés alapján optimalizáljuk valamilyen módszerrel, például erre alkalmas lehet a szimulált hűtés.

3.4. Konklúzió

Ebben a fejezetben áttekintettük azokat a gépi tanulási elveket és módszereket, melyeket a szerző az általa kidolgozott szintaktikai elemző módszerekben, vagy annak alkalmazásaiban valamilyen szinten felhasznált. Bemutattunk a szerző által kifejlesztett RGLearn mintatanuló működését, valamit annak alkalmazását szófaji egyértelműsítésre. A korpusz alapú módszerek alapvető eleme a gépi tanulási technikák alkalmazása. Bemutattuk hogyan lehet a nyelvtan szabályaira egy valószínűségi modellt előállítani létező korpusz esetén az előfordulási gyakoriság felhasználásával. A címkézés (tagging) módszerei alapvető fontosságúak a szintaktikai elemzésben, mivel annak bemenete általában egy szófajilag egyértelműsített szöveg, de a felszíni elemzés során is alkalmazhatunk taggert a szócsoportok határainak meghatározására. Szimulált hűtés segítségével optimalizálhatjuk a modellünk paramétereit, hogy az annotált szövegeken kiértékelve maximális felismerési pontosságot érjen el. További javulás érhető el, ha több módszer eredményeit kombináljuk és még több ha a módszereinket súlyozzuk is.

4. Faminta alapú komplex szintaktikai elemző módszer

Az automatikus szintaktikai elemzés megvalósítása több önálló részproblémára bontható. A feladat végrehajtás érdekében szükség van egy elegendően nagyméretű, nyelvész szakértők által annotált korpuszra, melyből a gépi tanulásra és kiértékelésre alkalmas példák halmazát lehet kinyerni. A korpusz adatait tréning és teszt halmazokra lehet felosztani, melyből a tréning halmaz a gépi tanulási modell előállításában játszik szerepet, míg a teszt halmaz példái melyek a megtanult modell számára ismeretlenek, a kiértékelésben vesznek részt. Finomító tesztpéldák alkalmazhatók a tanulási módszer paramétereinek optimális beállításának megtalálásához, valamint a tesztelés során szerzett tapasztalatok alapján a módszer továbbfejlesztéséhez. Ez a visszacsatolási folyamat akár többször is végrehajtható, amíg a minden szempontból optimális modell elő nem állt. A végső teszt ezután következhet. A szintaktikus elemzési modell előállításának, finomításának és kiértékelésének folyamatábrája a 4.1. ábrán látható.



4.1. Ábra. A szintaktikai elemzést megvalósító gépi tanulási modell előállításának és kiértékelésének folyamata.

Ebben a fejezetben bemutatásra kerül egy olyan szintaktikai elemző módszer, amely a 2.4 fejezetben felvázolt faminta nyelvtani formalizmusra épül, nyelvtan előállítása pedig a 3.2.3 fejezetben leírt RGLearn mintatanuló algoritmussal történik. A szintaktikai elemzés végrehajtása a 2.2.3 fejezetben részletezett chart parsing technika faminták elemzésére alkalmas továbbfejlesztett változatával történik. A szintaktikai elemzés kiértékelő módszerének kidolgozásával kapott mérőszám lehetőséget biztosít a gépi tanulás paramétereinek beállítására és végrehajtási módjának optimalizálására, azaz a modell finomítására.

4.1. Faminták tanulása az RGLearn algoritmussal

A faminták tanulására túlságosan körülményes felügyelt tanulási módszert alkalmazni, mert az túl sok kötöttséggel és kompromisszummal jár, mivel a tanuló példák megtervezésénél meg kellene adnunk egy tulajdonságvektort és az erre vonatkozó következtetést, de pont arra lennénk kíváncsiak, hogy mik a mondatszintaxis leírására alkalmas minták és ezt milyen tulajdonságok alapján lehet felismerni. Valamint az is érdekes lehet, hogy ezek a minták hogyan alkotnak olyan rendszert (modellt), melyet a optimalizálni lehet a modellt alkalmazó szintaktikai elemző algoritmus működését kiértékelve. Ezek alapján a faminta tanulási probléma inkább a nem-felügyelt módszerrel megoldható feladatok közé sorolható. Ilyen megfontolások vezettek a 3.2.3 fejezetben bemutatott RGLearn mintatanuló algoritmus kidolgozásához.

Ahhoz, hogy faminta formalizmust gyakorlatban is alkalmazni lehessen, ki kellett dolgozni egy komplex módszert melynek egy eleme az RGLearn algoritmussal végzett mintatanulás. A minták kinyeréséhez szintaktikailag annotált korpuszra van szükségünk melyet Angol nyelvre a Penn Treebank [Marcus93], Magyar nyelvre pedig a Szeged Treebank [Csendes05] biztosít. A korpusz jellege és információtartalma erősen meghatározza a belőle kinyerhető minták felhasználásával végezhető szintaktikai elemzés. Például ha az annotáció csak a környezetfüggetlen frázisstruktúrát tartalmazza, ebből fejvezérelt elemzést, távoli függőségeket és jegyszekezeteket direkt módon nem lehet kinyerni.

A tanuló algoritmus futtatása előtt az annotált korpusz formátumában rendelkezésre álló információt elő kell készíteni (*preprocesszálni*) a tanuló algoritmus számára feldolgozható formára hozni. Mivel a kiinduló állapot mondatok teljes szintaktikai elemzése, ezt le kell bontani olyan elemi részekre melyekből a gépi tanulás során a nyelvelméletünknek megfelelő minták származhatnak. Ezen a ponton szakértői tudást kell bevonnunk az elemi részekre vonatkozó kritériumaink megfogalmazásával és az itt hozott döntéseink erősen kihatnak a továbbiakban előálló modellünkre.

4.1.1. Faminták kigyűjtése faalak-típusok segítségével

A kiinduló szintaxis-elemekre vonatkozó kritériumainkat egy formalizált, számítógéppel feldolgozható szabályrendszerben adhatjuk meg, amely az előfeldolgozás nyelvelméletünk szerinti elvek alapján történő elvégzéséhez kell. Magával a kritériumrendszerrel szemben is vannak elvárásaink, mégpedig az, hogy különítsen el az mondatszintaxison belül feltételezhetően önálló objektumnak tekinthető nagyobb szerkezeti egységeket. Ezek az elkülönített objektumok egymásba ágyazódhatnak és teljesen le kell bontaniuk egy tetszőleges szintaxisfát. Az objektumok (részfák) több szint mélységűek is lehetnek, de ne legyenek túlságosan összetettek, mivel ezek túl specifikusak, túl kevés esetet fednének le. Ugyanakkor a másik véglet sem jó, amikor a kinyert részfák maximum egy szintűek, mert akkor lényegében egy CFG nyelvtant kapnánk vissza.

Egy ilyen kritériumrendszert megadhatunk faalak-típusok bevezetésével. A 4.2 ábrán bemutatott példa mindössze két faalak-típust tartalmaz. A ***beágyazás*** egy olyan többnyire rekurzív szerkezet, ami egynél több szint esetén úgy keletkezik, hogy a belső fához, balról vagy jobbról hozzácsapódik néhány terminális egy-egy újabb fa-szintet alkotva. Legfeljebb egy belső fa lehet benne. A ***füzér*** pedig egy olyan legfeljebb 2 szintű szerkezet amiben a lebontatlan részfák maximum 1 mélységűek és 1 szélességűek lehetnek. Ezek vagy felsorolás jellegű szerkezetek, vagy a már lebontott részfák összefűzései egy szerkezetbe. Az egyszintű szintaxisfa lebontása (ami egy $A \rightarrow \beta$ alakú CFG szabálynak felel meg) a meghatározás alapján a beágyazás kategóriába sorolható.

<u>*Példamondat:*</u>

$[_{CP}\ [_{NP}\ [_{NP}\ \text{Mihály}_{Noun}\ ]\ \text{és}_{Conj}\ [_{NP}\ \text{az}_{Det}\ \text{ügyvéd}_{Noun}\ ]\ ]\ [_{VP}\ \text{felkereste}_{Verb}\ ]$
$[_{NP}\ \text{a}_{Det}\ [_{ADJP}\ \text{budapesti}_{Adj}\ ]\ \text{egyesület}_{Noun}\ ]\ \text{elnökét}_{Noun}\ ]\ .{}_{Punct}\ ]$

<u>*Kinyerhető minták:*</u>

füzér: $[_{NP}\ [_{NP}\ \text{Mihály}_{Noun}\ ]\ \text{és}_{Conj}\ [_{NP}\ \text{az}_{Det}\ \text{ügyvéd}_{Noun}\ ]\ ]$

beágyazás: $[_{VP}\ \text{felkereste}_{Verb}\ ]$

beágyazás: $[_{NP}\ [_{NP}\ \text{a}_{Det}\ [_{ADJP}\ \text{budapesti}_{Adj}\ ]\ \text{egyesület}_{Noun}\ ]\ \text{elnökét}_{Noun}\ ]$

<u>*A mondat leírása a kinyert minták behelyettesítése után:*</u>

$[_{CP}\ \text{NP VP NP}\ .{}_{Punct}\ ]$

<u>*Kinyerhető minták:*</u>

beágyazás: $[_{CP}\ \text{NP VP NP}\ .{}_{Punct}\ ]$

4.2. Ábra. Faalak-típusokkal végzett faminta-gyűjtés egy annotált példamondatból.

A faalak-típusok között lehetnek átfedések, ezért ezek között fel kell állítanunk egy prioritási sorrendet, melyet a > relációval jelölve a 4.2 ábrán alkalmazott beágyazásra és a füzérre a következő:

$$beágyazás_{k+1} > beágyazás_k > \ldots > beágyazás_2 > füzér > beágyazás_1 \quad (4.1)$$

Beágyazás esetén az alsó indexben szereplő szám a fa mélységére utal. A nagyobb mélység magasabb prioritása azt jelenti, hogy a lehető legmélyebb beágyazásokat szeretnénk kinyerni. Ez alól kivétel lehet, ha bevezetünk egy *K* korlátot, aminél mélyebb részfát már túl bonyolultnak tartunk. Lehetőség van a kigyűjtött faminták – mint szócsoportok – különböző típusait is figyelembe venni a prioritási sorrend megállapításánál, például:

tagmondatok > igei szerkezetek > főnévi csoportok (4.2)

Azonkívül egy adott szócsoportnak is lehetnek olyan alcsoportjai melyeknél számíthat a kigyűjtés sorrendje, például a főnévi csoport feje alapján lehetnek birtokos, részes, tárgyas, stb. szerkezetek

Természetesen bevezethetünk több faalak-típust is az erre vonatkozó kritériumok betartásával, melyek segítségéve a különféle mondattani egységek árnyaltabban kinyerhetők. Ezeknél szintén el kell végezni a fentebb vázolt prioritási beállításokat.

4.1.2. A kiinduló szabályrendszer

A faalak-típusok segítségével kigyűjtött faminták halmaza már tekinthető szabályrendszernek, mivel a szintaxisfák lebontásának folyamatát megfordítva akár vissza is építhetjük az eredeti szintaxisfát, sőt ismeretlen mondatokra is megpróbálhatjuk alkalmazni. Ezzel a kiinduló szabályrendszerrel kapcsolatban az a probléma, hogy túl sok mintát tartalmaz, másrészt a minták túl speciálisak, ugyanis például a szavak illesztésénél minden lehetséges feltételnek egyeznie kell (szótő, szófaj, eset, szám, stb.). Ezekkel a mintákkal az ismeretlen mondatok szintaktikai elemzése túl sok időt venne igénybe, ennek ellenére nagyon sok mondatnál nem tudnánk teljes szintaxisfát felépíteni.

Ezen a ponton szakértői szabályokkal is kiegészíthetjük a kiinduló szabályrendszerünket. Kísérleti jelleggel megkértük az annotálást végző nyelvészeket, hogy fogalmazzák meg szabály formájában azt, hogy az adott esetben miért úgy kell az annotálást elvégezni, ahogy azt megtették. Az így kialakult a szabályokon végzett statisztika alapján olyan problémák derültek ki, hogy ezek egy része túl általános, más része pedig túl speciális vagy nem terjed ki minden esetre. A probléma abból adódik, hogy szemantikai információk alapján többnyire egyértelmű annotálási lépéseket kellett

úgy átfogalmazni, hogy csupán morfológiai jegyeket lehetett használni. Ez a művelet információvesztéssel jár, amely során valamilyen mértékű elemzési hiba keletkezik, melyet még valahogy minimalizálni is kellene fejben lefuttatott statisztikai számításokkal.

A kiinduló szabályrendszer ugyan lehetőséget biztosít arra, hogy szintaktikai elemzést végezzünk vele, de ismeretlen mondatokon ez nem fog elfogadható minőséget adni. A megoldást a szabályrendszer optimalizálása adhat. Ez egyrészt a szabályrendszer tömörítését jelenti, ugyanis egymáshoz hasonló minták a különbséget jelentő egyéni specialitásainak elhagyásával összevonhatók. Azonban különböző mértékű általánosítással kapott szabályrendszerrel az elemzés pontossága is különböző lesz, ezért meg kell találni az általánosítás és specializálás elemzési pontosság szerinti optimumát.

4.1.3. Faminták általánosítása és specializálása

Az alárendelés relációt alkalmazhatjuk famintákra:

4.1 Definíció: A ***faminták alárendelése (subsumption)***, azaz $F_1 \subseteq F_2$ akkor áll fenn, ha F_2-ből bizonyos morfológiai vagy szintaktikai információk elhagyásával megkapjuk F_1-et (illetve F_1-ből bizonyos morfológiai vagy szintaktikai információk hozzáadásával megkapjuk F_2-t).

A definícióban szereplő F_2 bővebb információtartalmú, azaz ***speciálisabb***, ezért kevesebb esetet fed le, mint az ***általánosabb*** F_1 faminta. Egy túl általános faminta sokszor olyan mondatrészt is lefedhet, melynek szintaktikai szerkezete eltér a felismerni vélttől. Egyre általánosabb famintákat alkalmazva megnő a hibásan felismert szintaktikai szerkezetek száma. A túl speciális faminták viszont kevés esetre alkalmazhatók, ezáltal sok olyan esettel találkozhatunk, amire egyáltalán nincs faminta. Ez is megnövelheti a hibák számát azzal, hogy elmarad a helyes szintaxis felismerése. Valahol középúton kell megtalálni az optimális megoldást olyan szabályrendszerrel, ami se nem túl általános, se nem túl speciális.

Az ***általánosítás*** során veszünk egy mintát, és sorban elhagyjuk belőle a különféle morfológiai jegyeket (fej, szófaj, eset, szám, stb.) a ***specializálás*** pedig egy ezzel ellentétes művelet, egy általánosított mintát az annotált korpuszban talált példa alapján kiegészítjük. A szavak szófaja illetve a nemterminálisok szintaktikai címkéje nem hagyható el, mivel ezek a legfontosabb információk, valamint ha mindent elhagyhatnánk az értelmetlen szabályokat eredményezne. Az is általánosítás, ha bevezetünk általánosabb morfológiai kategóriát, mivel például szótő esetén túl nagy

ugrás lenne az általánosításban ezek elhagyása. A szótő-csoportok alkalmazása egy köztes lépést jelenthet. Az egyik lehetőség erre, hogy ha rendelkezünk szemantikai hálóval (pl. WordNet [Fellbaum98]) a szótövet annak hipernímiájával helyettesítjük, a másik pedig az, hogy a szófajokat felosztjuk a szintaktikai szempontból releváns alcsoportokra (pl. tranzitív és intranzitív igék) és a szótőt ezekkel helyettesítjük. A általánosítás harmadik csoportja a hasonló szintaktikai szerkezetek összevonása amely részfák elhagyásával történhet. Tehát azt mondjuk, hogy *A* és *B* szintaktikai szerkezetek közül *A* általánosabb mint *B*, ha *B*-ből bizonyos részfákat elhagyva *A*-t kapunk. Az általánosítás különféle eseteire a 4.3. ábrán láthatunk példákat.

Morfológiai jegyek általánosítása:

$[_{NP}\ [_{NP}\ az_{Det}\ egyesület_{Noun,Nom}\ ]\ elnöke_{Noun,Gen}\ ]$

$[_{NP}\ [_{NP}\ (az,Det)\ (egyesület,Noun,Nom)\ ]\ (elnök,Noun,Gen)]$

$[_{NP}\ [_{NP}\ (Det)\ (Noun,Nom)\ ]\ (Noun,Gen)]$

$[_{NP}\ [_{NP}\ (Det)\ (Noun)\ ]\ (Noun)]$

Helyettesítés általánosabb kategóriával:

$[_{VP}\ [_{NP}\ a_{Det}\ kutya_{Noun,Nom}\ ]\ kergeti_{Verb}\ [_{NP}\ a_{Det}\ macskát_{Noun,Acc}\ ]\ ]$

$[_{VP}\ [_{NP}\ névelő_{Det}\ élőlény_{Noun,Nom}\ ]\ tranzitív\_ige_{Verb}\ [_{NP}\ névelő_{Det}\ élőlény_{Noun,Acc}\ ]\ ]$

Szintaktikai szerkezet általánosítása:

$[_{NP}\ [_{NP}\ a_{Det}\ [_{ADJP}\ legnagyobb_{Adj}\ biztosító_{Adj}\ ]\ cég_{Noun,Nom}\ ]\ munkatársát_{Noun,Acc}\ ]$

$[_{NP}\ [_{NP}\ a_{Det}\ [_{ADJP}\ legnagyobb_{Adj}\ ]\ cég_{Noun,Nom}\ ]\ munkatársát_{Noun,Acc}\ ]$

$[_{NP}\ [_{NP}\ a_{Det}\ cég_{Noun,Nom}\ ]\ munkatársát_{Noun,Acc}\ ]$

$[_{NP}\ munkatársát_{Noun,Acc}\ ]$

4.3. Ábra. Példák különféle típusú általánosításra.

A példákban szereplő általánosítási lépések többféle sorrendben pozíciónként külön-külön is végrehajthatóak. Ha egy adott *A* és *B* faminta között nem teljesül az $A \subseteq B$ alárendelés ezeknek lehet olyan *C* őse melyre teljesül $C \subseteq A$ és $C \subseteq B$ is. Így képezhetjük a minták általánosításának hierarchiáját. Ha megengedjük az üres mintát is, akkor ez a hierarchia egyetlen nagy irányított gráf lesz, melynek az üres minta lesz a kiinduló csúcsa, és tartalmazni fogja az összes mintát a kiinduló szabályrendszerünkből, valamint ezek összes lehetséges általánosítását. Az elemi transzformációs lépések lesznek a gráf élei. A gráfban megkereshetünk egy minimális utat az egyik mintától a másikig, és azt mondhatjuk, hogy a minimális út elemi lépéseinek a száma egy távolság mérték *A* és *B* faminta között:

$$Távolság(A,B) := \arg\min_{U \in Lehetséges\_utak(A,B)} |U| \qquad (4.3)$$

Ez a képlet azonban nem veszi figyelembe, hogy az elemi lépések nem egyforma súlyúak, például a szótő elhagyása drasztikusan megnöveli a minta fedőképességét, a szófaj elhagyásának nincs túl sok értelme, a szintaktikai általánosítás pedig egész más kategória mint a morfológiai jegyeké. Ezért az elemi lépéseket fel kell osztanunk *N* darab részcsoportra (pl. legegyszerűbben: morfológiai és szintaktikai elemi lépések) és az így kapott *euklideszi tér* lehetséges irányait súlyozva (skálázva) a következő módon adhatjuk meg a távolság mértéket:

$$Távolság(A,B) := \sqrt{\sum_{i=1}^{N} (\lambda_i * \underset{U \in Lehetséges\_i\_típusú\_utak(A,B)}{\arg\min} |U|)^2} \qquad (4.4)$$

Ha egy *A* famintát általánosítunk *B*-re (vagy *B*-t specializáljunk *A*-ra), akkor *B* lefedi az összes *A* által lefedett esetet (és ennél többet is melyek között lehetnek hibás fedések is). Egy annotált korpusz felhasználásával meghatározhatjuk a famintát és annak általánosabb alakja közötti $B \subseteq A$ alárendelés mértékét a korpuszban való előfordulás, illetve az általánosabb faminta esetén a helyes fedések számának felhasználásával melyet *C*()-vel jelölünk:

$$\|B \subseteq A\| := 1 - \frac{C(A)}{\sum_{A_i : B \subseteq A_i} C(A_i)} = 1 - \frac{C(A)}{C(B)} \qquad (4.5)$$

A 4.5 formula értéke egy 0 és 1 közötti szám. Az 1-hez közeli érték nagyon erős általánosítást jelent, például ha *B* az üres faminta, a másik véglet pedig $\|A \subseteq A\|$, azaz ha egyáltalán nincs általánosítás. Ennek felhasználásával is lehet távolsági mértéket megadni, melyet *A* és *B* faminták *C* legspeciálisabb közös általánosított famintájának megkeresésével kapunk meg:

$$Távolság(A,B) := \underset{C:\ \substack{C \subseteq A, \\ C \subseteq B}}{\arg\min} \frac{\|C \subseteq A\| + \|C \subseteq B\|}{2} \qquad (4.6)$$

4.1.4. A faminta tanuló módszer algoritmusa

A faminták általánosításával tömöríthetjük a kiinduló szabályrendszert, valamint a fedőképesség növelésével nő az ismeretlen mondatokon való alkalmazhatóság esélye, de a fő szempont az, hogy megtaláljuk azt az optimális szabályrendszert mellyel a szintaktikai elemzés során a legnagyobb pontosságot érhetjük el. A kiinduló szabályrendszer túlságosan specifikus, egy nagyon általános szabályrendszer esetén pedig túl sok lesz az illesztési hiba, ezért az optimum a két véglet között van, tehát valóban egy optimalizálási (tanulási) feladatot kell megoldanunk.

A tömörítés úgy valósul meg, hogy amikor egy *A* famintát általánosítunk *B*-re, kidobjuk az *A*–t és vele együtt az összes olyan A_1 ... A_n famintát melyre $B \subseteq A_i$ teljesül. Ezzel lényegében egy csoportosítást (klaszterezést) végzünk el, mivel az új szabályrendszer általánosított B_1 ... B_m famintái felbontják a kiinduló szabályrendszer famintáit diszjunkt részhalmazokra, ahol minden egyes A_{i1} ... A_{in} kiinduló faminta csoporthoz pontosan egy B_i általánosított faminta tartozik, mint az A_{ij} famináták közös általánosítása, valamint a B_i famináták között nincs átfedés. A 3.1.2 fejezetben már említett *optimális partícionálási feladat* elemeit a faminta tanulásra alkalmazva meg kell találnunk a kiinduló szabályrendszer olyan B_1 ... B_m famintákra történő általánosítását Ω lehetséges általánosítás közül, melyekre egy alkalmasan választott *Érték*(B_i) függvény optimális eredményt ad:

$$Optimális\ általánosítás = \arg\max_{B \in \Omega} \sum_{B_i \in B} Érték(B_i) \tag{4.7}$$

A súlyfüggvény megállapításánál szerepet kaphat az előző fejezetben ismertetett távolságfüggvény, mivel azt szeretnénk, hogy egymáshoz közel lévő famintákat csoportosítsunk össze. Ez azonban nem elég, mert a csoportokon belüli közelség akkor a legnagyobb, ha egyáltalán nincs általánosítás. Szükség van olyan szempontokat is felvenni az értékfüggvénybe, melyek az általánosítást részesíti előnyben és így a távolságra vonatkozó kritérium a túlzott általánosítást fékezheti. A több szemponton alapuló értékfüggvény a következő:

$$Érték(B_i) := \sum_{k=1}^{K} \lambda_k * Szempont_k(B_i) \tag{4.8}$$

A $Szempont_k(B_i)$-k a különféle szempontokat és követelményeket megvalósító, a tanító korpusz méretére invariáns, 0 és 1 közé normalizált valós értékű függvények, a λ_k számok pedig a szempontok súlyai ($0 \le \lambda_k \le 1$, $\sum \lambda_k = 1$). A λ_k súlyok helyes beállítása egy külön optimalizálási probléma, melyet egy alkalmas optimalizáló módszerrel (pl. szimulált hűtés) lehet meghatározni egy kisebb finomító teszt korpuszon a szintaktikai elemzés pontosságának maximalizálásával. A lehetséges szempontok függvényei például a következők:

- $$Közelség(B_i) := 1 - \max_{B \subseteq A_p, A_q} Távolság(A_p, A_q) \tag{4.9}$$
- $$Precizitás(B_i) := \frac{Pos(B_i)}{Pos(B_i) + Neg(B_i)} \tag{4.10}$$
- $$Fedés(B_i) := \frac{Pos(B_i)}{MAX\_POS} \tag{4.11}$$

A $Pos(B_i)$ a B_i faminta pozitív, azaz helyes fedéseinek számát, a *MAX_POS* konstans pedig ennek a felső becslését jelöli az aktuális tanító korpuszra, a $Neg(B_i)$ pedig azokat a

negatív azaz, hibás fedések számát jelöli, amikor a B_i faminta illeszthető egy B_i-étől eltérő szerkezettel annotált mondatrészre. Minden elemi általánosítási lépés behozhat újabb pozitív és negatív fedéseket. Az a jó általánosítás, ami úgy hoz be sok új pozitív fedést, hogy közben kevés új negatív fedés keletkezik. A *Precizitás* abban az esetben is nagy lehet, ha a 4.11 formulában szereplő *Fedés* kicsi, de sok ilyen minta esetén a szabályrendszer fedőképessége is kicsi lesz, emiatt elemzési hibák keletkeznek és ez jelentkezni fog a λ_k súlyok optimalizálásánál. Így a helyes arány megtalálásával nagy precizitású és fedésű, egymáshoz közeli eseteket lefedő famintákat fog megtanulni a rendszer.

Mivel a teljes korpuszon nagyon sok különböző kigyűjtött részfa előállhat, ennek a nagy adattömegnek az együttes kezelése komoly technikai problémát jelentene. Ezért faminta általánosítás irányát megfordítjuk. Nem a speciálisból megyünk az általános felé – mely folyamat során számolnunk kellene annak az esélyével, hogy bármely két minta összecsoportosítható egy közös általánosított faminta alá – hanem vesszük a korpuszból kigyűjtött faminták általunk engedélyezett legnagyobb általánosítását, és ezekből kiindulva specializálunk. Ezzel a lépéssel felosztjuk a nagy összevont faminta halmazt sok kisebbre (melyeket egy-egy legáltalalánosabb faminta lefed) és egyszerre csak egy részhalmazt dolgozunk fel. Ha rendezzük a faminta halmazt a legáltalánosabb alak szerint, akkor a rendezett lista szekvenciálisan feldolgozható.

A faminták tanulását a 3.2.3 fejezetben ismertetett RGLearn algoritmussal végezzük el, mely rendezi a mintahalmazt és a legáltalánosabb alak szerint feldolgozva a mintacsoportokat specializál, miközben minden új famintához egy pontszámot rendel (jelen esetben *Érték*(B_i)-t) és ez alapján eltárolja a legjobb famintákat. A pontszám *Precizitás* függvényéhet szükség van negatív példákra is, ezért miután előállítjuk a faminták legáltalánosabb alakjainak listáját, ezek segítségével az annotált korpuszból kigyűjtjük a hibás illesztéseket is a feldolgozhatóság érdekében legáltalánosabb alak szerint sorba rendezve és csoportosítva. Az előfeldolgozás során nemcsak kigyűjtjük a famintákat és a hibás fedéseket, hanem meg is számoljuk ezeket, a faminta tanulás során pedig kihasználjuk azt, hogy egy specializált faminta csak azokból a példákból fedhet le, melyet a nála általánosabb példa lefed. Így a faminták értékeléséhez használt pontszámot az aktuálisan betöltött pozitív és negatív példák felhasználásával, a lefedett példák statisztikai adataiból számítjuk ki, ami lényegesen gyorsabb és pontosabb mintha az annotált korpusz egy részén kellene elvégezni a kiértékelést.

4.1.5. Statisztikai információk hozzáadása a modellhez

Amikor előállt az optimális szabályrendszer, utófeldolgozási lépésként ezekhez hozzá rendelünk egy valószínűségi értéket, hogy a szintaktikai elemzés során, többértelműség esetén ki tudjuk választani a legjobb megoldást. A előző fejezetben ismertetett feldolgozási módszerben tanulás közben is rendelkezésre állnak a szükséges statisztikai adatok, így a valószínűség hozzárendeléséhez nincs szükség utófeldolgozásra.

Mivel a faminták tanulásához előfeltétel egy megfelelően nagy annotált korpusz megléte, feltételezhetjük, hogy ez statisztikai információk gyűjtéséhez is megfelelő hátteret biztosít. Ebben az esetben a 3.3.1 fejezetben ismertetett maximum likelihood becslés alkalmazható, melyben szereplő relatív gyakoriság egy adott *F* faminta esetére a következőképpen adaptálható:

$$\tilde{P}(A) := \frac{C(A)}{\sum_{B\,:\,Gyökér(B)=Gyökér(A)} C(B)} = \frac{C(A)}{C(Gyökér(A))} \qquad (4.12)$$

4.1.6. Többszintű szabályrendszer

Többszintű szabályrendszer alkalmazása a nemzetközi szakirodalomban is előfordul [Abney96], melyben reguláris kifejezésekkel leírt, többfokozatú (kaszkád) szabályrendszert alkalmaznak. Az Szeged Korpusz annotálásánál használt CLaRK rendszer [Simov01] is támogatja a többszintű szabályok alkalmazását.

A faalak-típusok ismertetésénél bevezetett prioritási sorrend alkalmazható általunk fontosnak tartott nagyobb mondattani egységek elkülönítésére, mint például főnévi csoportok, igei szerkezetek és tagmondatok. Ezzel a megoldással egymástól elkülönülő szabálycsoportok keletkeznek, melyek között nincs átfedés. A szabályok átláthatóbbak, karbantarthatóbbak lesznek és többféle feladatra is felhasználhatjuk ezeket, például felszíni és teljes szintaktikai elemzést is végezhetünk ugyanazzal a szabályrendszerrel a megadott beállításoktól függően.

A 4.4. ábrán látható algoritmus ami elvégzi a szintek szerinti példagyűjtést tartalmaz *GYŰJTÉS* eljárása kigyűjti az aktuális szint kritériumainak megfelelő új példákat az annotációból. A *LEBONTÁS* eljárás a kigyűjtött példák azaz részfák gyökér-szimbólumát hagyja csak meg.

```
SZINTEK-SZERINTI-GYŰJTÉS(Korpusz, Szintek) returns példák
   Példák = {}
   foreach Annotáció of Korpusz do
      foreach Szint of Szintek do
         Új-példák = GYŰJTÉS(Annotáció, Szint)
         Annotáció = LEBONTÁS(Annotáció, Új-példák)
         Példák ← Új-példák
return Példák
```

4.4. Ábra. A szintek szerinti példagyűjtés vázlata

4.2. Szintaktikai elemzés famintákkal

A szintaktikai elemzéshez választott formalizmus megadja a probléma bonyolultsági osztályát. Azonban a szintaxis elemző algoritmus is fontos része a rendszernek, hiszen közvetlenül ennek működésével összefüggésben mérhető le a rendszer hatékonysága, azaz a sebessége és a pontossága.

4.2.1. A famintákra alkalmazott chart parser

A 2.2.3 fejezetben ismertetett Chart parser egy hatékonyan alkalmazható dinamikus programozási eszköz. Az alap algoritmus rugalmasan kiegészíthető új, speciális tulajdonságokkal, melynek köszönhetően már számos formalizmusra implementálták.

Ezen szempontok miatt a famintákkal történő elemzésre a bottom-up chart parsert alkalmaztunk a 2.4.4. fejezetben leírtaknak megfelelően. Ehhez az eredeti algoritmusnak csak néhány kisebb változtatására volt szükség. Egy lényeges eltérés a CFG vázra épülő formalizmusokhoz képest, hogy a faminták által meghatározott belső csúcsok nem vesznek rész a magasabb szintű szerkezetek illesztésénél, mivel azt feltételezzük, hogy a faminták jól meghatározott részfa-objektumokat különítenek el a mondaton belül. Ez technikailag úgy valósul meg, hogy a mintaillesztés egyszerűen átolvassa a belső részfák határaira vonatkozó jelzéseket, melyeket csak akkor használunk fel ha rekonstruálnunk kell a teljes szintaxisfát. Ez azt jelenti, hogy az illesztett famintákból álló elemzési fának kevesebb szintje van, mintha teljes szintaxisfát építenénk.

4.2.2. A szintaktikai elemzés pontosságának mérése

Ahhoz, hogy különféle módszerekkel végzett szintaktikai elemzések végeredményét össze tudjuk hasonlítani, szükség van egy olyan kiértékelő módszerre, ami releváns információt ad a szintaktikai elemzés pontosságáról. Ezt úgy oldhatjuk meg, hogy referenciaként kiválasztott mondatokat – melyeknek ismerjük a kézi annotációval kapott elemzését – leelemzünk az automatikus módszerünkkel is és a kétféle elemzés hasonlóságát valamilyen alkalmas mérőszámmal jellemezzük. Nyilván az ember által előállított elemzés is tartalmazhat hibákat, ezért ezt tekinthetjük az elérhető minőség felső határának, mivel az ember a szöveg értelmezésével, annak tartalma alapján végzi el az elemzést, míg az automatikus módszerek morfológiai és statisztikai alapon hozzák meg a döntéseiket.

Két szintaktikai elemzés összehasonlítása összetett feladat mivel első körben csak az tudjuk megállapítani, hogy az szintaxisfák teljesen egyeznek-e, viszont eltérés esetén is lehetnek egyező részek melyeket valahogy figyelembe kell vennünk. Minél jobban hasonlít az automatikusan előállított elemzési fa a referencia elemzéshez, annál precízebb eredményről beszélhetünk. Ezért az elemzési fákból olyan elemi tulajdonságokat kell kivonni, amelyek egyrészt mennyiségileg összehasonlíthatóak, másrészt jól jellemzik az eltérések mértékét. A probléma megoldását az elemzési fák szócsoportokra bontása adja, melyre példát a 4.5. ábrán láthatunk. Mivel ugyanannak a mondatnak a kétféle elemzéséről van szó, a kétféle szócsoport halmaz elemei ugyanazokra a szópozíciókra épülnek. Ez kihasználva a szócsoportok összehasonlítása halmazműveletekre egyszerűsödik, mivel az egyezések a két szócsoport-halmaz metszetében, az eltérések (hibák) pedig szimmetrikus differenciában lesznek.

Egy példamondat szintaxisa:

$[_{CP}\ [_{VP}\ [_{NP}\ [_{NP}$ 1:Mihály] 2:arca] $[_{ADVP}$ 3:azonnal] $[_{V0}$ 4:megváltozott]] 5:.]

A szintaxisban található szócsoportok:

NP(1-1) NP(1-2) ADVP(3-3) V0(4-4) VP(1-4) CP(1-5)

4.5. Ábra. Példamondat szintaxisának konvertálása szócsoportok halmazára.

Elemzési hiba kétféle módon adódhat: az automatikus elemzés felismerni vél olyan szócsoportot, ami az referencia elemzésben nincs benne, vagy a referencia elemzés tartalmaz olyan szócsoportot, amit az automatikus elemzés nem talált meg. A ***PARSEVAL*** metrikákkal [Black91] úgy is jellemezhetjük az automatikus elemzést, hogy nem annak hibáját, hanem a jóságát fejezzük ki, azaz, hogy a felismert

szócsoportok közül mennyi a helyes, ez a ***pontosság (precision)***, illetve, hogy a referencia elemzésben található (helyes) szócsoportok közül mennyit talált meg, ez pedig a ***fedés (recall)***. Ezeket a jellemzőket fejezik ki 0 .. 1 közötti értékkel (1 jelenti a teljes egyezést) a következő képletekkel:

$$Pontosság := \frac{Helyesen\ \ felismert\ szócsoportok\ száma}{Összes\ \ felismert\ szócsoportok\ száma} \tag{4.13}$$

$$Fedés := \frac{Helyesen\ \ felismert\ szócsoportok\ száma}{Referencia\ elemzésbenban\ található\ szócsoportok\ száma} \tag{4.14}$$

A pontosság és a fedés csak együtt jellemzi jól az automatikus elemzés jóságát, mert ha „óvatos” módszert választunk (csak a biztos tippet fogadjuk el) a pontosság nagyon jó lesz a fedés rovására, ugyanakkor ha „mohó” módszert alkalmazunk (minden lehetséges tippet elfogadunk, így ezek közt lesz a jó is) a fedés lesz nagyon jó, viszont a pontosság lecsökken. Ezért ezt a két jellemzőt egy ún. ***F-mérték*** foglalja össze, ami a pontosság és fedés súlyozott harmonikus közepe:

$$F_\beta := \frac{(1+\beta^2)\cdot Pontosság \cdot Fedés}{\beta^2 \cdot Pontosság + Fedés} \tag{4.15}$$

A β adja meg a súlyozás arányát, de általában nincs választási alapunk arra, hogy a pontosságot vagy a fedést előnyben részesítsük a másik rovására, így $F_{\beta=1}$ adja meg azt a mérőszámot mellyel az automatikus elemzés (és sok más felismerési feladat eredményének) pontosságát jellemezni szokás.

Egy másik *PARSEVAL* metrika amit szintén szintaktikai elemzések kiértékelésére használhatunk az elemzési és a referencia szintaxisfa közötti ***átfedések (crossing brackets)*** számán, illetve ezek mondatokra vetített arányán alapul:

$$n-crossing := \frac{Mondatok\ száma\ melyben\ \leq n\ átfedés\ van}{Összes\ referencia\ mondat\ száma} \tag{4.16}$$

A képlet alapján érdekes lehet a 0-crossing, 1-crossing, stb., valamint szokás az elemzett mondatok maximális hossza szerint több csoportra osztva elvégezni a kiértékelést, mivel egy hosszú mondat esetén lényegesen nagyobb annak az esélye, hogy átfedést találjunk.

Ha ugyanolyan kiértékelési módszert használunk ugyanolyan feltételek mellett a különböző szintaktikai elemzők eredményeit össze lehet mérni. Erre példa a szintaktikailag annotált angol szövegeket tartalmazó Penn Treebank [Marcus93], melynek 23. szekciója referenciaként van kijelölve, a 2-21-es szekcióját pedig az elemző modelljének előállítására lehet használni. Mivel teljesen elfogadott és elterjedt

ennek a kiértékelési módnak a használata, az új eredmények számára ezzel egy azonnal hozzáférhető globális összehasonlítási lehetőség áll rendelkezésre.

Amennyiben nem állnak rendelkezésre referencia adatok, akkor is szükség lehet pontos kiértékelésre. Például ha ugyanannak a módszernek vizsgáljuk a különböző változatait, a tesztadatok informálhatnak minket arról, hogy milyen irányban érdemes a modellt továbbfejleszteni. Ugyanakkor ezeket a (finomító) tesztadatokat is a tanítás részének kell tekinteni és lehetőleg mindig más felosztást kell alkalmazni, mert különben a rendszerünk „rátanul" a teszt adatokra és a kiértékelés félrevezető eredményt ad. Másrészt modell kiértékelésénél kapott pontosság értéke függhet attól is, hogy a példák milyen módon voltak felosztva tréning és teszt halmazokra. Ezekre a problémákra egy általánosan elterjedt módszer a ***tenfold crossvalidation***, melyben az adatokat véletlenszerűen felosztjuk 10 egyenlő részre, majd mindig más részen tesztelve a többi részen pedig tanítva átlagoljuk az eredményeket.

4.3. A modell optimalizálása

Amikor kézzel „kipróbálunk" egy-két változtatást és a kiértékelés eredményétől függően módosítjuk a modell létrehozásának a feltételeit, akkor tulajdonképpen egy optimalizálást hajtunk végre az eredmények visszacsatolásával. Ennél kényelmesebb megoldás, ha módszereinket átszervezve automatizáljuk ezt a folyamatot úgy, hogy egy alkalmas optimalizáló algoritmusból, a modellgeneráló módszerünk és a modell-kiértékelés fölé egy kereteljárást készítünk.

4.3.1. Optimalizáló keret-algoritmus

A szerző és társai által készített LL rendszerben a szerző kifejlesztett egy ilyen keret-optimalizáló módszert [Hócza02]. Az LL rendszerben egy adott feladatra különféle gépi tanulási algoritmusokat lehetett futtatni, kiértékelni. A rendszer egy külön szolgáltatása volt, hogy a különféle tanulási feladatok végrehajtását szimulált hűtéssel automatikusan optimalizálta, azaz megkereste a kiválasztott tanuló algoritmusnak azt a paraméter beállítását, ami a kiértékelés alapján a legnagyobb pontosságot éri el a teszt példákon.

Az optimalizálhatóság érdekében a modellt előállító módszernek paraméterezhetőnek kell lennie és néhány érték megváltoztatásával modellváltozatok széles spektrumát kell tudnia előállítani. Az optimalizáló algoritmusnak képesnek kell lennie lépésenként működve meghatározni az új paramétereket és azt, hogy a pontosság (vagy lépésszám) elért-e már egy adott küszöböt. Mivel sok adatra és hosszú futásra számíthatunk az

adatok felosztása ***n-fold crossvalidation*** módszerrel történik, ahol ***n*** értéke 10 helyett akár 2 vagy 1 is lehet, illetve a futtatást egy szűkebb korpuszon végezzük és amikor megvan az optimális paraméter beállítás, akkor fut le a végleges modell elkészítése a teljes tanításra szánt korpuszon. Az optimalizálás keret-algoritmusa a 4.5. ábrán látható.

```
PARAMÉTER-OPTIMALIZÁLÁS(Korpusz, N) returns paraméterek
   Paraméterek, Pontosság = OPTIMALIZÁLÓ.INICIALIZÁLÁS()
   while not OPTIMALIZÁLÓ.LEÁLLÁS(Pontosság) do
      Paraméterek = OPTIMALIZÁLÓ.ÚJ-MEGOLDÁS(Paraméterek)
      Pontosság = 0
      for I = 1 to N do
         Tréning, Teszt = N-FOLD(Korpusz, I)
         Modell = MODELL-GENERÁLÁS(Tréning, Paraméterek)
         Pontosság += KIÉRTÉKELÉS{Teszt, Modell}
      Pontosság /= N
   return Paraméterek
```

4.5. Ábra. A modell-paraméterek optimalizálásának keret-algoritmusa

4.3.2. A szimulált hűtés almodulokra bontása

A optimalizáló modul megvalósítására megfelelő a 3.3.3. fejezetben bemutatott szimulált hűtés algoritmusa, melyet külön hívható almodulokra kell bontani, hogy megfeleljen keret-algoritmus céljainak. Ezek az almodulok megadják a kezdeti értékeket, megállapítják leállási feltételt a pontosság alapján és kiszámítják az új paramétereket az előző állapot alapján. Mindezt úgy, hogy a működéshez szükséges belső információkat (például hőmérséklet) karbantartják az iterációs lépések során. Az almodulokra bontott külön hívható metódusok megvalósítása a 4.6 ábrán látható.

```
OPTIMALIZÁLÓ.INICIALIZÁLÁS()
   T, M, M_best = INICIALIZÁLÁS()

OPTIMALIZÁLÓ.LEÁLLÁS ()
   return LEÁLLÁS(T, M_best)

OPTIMALIZÁLÓ.ÚJ-MEGOLDÁS()
   M = ÚJ-MEGOLDÁS(M)
   ΔE = E(M_best) - E(M)
   if (ΔE < 0) or
      (ΔE ≥ 0 and e^(-ΔE/λT) > random[0,1]) then M_best = M
   T = L(T)
```

4.6. Ábra. A szimulált hűtés almodulokra bontása

Az almodulokat megvalósító metódusokban szereplő függvényhívások (*INICIALIZÁLÁS, LEÁLLÁS, ÚJ-MEGOLDÁS*) az eredeti algoritmus privát függvényei.

4.4. Konklúzió

Ebben a fejezetben áttekintettük a szerző által kidolgozott komplex szintaktikai elemző módszert. Az a modell építése a kiinduló faminta halmaz korpuszból való kigyűjtésével indul, mely faalak típusok felhasználásával történt. A kiinduló faminta halmazból az RGLearn algoritmus segítségével tanulunk famintákat. Az így kapott faminta halmazt a chart parser módosított változata használja a szintaktikai elemzés során. A kiértékelési módszer bevezetése lehetőséget biztosít a szerzett tapasztalatok visszacsatolására. A szimulált hűtés algoritmusának egy módosított változatával úgy optimalizáljuk a modellkészítés paramétereit, hogy a felismerési pontosság maximális legyen.

5. Szintaktikus elemzési módszerek alkalmazásai magyar nyelvre

Ebben a fejezetben a szerző beszámol a felszíni és teljes szintaktikai elemzéssel elért eredményeiről valamint ezek alkalmazásairól információkinyerő illetve magyar-angol gépi fordító rendszerekben. A megvalósítás lehetőségei erősen kötődtek a Szeged Treebank [Csendes05] különböző munkafázisaihoz, melyek annotálási munkáinak befejezésével létrejött egy olyan adatbázis, amely magyar nyelvre is lehetőséget biztosít gépi tanulási technikákon alapuló szintaktikus elemzési módszerek kidolgozására.

A treebank létrehozásának folyamata több fázisra bontható. Az első fázisban a szövegek kiválasztása, mondat és szó szegmentálás, a szavak lehetséges morfológiai kódjainak hozzárendelése és kontextus szerinti egyértelműsítése történt meg. Az ekkor még Szeged Korpusznak [Alexin03] nevezett adattárat gépi tanulási technikákon alapuló ***szófaji egyértelműsítő (POS-tagger)*** készítésre lehetett használni. A szerző és társai több különféle módszeren alapuló szófaji egyértelműsítőt is készített, melyek összehasonlításáról bővebben a [Kuba03] és a [Kuba04] cikkekben olvashatunk.

A második fázisban a Szeged Korpusz annotált szövegeit továbbfejlesztve létrejött a Szeged Treebank 1.0, amely már a főnévi csoportok (NP) bejelölését is tartalmazta. Ez az állapot lehetőséget biztosított a ***felszíni elemzés (Shallow Parsing)*** automatikus, gépi tanulási technikákon alapuló végrehajtására, melyekről a szerző a [Hócza03a] és [Hócza04a] cikkekben számol be. Az elért eredmények egyesítésével a szerző és társai elkészítettek egy olyan ***információkinyerő (Information Extraction)*** rendszert [Hócza03b], [Hócza04b], mellyel üzleti híreket tartalmazó szövegekből lehet különféle tranzakciók adatait egy strukturált adatbázisba kigyűjteni.

A harmadik fázisban folytatódtak az annotálási munkálatok további szócsoportok (melléknévi, határozói, névutós csoportok, igei vonzatok, tagmondatok, stb.) bejelölésével és létrejött a Szeged Treebank 2.0, a végleges állapot. A bővebb mondattani információkat felhasználva elkészül a 4. fejezetben részletezett módszer szerinti teljes szintaktikai elemző, melynek megvalósításáról és kiértékelésének eredményeiről a szerző a [Hócza04c] és [Hócza06a] cikkekben számol be. A [Hócza05a] cikkben a szerző és társai a szintaktikai faminták tanulását a Boosting algoritmussal javítják fel és az eredményeket összehasonlítják más kombinációs sémákkal. Más módszerekkel történő összehasonlításról a [Hócza05c] és a [Barta05] cikkekben olvashatunk. A magyar nyelvre készített teljes szintaktikai elemző módszer megvalósítása lehetőséget teremt arra, hogy azt felhasználjuk olyan feladatokban – mint például a ***gépi fordítás (machine translation, MT)*** –, melyek a mondattan részletesebb

feltárását igénylik. Erre példa egy létező fordító rendszer a GenPar kibővítése magyar-angol modullal, mely a szerző szerepe meghatározó volt [Hócza06b].

5.1. A szintaktikai elemzés problémakörének áttekintése

Ebben a fejezetben áttekintjük a magyar nyelv szintaktikai elemzésének területét. Először részletezzük, hogy a magyar nyelv milyen speciális tulajdonságokkal rendelkezik ami megnehezítheti a szintaktikai elemzés más nyelvekhez (pl. angol) képest. Ezután összehasonlítjuk a magyar és az angol nyelv szintaktikai elemzésének technikai lehetőségeit és felsoroljunk, hogy a szerző munkássága előtt milyen szintaktikai elemzési eredmények születtek magyar nyelvre.

5.1.1. A magyar nyelv szintaktikai elemzésének nehézségei

A magyar nyelv számos olyan nyelvi sajátossággal rendelkezik, ami megnehezíti a szintaxisfelismerést az indoeurópai nyelvekhez (pl. angolhoz) képest. Az egyik jelentős különbség a viszonylag szabad szórend, azaz egy mondat szintaktikai egységei többféleképpen átrendezhetők úgy, hogy a kapott mondatok nyelvtanilag szintén szabályosak lesznek. Azonban az így kapott mondatok jelentései módosulhatnak a kiinduló mondatéhoz képest. A mondatrészi szerepet a magyar nyelv ragozással és névutók alkalmazásával oldja meg. Ebből adódik a másik probléma, a nagyfokú morfológiai változatosság. Az említett sajátosságok összességében jelentősen megnövelik a lehetséges minták, nyelvi sémák számát, melyek rontják a statisztikai alapú gépi tanulás hatékonyságát.

A szintaktikai elemzés leggyakrabban előforduló és egyik legfontosabb egysége a ***főnévi csoport (NP)***, mely általában névelővel kezdődik és főnévvel végződik, ez utóbbit az NP fejének is nevezünk. Ha nem lennének ez alól kivételek az NP-k felismerése nagyon pontos lehetne. Azonban névelő bizonyos esetekben elhagyható, bizonyos esetekben nem:

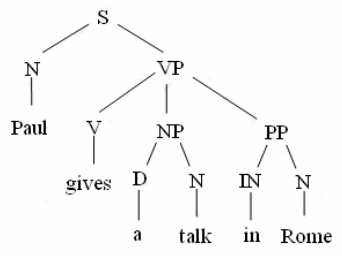
$[_{NP}$ *Péter* $]$ $[_{NP}$ ~~*(egy)*~~ *könyvet* $]$ *olvas* .

$[_{NP}$ *Péter* $]$ *olvassa* $[_{NP}$ *a könyvet* $]$.

Ha a kontextus ezt lehetővé teszi, az NP feje is hagyható, tehát ez alapján előfordulhat olyan NP is melynek az utolsó szava nem főnév:

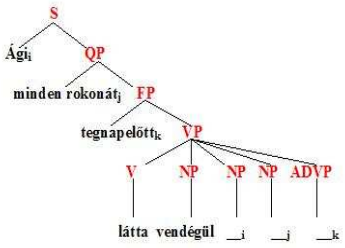
$[_{NP}$ Péter $]$ $[_{NP}$ a régi könyvet $]$ olvassa , $[_{NP}$ Mari $]$ pedig **[**$_{NP}$ az **újat]** .

A szabad szórend alkalmazása a magyar nyelvben (példa az 5.1. ábrán) nem jelenti azt, hogy bármilyen szót bárhova áthelyezhetünk a mondatban, mert ha ezt tennénk nagyon hamar értelmezhetetlen vagy magyartalan mondatokat kapnánk. A szabad átrendezhetőség az igei vonzatkeret elemeire vonatkozik. Ezek az elemek NP-k vagy más szócsoportok lehetnek, melyek szavai együtt mozoghatnak, a belső sorrendjük viszont kötött. Az igei vonzatkeret elemeinek átrendezése is csak tagmondathatárokon belül lehetséges, ha meg akarjuk tartani a mondat eredeti értelmét. Mindez viszont azt jelenti, hogy az angol nyelvre nagyszerűen alkalmazható generatív felfogásban használt VP – mely arra épül, hogy a szintaktikai szerepeket a kötött szórend osztja ki – magyar nyelvre ebben a formában használhatatlan, mert a VP $\rightarrow \beta$ szabály, melynek jobboldalát mindenféle sorrendben fel kellene venni a nyelvtanunkba, ráadásul a magyar igék vonzatkerete nem feltétlenül összefüggő részt fed le a mondatból, azaz a szabály jobboldalát (β) szócsoportok nem összefüggő sorozatára kellene illeszteni. Ez a magyar nyelvi specialitás azért is probléma, mert a szintaktikai elemzés legfontosabb eleme a VP (illetve az igei vonzatkerettel végzett egyeztetés), mivel ez alapján zárhatjuk ki azokat a hibás részelemzéseket, melyeket az alacsonyabb szinteken (bottom-up elemzéssel) a rendelkezésre álló információk alapján létre tudtunk hozni.



Paul gives a talk in Rome.

(nincs más lehetséges szórend)



Ági minden rokonát vendégül látta tegnapelőtt.

Tegnapelőtt vendégül látta minden rokonát Ági.

Minden rokonát vendégül látta Ági tegnapelőtt.

stb.

Jelmagyarázat:

- Mondatszerkezet: S (mondat), QP (kvantorpozíció), FP (fókuszpozíció)
- Szócsoportok: VP (igei csoport), NP (főnévi csoport), ADVP (határozószói csoport), PP (névutói csoport)
- Szófajok: V (ige), N (főnév), D (névelő), IN (prepozíció)

5.1. Ábra: A magyarban a viszonylag szabad szórend miatt egy adott VP sokféle elrendezésben előfordulhat, sőt nem is mindig alkot összefüggő szerkezetet.

A felsorolt problémák kezelésére választott tág elmélet, a magyar generatív szintaxis ([Kiss99], [Alberti02], [Kiefer92]) szerint a mondatelemzés első lépésként az újraíró szabályok és a lexikon minden esetben létrehozzák a mondat kiinduló vagy ún. mélyszerkezetét. Ebből a végső vagy ún. felszíni szerkezet transzformációkkal (mozgatások, törlések) hozható létre. A magyar mondat mélyszerkezetében az ige áll elöl, mögötte pedig tetszőleges sorrendben a bővítményei sorakoznak. A levél nélküli ágak a mélyszerkezetben kitöltetlenek, ún. funkcionális pozíciók, ezekre a helyekre lehet majd mozgatni az ige mögül az összetevőket. A funkcionális pozíciók össze vannak indexelve az elmozgatott összetevőkkel, így a mozgatást nem kell nyilakkal ábrázolni. A szintaktikai fa (sem a mély-, sem a felszíni struktúra) nem ad számot arról, hogy a jelenlévő igebővítmények közül melyek kötelezőek. Ezt az információt a lexikon tartalmazza.

A magyar mondatok struktúráját leíró konzisztens szintaktikai szabályrendszer kidolgozásánál figyelembe kell venni olyan annotálási szempontokat is, melyek leginkább alkalmazkodnak a későbbi számítógépes feldolgozás elveihez. Ezek alapján eltekinthetünk a funkcionális pozíciók ábrázolásától, mivel ezek betöltöttsége egyes esetekben egyértelműen kiszámolható az ige helyzetéből, a többi esetben viszont a hangzó beszéd prozódiai jellemzőinek függvénye, ilyen jellemzők viszont nincsenek kódolva az írott szövegekben.

5.1.2. Kapcsolódó eredmények

Angol nyelvre számos eredmény létezik a mondatszintaxis felismerésének témakörében. A Penn Treebank [Marcus93] annotált szövegeiben elkülönítettek egy részt, amely a megjelenése óta összehasonlítási alapot képez a témához kapcsolódó publikációk eredményeihez. A chunking módszer [Abney91] nyelvtani kódok alapján osztályozta a szavakat, hogy azok kezdő, vég- vagy belső elemei-e egy adott típusú frázisnak. További eredmények a transzformáción alapuló tanulás [Ramshaw95], a főnévi és igei szerkezetek egyszerre történő felismerése [Argamon98] és a több fokozatú kaszkád alkalmazása [Tjong99]. A legújabb módszerek úgy érnek el javulást az eredményekben, hogy több módszert összekombinálva, szavazással hozzák meg a döntéseket [Tjong00]. Az eredmények összefoglalása az 5.1. táblázatban látható.

Hivatkozás	Módszer	Kiértékelés	$F_{\beta=1}$
[Abney91]	Chunking	–	–
[Ramshaw95]	Transzformációs tanulás	Penn Treebank	92.00%
[Argamon98]	NP és VP szerkezetek	Penn Treebank	91.60%
[Tjong99]	Többszintű nyelvtan	Penn Treebank	92.37%
[Tjong00]	Módszerek kombinációja	Penn Treebank	93.26%

5.1. Táblázat: Néhány fontosabb angol nyelvre elért eredmény, 1993 óta az összehasonlíthatóság érdekében a Penn Treebank adatain történik a tesztelés.

Magyar nyelvre a Szeged Treebank verzióinak a megjelenése előtt lényegesen kevesebb szintaktikai elemző készült. Megfelelő minőségű és mennyiségű annotált magyar szövegek nélkül csak kézzel készített szakértői szabályrendszert tartalmazó szintaktikai elemző alkalmazására van lehetőség. Egy ilyen rendszer a MorphoLogic Kft. által kifejlesztett HumorESK szintaktikai elemző (Kis, 2003), mely 1995 óta folyamatosan fejlődik. Ez idő alatt különféle nyelvészeti területeken alkalmazták. Fő jellemzője, hogy a szimbólumokhoz jegyszerkezeteket kapcsol, és elemzési erdőt épít az egyes jegyek öröklтetésével. A Nyelvtudományi Intézet is beszámolt egy szintaktikai elemzőről [Váradi03], ami főnévi szerkezeteket ismer fel reguláris kifejezésekkel leírt, többfokozatú (kaszkád) szakértői szabályrendszer alkalmazásával, a CLaRK rendszer [Simov01] segítségével. A szerző által közölt eredmények ([Hócza04a], [Hócza05a]), melyek a Szeged Treebank adatait felhasználva gépi tanulási technikák alkalmazásával készültek, jelentősen túlléptek az előzőekben a magyar nyelvre elért eredményeken. A felsorolt magyar nyelvre elért eredmények összefoglalása az 5.2. táblázatban látható.

Hivatkozás	Módszer	Kiértékelés	$F_{\beta=1}$
[Kis03]	Szakértői szabályok	–	–
[Váradi03]	Szakértői szabályok	100 annotált mondat	58.78%
[Hócza04a]	Főnévi szerkezetek	Szeged Treebank 1.0	83.11%
[Hócza05a]	Teljes szintaxis	Szeged Treebank 2.0	78.59%

5.2. Táblázat: A magyar nyelvre elért eredmények a Szeged Treebank bevezetéséig.

5.2. A Szeged Treebank kialakításának folyamata és tartalma

A mondatok szintaktikai szerkezetét leíró, ún. treebank reprezentáció a legtöbb nyugat-európai nyelvre, de számos közép-, ill. kelet-európai nyelvre már létezik, ezért időszerűnek bizonyult egy pontosan elemzett magyar nyelvű treebank létrehozása is. A Szeged Treebank [Csendes05] kialakításakor a már ismert forrásmunkákra és meglévő elméletekre támaszkodva nyelvész szakértők egy konzisztens szintaktikai szabályrendszert dolgoztak ki. A treebank kidolgozása több fázisban történt az IKTA 27/2000, az NKFP 2/017/2001 és az IKTA 37/2002 kutatás és fejlesztési projektek keretében. A munkafázisok részleteiről a következő alfejezetekben olvashatunk.

5.2.1. Szeged Korpusz

Az IKTA 27/2000 kutatás és fejlesztési projekt keretében létrejött Szeged Korpusz [Alexin03] szövegállományának összeállításakor azt volt a fő szempont, hogy az elektronikus formában rendelkezésre álló szövegekből egy sokszínű válogatás készüljön, melyben a magyar nyelv egyes megnyilvánulási módjaira elegendő számú példamondat áll rendelkezésre. A szövegek végül hat különböző témakörből kerültek ki, témakörönként ~200 ezer szó terjedelemben. A hat témakör – melyek mindegyikéről elmondható valamilyen speciális jellemző – következő:

- *Szépirodalom*: általános szókincs, helyesírási és stilisztikai szempontok magas szintű betartása, párbeszédek.
- *14-16 éves korú tanulók fogalmazásai*: köznyelvi szókincs és mondathasználat, helyesírási hibák.
- *Újságcikkek* (Népszabadság, Népszava, Magyar Hírlap, HVG): hivatalos közlési nyelvezet.
- *Számítástechnikai szövegek:* speciális szókészlet, speciális tokenek (könyvtár, webcím, stb.).
- *Jogi szövegek*: speciális karakterek, szókészlet és nyelvezet, esetenként hosszú (több száz szavas) mondatok.
- *Gazdasági és pénzügyi rövidhírek*: Gyakran ismétlődő szófordulatokra épülő tényközlés, struktúrába rendezhető információk.

A korpusz eredetileg az első 5 témakört (~1 millió szövegszó) tartalmazta, ez az állapot tekinthető a Szeged Korpusz 1.0-ás verziónak. A ~200 ezer szót tartalmazó kiegészítés, a „*Gazdasági és pénzügyi rövidhírek*” (rövidebben: „*Üzleti hírek*”) később

lett felvéve a korpuszba – az információkinyerő rendszer fejlesztéséhez kapcsolódóan – és átesett ugyanazon az annotálási munkafolyamaton, mint az 1.0-ás szövegek.

A teljes korpusz együttesen ~82.000 mondatot (~1,2 millió szövegszót + ~250 ezer írásjelet) tartalmaznak. A korpusz fájljai 100 mondatos egységekbe tagolva a későbbi annotálási munkálatok számítógépes megvalósítását támogató XML-formátumban kerültek tárolásra, belső szerkezetüket a TEI P4 DTD (Document Type Definition) séma írja le. A szegmentált mondatok <s> és </s> XML-címkék között találhatók, melyek között először a mondat szövege, majd a mondatot alkotó szavak és írásjelek felsorolása található <w> és </w>, ill. <c> és </c> címkék között. A szavak leírásánál először maga a szó aktuális alakja áll, majd a szó lehetséges morfo-szintaktikai (MSD) kódjai következnek a hozzájuk tartozó szótövekkel együtt. A szavak leírása tartalmazza a szófaji egyértelműsítést is, azaz a lehetséges (szótő, MSD-kód) párok közül pontosan egy ki van választva a szövegkörnyezet alapján.

Az MSD kódrendszer [Erjavec97] a szavak legkülönbözőbb attribútumainak kódolására szolgál, mely az európai nyelvek többségére alkalmazható. A szavak morfo-szintaktikai attribútumait (típus, nem, szám, személy, eset) egy karaktersorozat reprezentálja. Az első pozíción a szó fő kategóriája áll. Például a magyar *asztalnak* szó az `Nc-sg------` kódot kapja, amely a következőket jelenti: főnév, köznév, egyes szám, birtokos eset (*genitivus*). A hiányzó vagy az adott természetes nyelvben nem értelmezhető attribútumok helyén – jel áll. A magyar nyelvben például a főneveknek nincs *neme*, ami más nyelvekben igen elterjedt. A Szeged Korpusz szövegeiben a magyar nyelvre alkalmazott MSD kódrendszer körülbelül 1800 féle kódot használ a szavak morfo-szintaktikai jellemzésére.

5.2.2. Szeged Treebank 1.0

A természetes nyelvi feldolgozás fontos lépése a szintaktikai elemzés és annotálás Mivel a mondatok többségében, az egész mondat jelentése szempontjából a főnévi csoportok kulcsfontosságú szerepet játszanak, ezért az IKTA 37/2002 kutatás és fejlesztési projekt keretében előállított Szeged Treebank 1.0 verziójában ezeknek a szerkezeteknek a bejelölése volt az elsődleges cél. A főnévi csoportok, mint az igék legjellemzőbb bővítményei, hordozzák az információk nagy részét, ezért például információkinyerő alkalmazásokban kiemelkedő szerepük van.

Ezen kívül, ugyancsak a mondatok tartalmának értelmezhetősége szempontjából, fontos szerepe van a tagmondatok elkülönülésének és egymáshoz való viszonyának. További szintaktikai elemzésnél is fontos támpontot jelentenek a tagmondat-bejelölések, hiszen a tagmondathatáron szintaktikai egységek nem nyúlhatnak át. Éppen

ezért a treebank jelen verziójában a tagmondatok határai és a viszonyukra vonatkozó információ (alárendelés, mellérendelés) egyaránt szerepel.

A Szeged Treebank 1.0 a hat témakört tartalmazó mofológiailag egyértelműsített Szeged Korpusz szövegeinek feldolgozásával jött létre. A szövegek szintaktikai címkékkel történő ellátásához a nemzetközileg elfogadott XML-helyes NP (***noun phrase*** = főnévi csoport) és CP (***clausal phrase*** = mondat értékű frázis) jelölések voltak használva. Az alá- illetve mellérendelt tagmondatok szintén CP címkét kaptak. Az annotálás folyamán XP címkét is alkalmaztak, melyek a szövegbe szervesen nem illeszkedő részek (közbevetés gondolatjelek között vagy zárójelben, rövidítés feloldása zárójelben stb.) elkülönítésére szolgáltak.

Főnévi csoportok tulajdonságai

A szófajilag egyértelműsített szövegben egy főnévi csoport egy vagy több szóból és esetleg írásjelből álló összefüggő sorozat. Nyelvészeti szempontból az a kifejezés tekintendő főnévi csoportnak, amely főnévi fejű, nem választható szét nem összefüggő részekre, és maximális nagyságú. Minden főnév feje egy főnévi csoportnak, de van olyan főnévi csoport, amelynek nincs látható főnévi feje. Ilyenkor feltételezzük, hogy a főnévi fej ott volt ugyan, csak törölve lett, például:

Vettem [$_{\text{NP}}$ *két almát*] . [$_{\text{NP}}$ ***A pirosabbat***] *megettem*, azaz [$_{\text{NP}}$ *a pirosabb almát*]

Ha egy kifejezés főnévi fejű, de a kifejezés szétválasztható két vagy több olyan részre, amelyek a mondatban áthelyezhetők a mondatban egymással nem szomszédos helyekre a mondat jelentésének a megváltoztatása nélkül, akkor a kifejezés nem tekintendő egy főnévi csoportnak, például ha egy birtokos szerkezetben a birtokos genitívuszi esetben áll, akkor az (legtöbbször) elmozdítható a birtok mellől, ezért nem alkotnak egy főnévi csoportot akkor sem, ha egymás mellett állnak:

Láttam [$_{\text{NP}}$ *Péternek*] [$_{\text{NP}}$ *a húgát*] . $\rightarrow$ [$_{\text{NP}}$ *Péternek*] *láttam* [$_{\text{NP}}$ *a húgát*] .

Ugyanez nem igaz az alanyesetű birtokosra:

Láttam [$_{\text{NP}}$ *Péter húgát*] .

A maximális hosszúságú a főnévi csoport szabályát kell alkalmazni a következő példamondatban: a *Péter látta a népdalokat nagy lelkesedéssel éneklő lányt.* A főnévi csoport *a népdalokat nagy lelkesedéssel éneklő lányt* lesz, nem pedig a *nagy lelkesedéssel éneklő lányt*, mivel az első hosszabb a másodiknál.

Főnévi csoportot alkotnak még a fentieken kívül azok a kifejezések, amelyek két vagy több főnévi csoport egymás mellé rendelésével kapunk. Az ilyen mellérendeléseknek az alábbi szerkezetük:

[$_{\text{NP}}$ [$_{\text{NP}}$ *Péter*] *és* [$_{\text{NP}}$ *az öcsém*]], nem pedig: [$_{\text{NP}}$ *Péter és az öcsém*]

Tagmondatok tulajdonságai

A mondatok lehetnek egyszerűek vagy összetettek. Az összetett mondatok tagmondatai alárendelt vagy mellérendelt tagmondatok lehetnek. A tagmondatok bejelölésénél egyrészt meg kell határozni magukat a tagmondatokat, másrészt el kell helyezni azokat az egész mondatszerkezetben.

A mellérendelt tagmondatok együtt (egyenrangúan) alkotnak egy bővebb tagmondatot:

[$_{CP}$ [$_{CP}$ *Péter megérkezett*] *és* [$_{CP}$ *elénekelt egy dalt*] .]

Az alárendelt tagmondatok teljes egészében benne találhatóak egy másik (tag)mondatban:

[$_{CP}$ *Péter azt akarja, hogy* [$_{CP}$ *elénekeljek egy dalt*] .]

A tagmondatokat legtöbbször kötőszó vezeti be, és egy következő kötőszóig, vagy írásjelig tartanak. A kötőszavak általában nem részei a tagmondatnak: a *hogy, és, bár, vagy,* stb. kötőszavak a tagmondaton kívül találhatóak. Kivétel lehet a *pedig, viszont, azonban, meg, ugyanis* stb. kötőszók a tagmondat első összetevője (általában egy főnévi csoport) után is állhatnak:

[$_{CP}$ [$_{CP}$ *Péter iskolába ment*] , [$_{CP}$ *Mari pedig orvoshoz*] .]

Ha két tagmondat között **kettőspont** áll, akkor az úgy viselkedik, mintha kötőszó lenne. A kettőspont lehet alárendelő kötőszó szerepű, ekkor *hogy*-gyal lehet helyettesíteni, vagy pedig mellérendelő: *azaz* szerepű:

[$_{CP}$ *Péter elmondta:* [$_{CP}$ *sikerült a terv*] .]

[$_{CP}$ [$_{CP}$ *Péter szórakozni ment*]*:* [$_{CP}$ *egy új magyar filmet nézett meg a moziban*] .]

A tagmondatokban legtöbbször található egy ragozott ige, de tagmondatonként legfeljebb egy. Egy tagmondatban csak akkor lehet egynél több ragozott ige, ha azok egymás mellett szerepelnek mellérendelő kötőszóval elválasztva, valamint jelentésszerkezetük hasonló (ugyanannyi és ugyanolyan tematikus szerepet osztanak ki): *Péter ismeri és élvezi a vietnami filmeket.* Egy tagmondat két esetben lehet ragozott ige nélkül. Ha a mondat jelen idejű kijelentő módú, az alanya egyes szám harmadik személyű, és predikatív főnevet vagy melléknevet tartalmaz: *Péter orvos (?van).* Ha a tagmondat elliptikus, akkor is hiányozhat belőle a ragozott ige, de ekkor egy másik (általában neki mellérendelt) tagmondat igéjét értjük oda:

[$_{CP}$ [$_{CP}$ *Péter iskolába ment*] , [$_{CP}$ *Mari pedig orvoshoz (~~ment~~)*] .]

A treebank előállításának lépései

A Szeged Korpusz szófajilag egyértelműsített szövegei egy automatikus szintaktikai előelemzési folyamaton mentek keresztül, ahol az előelemző program bejelölte a tagmondatokat, illetve a később nagy valószínűséggel főnévi csoportnak bizonyuló kifejezéseket. Az automatikus előelemzés a Bolgár Tudományos Akadémia nyelvészei által kifejlesztett CLaRK rendszerrel [Simov01] történt, amely lehetővé teszi szintaktikai egységek automatikus felismerését reguláris szabályok segítségével.

Az előelemzés során az elsődleges cél az volt, hogy az a lehető legjobban segítse az annotátorok munkáját, tehát az előelemzéssel helyesen jósolt tagmondat és főnévi csoport, több legyen mint a hibásan megjelölt. Emiatt az előzetes annotálás csak az egyértelmű eseteket kereste meg, azokat melyeknek a találati aránya kisebb volt 50 százaléknál, kihagyta. Az előelemzéshez használt CLaRK rendszer reguláris kifejezések illesztésével jelölte be a vélt tagmondatokat és főnévi csoportokat. Az előzetes elemzés során a reguláris szabályok csoportokra, szabályszintekre voltak osztva és kaszkád-szerűen egymásra épülve történt az alkalmazásuk.

A folyamat következő lépése az automatikusan előállított annotáció felülbírálása és javítása volt. CP-k esetében az előelemzés a tagmondatokat nem tudta meghatározni, mivel ezen a ponton ehhez nem állt rendelkezésre elegendő információ, ezért ezek bejelölése teljes mértékben kézzel történt. Az NP szerkezetek esetén felül kellett vizsgálni az összetettebb eseteket, melyekre az előelemzés bizonytalan eredményt adott volna és ezért inkább kihagyott. A annotálást végzők munkája szigorú minőség-ellenőrzésen estek át, melynek alapját a 4.2.2. fejezetben leírt kiértékelési technika képezte. Az elfogadási procedúra az volt, hogy néhány véletlenszerűen kiválasztott mondatot az ellenőrzést végző személy is annotált és az így kapott kétféle elemzésnek legalább 95%-os $F_{\beta=1}$ értékkel egyeznie kellett egymással.

5.2.3. Szeged Treebank 2.0

A az IKTA 37/2002 kutatás és fejlesztési projekt következő munkafázisa a teljes szintaktikai elemzést tartalmazó Szeged Treebank 2.0 kialakítása volt. A kiinduló állapot a Szeged Treebank 1.0 volt, az ebben szereplő NP és CP annotációt már tartalmazó mondatok kaptak újabb szintaktikai címkéket.

Ebben a munkafázisban is az első lépés a kézi annotáció segítése érdekében végzett automatikus előannotálás volt. Mivel a szavak szintaktikai szerepét az esetek többségében a szó MSD kódja, vagyis morfo-szintaktikai jegyei alapján meg lehetett állapítani, ebben az esetben már nem volt szükség szakértők által definiált reguláris

szabályok használatára. A szintaktikai egységek automatikus bejelölésére egy egyszerű szabályokon alapuló programmal történt. Ennek kimenetétől természetesen nem volt elvárható, hogy 100%-os pontossággal határozza meg a szerkezeteket, tehát a szakértői ellenőrzés és javítás ebben a fázisban sem maradhatott el. A folyamat következő lépése az automatikusan előállított szintaktikai annotáció felülbírálása és javítása volt, melynek minőségellenőrzése az 1.0 verziónál alkalmazott módszer szerint történt.

Tekintsük át, hogy milyen annotált szócsoportok találhatók meg a korpuszban és ezeknek mik a főbb jellemzőik:

A ***tagmondatok (CP)*** és a ***főnévi csoportok (NP)*** jellemzőit az 1.0 verziónál áttekintettük.

Melléknévi csoportok (ADJP): minőségjelzői szerepű szavak ADJP-t alkotnak. Az ilyen szavak jellemzően melléknevek vagy melléknévi igenévek lehetnek. Az ADJP-nek magába kell foglalnia a fej bővítményeit is. Ez melléknevek esetén általában határozószó (ADVP), melléknévi igeneveknél pedig határozószó vagy NP. Mellérendelés is előfordulhat, ha a két ADJP egyenrangú ezek újabb ADJP-t alkotnak. Például:

a [$_{\text{ADJP}}$ [$_{\text{ADJP}}$ [$_{\text{ADJP}}$ *nagyon*] *szép*] *és* [$_{\text{ADJP}}$ *okos*]]] *gyerek*

Határozószói csoportok (ADVP): Ide tartoznak a határozószók (pl. *most*, *itt*, *nagyon*, *mindig*), a „határozóragos” melléknevek (pl. *betegen*, *angolul*, *jogilag*), a számnevek (pl. *öten*, *sokan*) és a „névutós” személyes névmások (pl. *mellettem*, *miattad*, *utánunk*). Több egymás után következő határozószót egymásba ágyazott szerkezettel ábrázolunk:

[$_{\text{ADVP}}$ [$_{\text{ADVP}}$ *nagyon*] *szépen*]

Névutós csoportok (PP): Névutós csoportot akkor kapunk, ha főnévi csoporthoz névutót kapcsolunk. A névutó általában az NP után áll, de ritkán meg is előzheti:

[$_{\text{PP}}$ [$_{\text{NP}}$ *Pistával*] *szemben*]

[$_{\text{PP}}$ *szemben* [$_{\text{NP}}$ *Pistával*]]

Kötőszavak (C0): A kötőszavak megkeresésénél is az MSD kódokra támaszkodunk. Ezek kódjának első karaktere **C**. A tagmondatokat legtöbbször kötőszó vezeti be, de nem része a tagmondatnak.

[$_{\text{CP}}$ Azért jött , [$_{\text{C0}}$ hogy] [$_{\text{CP}}$ magával vigyen] .]

Elváló igekötők (PREVERB): Az elváló igekötők külön tokenként vannak jelen a mondatokban, MSD kódjuk **Rp**, például:

Zsuzsa jött hozzám [$_{\text{PREVERB}}$ el] .

Tagadószók (NEG): A tagadószavakat MSD kódjuk alapján azonosítjuk, ez **Rm**, például:

Zsuzsa [$_{NEG}$ nem] jött el hozzám .

Igei csoport (V_): Az igék MSD kódjuk alapján ismerhetők fel. A kód első eleme **V**, a harmadik eleme pedig nem **n** (ez a főnévi igenevek jellemzője). A ***verb_index (V0)*** attribútuma a vonzatkeretes igét azonosítja egy igei lista alapján. Második attribútuma, a ***preverb_ref*** akkor kap értéket, ha van a mondatban az igéhez tartozó elváló igekötő. Az attribútum értéke az igekötői címke, tehát a PREVERB mondatbeli azonosítója. Amennyiben az ige igekötős, de az igekötő nem elváló, akkor a *preverb_ref* helyett a ***preverb_body*** attribútum szerepel. Ennek értéke az igekötő betűkkel leírt alakja. Az ige tagmondatban jelen lévő bővítményeinek lehetséges attribútumai a következők:

- ***idref***: értéke a bővítmény mondatbeli azonosítója
- ***type***: a bővítmény szintaktikai címkéje
- ***role***: a bővítmény esetragjának kódja vagy ennek hiányában a mondatbeli szerep kódja, mely több mint 30 féle lehet, például: alany (NOM), tárgy (ACC), birtokos (GEN), részeshatározó (DAT), eszközhatározó (INS), stb. (a példában: időpont (TLOCY), essive (ESS)).
- ***obl***: a bővítmény kötelező-e?; értéke YES vagy NO

Az 5.1. ábrán szereplő példamondat treebank reprezentációja:

[$_{1:CP}$ [$_{2:NP}$ *Ági*] [$_{3:NP}$ [$_{ADJP}$ *minden*] *rokonát*] [$_{4:ADVP}$ *tegnapelőtt*]

[$_{5:V\_}$ [$_{V0}$ *látta*] ⊕] [$_{6:NP}$ *vendégül*] .]

a *látta* ige vonzatkerete: ⊕ = {2:NP_NOM, 3:NP_ACC, 4:ADVP_TLOCY, 6:NP_ESS}

Főnévi igenevek (INF): A főnévi igenevek ún. infinit, azaz időjel nélküli alakulatok, önállóan nem szerepelhetnek mondat állítmányaként. A főnévi igenév MSD kódja **V** betűvel kezdődik, harmadik tagja pedig **n**. A főnévi igenevek INF0 címkében vannak. Mindig csak ez az egyetlen szó kerül az INF0-ba, például:

[$_{INF\_}$ [$_{INF0}$ *találni*]]

Határozói igenevek (PA): A határozói igenevek MSD kódja **Rv**, képzőjük –va, -ván. A főnévi igenevek PA0 címkében vannak. Mindig csak ez az egyetlen szó kerül az PA0-ba, például:

[$_{PA\_}$ [$_{PA0}$ *hallhatva*]]

5.3. Felszíni szintaktikai elemzés

A ***felszíni szintaktikai elemzés (Shallow Parsing)*** során nem törekszünk arra, hogy feltárjuk a teljes szintaxist és ez olyan egyszerűsítésekre ad lehetőséget mely által az elemzési fázis felgyorsítható és a felismerés pontossága is javítható. Ilyen leegyszerűsített feladat a szócsoportok határait név nélkül megadó zárójelezés (***bracketing***) vagy a legbelső/legkülső főnévi csoportok határainak meghatározása (***base-NP/top-NP chunking***).

Számos olyan alkalmazásról tudunk, ahol elegendő a szövegek felszíni szintaktikai elemzése. Ilyen például az automatikus ***információkinyerés (Information Extraction)*** vagy a ***szöveg kivonatolás (Text Summarisation)*** is.

Az itt leírt Szeged Treebank 1.0 verzió ilyen alkalmazásokban került felhasználásra, illetve további nyelvészeti, nyelvtechnológiai kutatások számára megbízható kiindulópontként szolgál.

5.3.1. Főnévi szerkezetek felismerése

A főnévi szerkezetek (NP-k) a legfontosabb elemei az igei vonzatkeretnek, mivel annak legfontosabb és leggyakrabban előforduló egységei (pl. alany, tárgy, stb.) szinte kizárólag NP-k. Emiatt az olyan alkalmazásoknak, melyek néhány alapvető mondattani információ alapján is hatékonyan tudnak működni, elegendő támpontot jelent az NP (főleg a top-NP) szerkezetek kigyűjtése az elemzendő mondatokból.

A főnévi szerkezetek felismerését a Szeged Treebank 1.0 kidolgozása után a [Hócza04a] cikkben szereplő módszer NP faminták segítségével oldotta meg, ami lényegében tekinthető a 4. fejezetben ismertetett komplex módszer egy NP felismerésre alkalmazott változatának. Az említett cikk előtti időszakban alkalmas tanító korpusz híján kevés magyar nyelvre kidolgozott NP-felismerő készült. Az egyik ilyen rendszer az unifikációs frázisstruktúra-nyelvtant alkalmazó HumorESK [Kis03], a másik módszer [Váradi03] a CLaRK rendszer [Simov01] segítségével végzett NP felismerést reguláris szabálycsoportok többszintű (kaszkád) elvű alkalmazásával.

A NP-felismerés az eddig bemutatott korpusz alapú megközelítésben két részre bontható: modell tanulásra és a tanult modellen alapuló elemző algoritmus futtatására. A 4. fejezetben ismertetett általános módszerben NP-k esetén a tanulás és a felismerés is bizonyos mértékben leegyszerűsödik, valamint lehetőség nyílik a helyzet kihasználására speciális módszerek alkalmazásával. Mindez javítja a hatékonyságot, azaz gyorsabb és

pontosabb eredményt kapunk, mintha a teljes szintaktikai elemzés eredményéből nyernénk ki az NP-ket.

A mintatanulás során – ha vannak is – nem kell foglalkoznunk a nem NP szócsoportokkal, ezáltal a mondat teljes szintaxisfája több kisebb NP részfára bomlik, melyek belseje is leegyszerűsödik, nem lesznek benne nem NP szócsoportok és ezért ezek mélysége is lecsökken. A faalak-típusokkal (beágyazás, füzér) végzett famintagyűjtést a top-NP-k (más NP által nem befoglalt NP-k) belsejében kell elvégeznünk, és ha alkalmazunk mélységi korlátot a nem NP szintek összevonása miatt nagyobb szerkezeteket is ki tudunk gyűjteni, ami később növelheti az elemző pontosságát. A kigyűjtött kiinduló NP faminta halmaz alapján a tanulás az RGLearn algoritmussal történt a 4.1.4. fejezetnek megfelelően és a további lépések is (statisztikai valószínűségek hozzáadása, modell paraméterek optimalizálása) az általános módszer szerint volt végrehajtva.

Az NP-felismerés szintén egyszerűbb feladatot jelent mintha teljes szintaktikai elemzést kellene végrehajtanunk, mert a mondat a feltételezhető top-NP-k mentén felbomlik több kisebb önállóan is elemezhető részre. Vannak olyan szavak illetve tokenek, melyek szinte biztosan nincsenek benne NP-ben, vagyis az NP-k által lefedett mondatrészeken kívül helyezkednek el, azaz NP felismerés szempontjából kisebb darabokra tagolják a mondatot. Ilyenek tokenek például az igék, igenevek és a tagmondat határait jelző írásjelek és kötőszavak.

A NP határok jóslására egy gépi tanulással felkészített tagger a legalkalmasabb. A feladat úgy fogalmazható meg, hogy egy adott szópozícióhoz annak környezete alapján rendeljünk hozzá a következő 5 címke valamelyikét: NP eleje (B), NP belső szava (I), NP vége (E), egy tagú NP (BE) vagy NP-n kívül esik (O). Ez lényegében egy HMM-el megoldható címkézési feladat [Charniak93], vagy felügyelt tanulással (például C4.5 [Quinlan93]) megoldható osztályozási feladat, vagy több módszer kombinációja optimalizált súlyok szerint történő szavazással, általában ez utóbbi módszerrel lehet elérni a legnagyobb pontosságot.

A tagger által visszaadott információ csak arra elég, hogy szegmentáljuk a mondatot NP és nem NP tartományokra, arra viszont nem elég, hogy feltárja a pontos NP szerkezetet. Az olyan szócsoportok esetén, mint például [$_{NP}$ [$_{NP}$ *a fiú*] *bátyja*] egy NP kezdő szó (*a*) két NP vége szóhoz (*fiú*, *bátyja*) is tartozik. Ebből az következik, hogy utóelemzésre van szükség a helyes zárójelezés megállapításához. Erre legjobb megoldás, ha létező szerkezeteket, vagyis a korpuszból tanult famintákat próbálunk meg illeszteni az NP tartományokra. Erre a feladatra bottom-up chart parsert használunk. Tehát összefoglalva NP-felismerés esetén annyi a specialitás a 4. fejezetben leírt általános módszerhez képest, hogy a chart parser működését megtámogatjuk a tagger

által visszaadott információkkal és nem a teljes mondatra, hanem elkülönült mondatrészekre futtatjuk.

Az előzőekben vázolt módszer és a vele elért eredmények [Hócza04a] cikkben kerültek ismertetésre. A kiértékelés a Szeged Treebank két szekcióján történt. Az első szekció a az első 5 témakörből (szépirodalom, iskolai fogalmazások, újságcikkek, jogi szövegek, számítástechnika – röviden általános szövegek) kerül ki, a második szekció pedig a 6. témakör (üzleti hírek) volt. A szekciók mondatonként véletlenszerű választással 90% tréning és 10% teszt részekre voltak felosztva. Ezt a felosztást 10-szer megismételtük és az eredményeket átlagoltuk a tenfold crossvalidation módszer szerint. Az előfeldolgozás az általános szövegekből átlagosan ~300 ezer top-NP részfát gyűjtött ki, melyből a tanuló eljárás ~7500 NP famintát generált. Ugyanezek a statisztikai értékek üzleti hírek esetén: ~51 ezer top-NP illetve ~2 ezer NP faminta. A kiértékelés az elkülönített 10% teszt mondaton volt elvégezve. Az eredmények összefoglalását az 5.3. táblázatban láthatjuk.

Szekció	Pontosság	Fedés	$F_{\beta=1}$
Általános szövegek	75.72%	81.69%	78.59%
Üzleti hírek	79.86%	86.63%	83.11%

5.3. Táblázat: Az NP-felismerés kiértékelése általános szövegeken és üzleti híreken

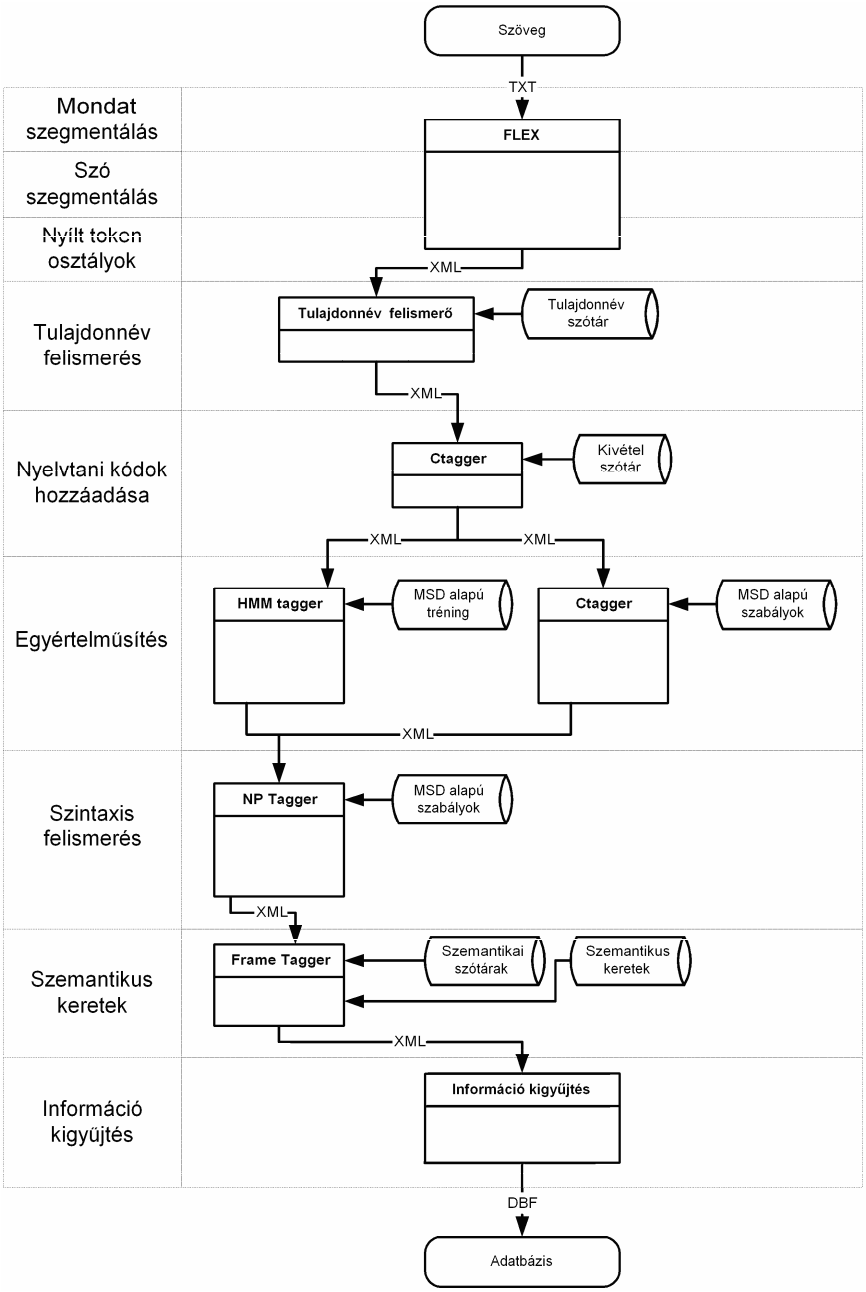
Az eredményekből az a tanulság vonható le, hogy a viszonylag homogén tartalmú és ismétlődő szófordulatokat gyakran alkalmazó üzleti hírek esetén szignifikánsan jobb eredményeket kaptunk mint a heterogén, 5 különböző témakörből álló általános szövegeken.

5.3.2. Információkinyerés üzleti hírekből

Az ***információkinyerés (Information Extraction)*** célja, hogy a szövegek tartalmából a releváns adatokat a lehető legpontosabban megtalálja (és az irreleváns adatokat kiszűrje), hogy azokból egy gépileg feldolgozható, strukturált formában tárolt, lekérdezhető adathalmazt állítson elő. Ehhez nincs szükség a szövegek értelmének elemzésére mint a ***szöveg megértés (Natural Language Text Understanding)*** esetén. Ugyanakkor az információkinyerést el kell különítenünk az ***információkigyűjtéstől (Information Retrieval)*** is, melynek célja az adott feltételeknek megfelelő vagy egy lekérdezésre illeszkedő adat (dokumentum) megkeresése, melyhez nincs szüksége

strukturált adathalmazra, ehhez elegendő releváns szavak és kifejezések statisztikáját összevetni referencia dokumentumok adataival.

Az NKFP 2/017/2001 projekt témája az automatikus információkinyerés (gazdasági, üzleti) rövidhírekből. A projekt célkitűzése volt egyrészt egy információkinyerő technológia kifejlesztése a gazdasági és üzleti hírek témakörére, másrészt a kifejlesztett technológia implementálása egy program prototípus formájában. A szerző jelentős részt vállalt ennek a prototípusnak a kidolgozásában, a felszíni szintaktikai elemző modul beépítésén túl elkészítette a prototípus kereteljárását, amely összekapcsolta az egymásra épülő, részfeladatokat megoldó modulok működését. Egy ilyen elven működő ***„toolchain”*** került ismertetésre a [Hócza03b], [Hócza04b] cikkekben. A prototípus első verziójában a szerzőnek további modulok elkészítésében és beillesztésében meghatározó is szerepe volt, ezek: a mondat- és szószegmentálás, szófaji egyértelműsítés, tulajdonnév felismerés és szemantikus keretek illesztése. A toolchain felépítésének vázlata az 5.2. ábrán látható.



5.2. Ábra: Az információkinyerésre alkalmazott toolchain vázlata

Tekintsük át a toolchain moduljait:

Mondatok, szavak, speciális tokenek és tulajdonnevek szegmentálása: Az összefüggő forrásszövegen, mely csak hírek szerinti tagolást tartalmaz, be kell jelölni az egyes mondathatárokat, valamint a mondaton belüli a nyelvi egységeket is azonosítani kell. A mondatok és szavak szegmentálásától nem választható el a tulajdonnevek, és különböző speciális tokenek felismerése, hiszen ezek bizonyos esetekben összetéveszthetőek (például pont után nagykezdőbetűs szó lehet egy új mondat kezdete, de lehet bizonyos esetekben tulajdonnév is). A tulajdonnevek felismerését legegyszerűbb listák alapján végezni (kereszt-, család- és földrajzi nevek, stb.), azonban

az ilyen listák sosem tekinthetők teljesnek, ezért az ilyen jellegű csoportokat ***nyílt tokenosztályoknak*** is nevezzük. A mondat- illetve szószegmentálás, és a speciális tokenek felismerésének elvégzésére automata és döntési fa alapú módszerek kerültek kifejlesztésre, melyek tanuló és teszt adatbázisát a Szeged Korpusz képezte.

Nyelvtani kódok hozzáadása: A további modulok működése szövegben előforduló szavak szófaji kódjára épül, melyek bejelölésére a MorphoLogic Kft által fejlesztett Humor elemzőt [Prószéky99] használtuk fel. Mivel a Humor elemző nem MSD kódrendszert használ (melyre a korpuszból tanult modelljeink is épülnek), a Humor által visszaadott kódot egy keretprogram, a CTagger futás közben átkonvertálja. Azokban az esetekben amikor a konvertálást nem lehet a korpuszban annotált állapotokkal konzisztensen végrehajtani a szó elemzését egy kivételszótárból vesszük.

Szófaji egyértelműsítés (POS tagging): A morfológiai elemzés általában egy szóhoz több kódot is rendel (pl. a *várat* szó lehet műveltető ige vagy tárgyesetű főnév is). Ezért a szókörnyezet alapján ki kell választani a kontextusnak megfelelő kódot. Ezt a műveletet ***szófaji egyértelműsítésnek (POS tagging)*** nevezzük. Ennek a végrehajtása a látszattal ellentétben nem igényel mélyebb mondatelemzést, a szó/címke ***n-esek (n-gram)*** előfordulási valószínűségén alapuló módszerek mint például a HMM kiváló eredményt adnak 95% feletti pontossággal. Tovább lehet javítani az eredményeket, ha a szópozíciókhoz rendelt kódot több módszer jóslatának súlyozott kombinációjával állítjuk elő. A toolchain szófaji egyértelműsítését a egy HMM alapú és egy a CTagger-be beépített gépi tanulási szabályokon alapuló módszer kombinálásával kaptuk.

Szintaxis (NP) felismerés: Mivel a gazdasági hírekben előforduló események szereplőit (az eseményt leíró, fontos információt hordozó összetevő) gyakran a szónál nagyobb nyelvtani egységek írják le, így kritikus fontosságú a jó IE rendszer szempontjából, hogy az újsághírek szövegén ezeket a nyelvi elemeket jó pontossággal azonosítani tudjuk. Az üzleti hírek doménjét figyelembe véve, ezek a szereplők az esetek legnagyobb többségében főnévi szerkezetek, ezért ezek felismerésére fordítottunk kiemelt figyelmet. Az erre a célra alkalmazott NP-Tagger az 5.3.1. fejezetben leírt módszer szerint volt implementálva és felkészítve az Szeged Korpusz 1.0 üzleti híreket tartalmazó szekcióján.

Szemantikus keretek illesztése: A szintaktikai elemzés után a hírek szövegét különböző szemantikai jegyekkel kell ellátni. Ez a feladat egy segédtáblázat segítségével valósul meg, mely tartalmazta a korpusz üzleti hírekben szekciójában gyakran előforduló szavakat és azokhoz további tulajdonságokat ~30 bináris szemantikai attribútumot rendel (pl. tulajdonnév, helység, intézmény, cég, pénzügyi, emberi, testrész, elvont, stb.). Ezeket a tulajdonságokat FrameTagger az NP-hez az NP-k feje alapján rendeli hozzá.

A következő lépésben a FrameTagger megkeresi a tagmondatokra legjobban illeszkedő ***szemantikus keretet (frame)***, melyek az üzleti hírek különféle típusainak beazonosítására kézzel készültek. A szemantikus keretek központi helyén állnak a különféle tranzakciókat jelentő igék (*vesz*, *elad*, *alapít*, *csődbe megy*, stb.), illetve az azonos tranzakciók lehetséges kifejezési módjai, melyben jelentős szerepet kapnak az adott ige lehetséges szinonímái (*vesz*, *megszerez*, *felvásárol*, stb.). A szemantikus keretek további részeit alkotják az igék vonzatkeretének elemei, az ún. ***szemantikus szerepek (role)***, melyek a tranzakció további összetevőit azonosítja be (pl. *vevő*, *eladó*, *áru*, *mikor*, *hol*, *mennyiért*, stb.). A szemantikus keretek alkalmazása során a FrameTagger egyrészt (szintaktikai és szemantikai) illesztési problémát old meg, másrészt keresési problémát is jelen, mivel ~50 témakör ~800 szemantikus kerete közül kell a legjobbat kiválasztani. Egy adott szemantikus keret illeszkedésének mértékét az illeszkedő és össze feltétel súlyozott aránya adja meg és az így meghatározott pontszám lehetővé teszi heurisztikák alkalmazását a keresés gyorsítására.

Információ kigyűjtés: Ebben a lépésben a FrameTagger által sikeresen kitöltött szemantikus keretek átírása történik adatbázis rekordokba azokban az esetekben amikor az illeszkedés mértéke elér egy megadott küszöböt.

5.4. Teljes szintaktikai elemzés

Az IKTA 37/2002 K+F projekt témája a mondatszintaxis gépi tanulása volt. A projekt egy számítógépes tanulóalgoritmusok által támogatott szintaktikai elemzési módszer kialakítását célozta. További cél volt egy szintaktikailag részletesen elemzett szöveges adatbázis (a Szeged Treebank 2.0) kialakítása, valamint nyílt tokenosztályok (tulajdonnevek, dátumok, számok, Internet és e-mail címek, és egyéb nehezen kezelhető szerkezetek) felismerésére és kezelésére alkalmas módszer kidolgozása is. A szerző meghatározó rész vállalt a projekt keretén belül a teljes szintaktikai elemzési módszer kifejlesztésében és alkalmazásában, valamint egyéb munkálatokban (pl. a kézi annotálást támogató grafikus szerkesztő, valamint az annotátorok munkáját etalon elemzésekkel kiértékelő program elkészítése).

A teljes szintaktikai elemzés több szempontból nehezebb probléma mint az NP-felismerés, mivel egy helyett sokféle szócsoport van, ezek mélyebb, összetettebb szerkezeteket alkotnak, emiatt a tanulás több mintát állít elő, valamint (a felszíni elemzéssel ellentétben) teljes szintaxisfát kell építeni. De a legnagyobb problémát magyar nyelv esetén az igei vonzatkeret modellezése jelenti, mivel azt – mint azt az 5.1. fejezetben kifejtettük – a vonzatkeret elemek szabad átrendezhetősége, elhagyhatósága és nem összefüggősége miatt generatív jelegű szabályokkal nem lehet ábrázolni.

5.4.1. Az igei vonzatkeret modellezése

Jelöljön v_i egy adott igét, mellyel kapcsolatban az $<a_1,..,a_m>$ attribútum-vektorral jellemezzük azt, hogy mely vonzatkeret elemek vannak jelen egy szintaktikai elemzésben. Ennek valószínűsége $P(a_1,..,a_m \mid v_i)$ melynek becsléséhez hatalmas tanulóadat kellene, ezért azt feltételezzük, hogy a vonzatkeret elemek feltételesen függetlenek egymástól. Azt a szintaktikai elemzést keressük ami rögzített v_i mellett a legnagyobb valószínűségű vonzatkeretet adja:

$$\begin{aligned} <a_1,..,a_m>_{\max} &:= \arg\max_{<a_1,..,a_n>} P(a_1,..,a_n \mid v_i) \\ &= \arg\max_{<a_1,..,a_m>} \prod_{j=1}^{m} P(a_j \mid v_i) \\ &= \arg\max_{<a_1,..,a_m>} \prod_{j=1}^{m} P(v_i \mid a_j) P(a_j) \end{aligned} \qquad (5.1)$$

Az utolsó átalakításnál *Bayes szabályt* alkalmaztunk és egyúttal kihasználtuk azt, hogy $P(v_i)$ minden vonzatkeret esetén egyforma. A $P(a_j)$ annak a részfának a valószínűsége, mely a j-edik vonzatkeretet adja, a $P(v_i \mid a_j)$ valószínűségeket pedig a korpuszon becsüljük a relatív gyakoriságok alapján:

$$P(v_i \mid a_j) \approx \frac{C(v_i, a_j)}{C(a_j)} \qquad (5.2)$$

A vonzatkeret felismeréséhez többszintű modell és többszintű elemzés szükséges. A lehetséges vonzatkeret elemeket alkotó részfákat és azok valószínűségét bottom-up elemzéssel számítjuk ki chart parser segítségével. A beazonosított szócsoportokhoz fejük alapján meghatározzuk a lehetséges vonzatkeretbeli szerepét. Általában – hacsak nem azonos szerepű igék halmozásáról van szó – egy tagmondat egy igét tartalmaz, ha több tagmondatunk van több igével, akkor a vonzatkeret elemeket több csoportba kell rendeznünk és ennek az elrendezésnek kell megtalálnunk az optimumát, így az (5.1)-ben megadott keresés n darab mondatbeli ige esetén az alábbiak szerint módosul:

$$(<a_{1_1},..,a_{1_m}>,..,<a_{n_1},..,a_{n_m}>)_{\max} := \arg\max_{<a_{1_1},..,a_{1_m}>,..,<a_{n_1},..,a_{n_m}>} \prod_{i=1}^{n}\prod_{j=1}^{m} P(v_i \mid a_{i_j}) P(a_{i_j}) \qquad (5.3)$$

A lehetséges vonzatkeretelem-változatok között lehetnek átfedések, azaz a chartban szereplő az egyik részfa belelóghat egy másikba. Egy érvényes csoportosítás: $<a_{1_1},..,a_{1_m}>,..,<a_{n_1},..,a_{n_m}>$, ami nem tartalmazhat átfedéseket, és maximálisan lefedi a bottom-up elemzéssel megtalált szócsoportokat. A feladat az összes érvényes csoportosítás felsorolása, kiértékelése és az optimum megkeresése. Ez egy a chart parser bonyolultságához viszonyítva kedvezőtlenebb algoritmussal, ***visszalépéses kereséssel***

(backtracking) oldható meg (ha nem lennének átfedések akkor címkéző (tagger) algoritmust is alkalmazhatnánk).

Az igei vonzatkeretek fentebb leírt kezelési módjának nem az a célja, hogy újabb chart élek, azaz VP-k keletkezzenek, mert mint azt az 5.1. fejezetben kifejtettük, ez magyar nyelvre generatív megközelítésben nem lehetséges. Ez a kitérő a szabályszintek alkalmazása között arra jó, hogy az 5.2.3 részben leírt *V_* adatszerkezetet (az ige és a vonzatkeretének leírását) előállítsuk és ez alapján kizárjuk a nem a maximális valószínűségű vonzatkeret(ek)be tartozó chart éleket. Ezután újra folytatódhat chart parser működése tagmondatok és magasabb szintű szerkezeteket felismerő szabályszintek alkalmazásával a chart megmaradt „egyértelműsített” éleiből kiindulva.

5.4.2. Teljes szintaktikai elemzéssel elért eredmények

A [Hócza06a] cikkben a szerző beszámol az általa kifejlesztett famintákon alapuló szintaktikai elemző működési elveiről, valamint annak a Szeged Korpusz 2.0 adatain végzett felkészítéséről és a kiértékelés eredményeiről. A teljes szintaktikai elemző megvalósításának részletei a 4. fejezetben leírtak szerint történt, valamint az igei vonzatkeretek az előző (5.4.1.) fejezet szerint voltak kezelve. A szintaktikai elemző modelljének felkészítése és kiértékelése során ugyanazt a felosztást (általános szövegek, üzleti hírek) és tesztelési módszert (tenfold crossvalidation) alkalmaztuk, mint korábban az NP felismerésnél, hogy legyen összehasonlítási alapunk a másik módszerrel. Az így kapott eredményeket az 5.4. táblázatban láthatjuk.

Szekció	Pontosság	Fedés	$F_{\beta=1}$
Általános szövegek	82.27%	79.39%	80.78%
Üzleti hírek	85.83%	81.46%	83.59%

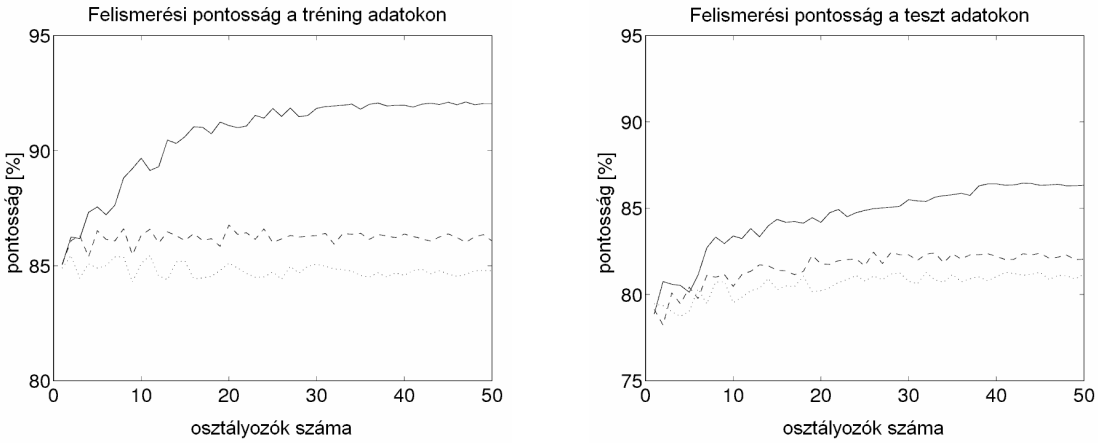
5.4. Táblázat: Az teljes szintaktikai elemzéssel elért eredmények

5.4.3. Faminták tanulásának javítása Boosting algoritmussal

A szerző és társai a megtanult faminták felismerési pontosságának javítására alkalmazták a Boosting algoritmust [Hócza05a].

Az elért eredményeken javítani lehet, ha több módszert kombinálunk. A Boosting algoritmus [Shapire90] egy adott tanuló módszerrel több modellt is készít úgy, hogy a

tanulópéldákat különböző súlyeloszlással (gyakorisággal) adagolja a tanuló algoritmusnak, mely folyamat során a kiértékelés eredményeit is figyelembe veszi. Ezt a módszert alkalmaztuk az üzleti híreken az faminták felismerési pontosságának javítására. A kísérletek az mutatták, hogy míg az alap kombinációs módszerek, melyek kombinációs sémája a minimum, maximum, szorzat vagy az összeg szabály alapján működött, nem hoztak szignifikáns eredményt, addig a Boosting algoritmussal átlagosan 7.5%-os javulást sikerült elérni a kiinduló állapothoz képest. A kísérleti eredmények alapján készült grafikon az 5.2. ábrán látható.



5.2. Ábra: Kombinációs sémák osztályozási pontossága tréning és teszt adatokon (vonal: Boosting, szaggatott: Összeg szabály, pontozott: Max szabály)

5.4.4. Különféle szintaktikus elemzési módszerek összehasonlítása

A [Hócza05b] cikkben a szerző és társai beszámolnak arról, hogy kialakítottak a Szeged Treebank 2.0 állományaiból egy mintaadatbázist és javasolták, hogy az eddig elkészült és az ezután kifejlesztett magyar nyelvű szintaktikai elemzők a pontos összehasonlíthatóság érdekében ezen legyenek felkészítve és kiértékelve.

Az angol nyelv szintaktikai elemzésének meglehetősen nagy szakirodalma van, mivel lényegesen korábban kialakítottak szintaktikailag annotált szöveges adatbázisokat. Ilyen például a Penn Treebank [Marcus93], amelyen a tudományos cikkekben mérni szokták az angol nyelvre készült elemzők pontosságát. A 2-21-es szekció használható a módszer felkészítésére a 23. pedig a kiértékelésre. Mivel az angol nyelvre írt elemzők hagyományosan ilyen feltételek szerint vannak kiértékelve és publikálva ez egy új módszer számára gyorsan elérhető és pontos összehasonlítási lehetőséget biztosít. Ennek az ötletnek a mintájára a szerző és társai kialakítottak a Szeged Treebank 2.0 állományaiból a szintaktikai elemzők felkészítésére és kiértékelésére alkalmazható mintaadatbázist.

Két módszer pontos összehasonlítása megkívánja, hogy ugyanazokon az adatokon alkalmazzuk őket. A magyar nyelvre eddig még nem volt kialakítva olyan nyilvános adatbázis, mint az angol esetén a Penn Treebank, ezért, hogy ez magyar nyelvű szövegek esetén is megvalósítható legyen a következő módszert dolgoztuk ki:

- Kifejlesztettünk magyar nyelvre egy teljes szintaktikai elemzőt (Hócza, 2005), és alkalmaztuk a Szeged Treebank adatain.
- A Szeged Treebank adataiból kialakítottunk egy mintaadatbázist, amely a jövőben lehetőséget teremt minden érdekelt számára, hogy a treebank adatait felhasználva kipróbálja a szintaktikai elemzőjét, és összevesse annak hatékonyságát a magyar nyelvre alkalmazott más módszerekével.
- Néhány további módszert is kipróbáltunk a mintaadatbázison (5.5. táblázat).

Példák véletlenszerű kiválasztása még nem garantálja azt, hogy egy módszer tesztelésénél ne kapjunk torz eredményeket. Például ha véletlenül egymáshoz nagyon hasonló példák kerülnek a tesztadatokba, a kapott végeredmény pontossága több százalékkal is eltérhet egy másik felosztással kapott értéktől. A nemzetközi szakirodalomban az eredmények közlésénél ennek a problémának az elkerülésére alkalmazzák a ***tenfold crossvalidation*** módszert, melynek lépései a következők:

(1) A példákat véletlenszerűen szétosztjuk 10 egyforma méretű csoportba. Előállítjuk a tanító állományokat úgy, hogy mindegyik 9 különböző csoportból adódjon össze (mindig 1 kimarad), valamint a tanító állományokhoz hozzárendeljük azok teszt párját, ami kimaradt a tréningből.

(2) Lefuttatjuk a tanulást a 10 tréningcsoportra, majd elvégezzük a tesztelést a tréningállomány megfelelő teszt párján.

(3) A teszteredményeket összesítjük és átlagoljuk; ez az átlag lesz a példákon végzett tanulás végső eredménye.

A Szeged Treebank mintaadatbázisán 4 módszert alkalmaztunk a teljes szintaxis felismerésére. Az alap (baseline) módszer a treebank adataiból kigyűjtött környezetfüggetlen valószínűségi nyelvtant (PCFG) használ, amely chart parsing segítségével építi fel a teljes szintaxisfát. A második módszer esetén a chart parser nyelvtanát kb. 22 ezer szabály képezte, amelyet nyelvész szakértők állítottak elő. A harmadik egy memória alapú módszer volt, amely a treebank adataiból kigyűjtött teljes mondatok szintaxisfáját illesztette a nyelvtani kódok figyelembevételével. A negyedik módszerben a chart parsert gépi tanulással előállított famintákra alkalmaztuk.

Szekció	Pontosság	Fedés	$F_{\beta=1}$
Baseline	53.64%	58.60%	56.01%
Szakértői szabályok	54.92%	56.25%	55.58%
Mondat memória	92.34%	3.66%	7.04%
Faminták	80,16%	71,29%	75.47%

5.5. Táblázat: A mintaadatbázison különféle módszerekkel elért eredmények.

5.4.5. Magyar szintaktikai elemző beépítése gépi fordító rendszerbe

Az NKFP 2/008/2004 kutatás és fejlesztési projekt egy magyar-angol gépi fordítórendszer létrehozását tűzte ki célul. A szerző ehhez a projekthez kapcsolódóan előállította egy gépi tanulási technikákon alapuló meglévő rendszer magyar-angol fordításra alkalmas bővítményét.

A ***gépi fordítás (Machine Translation, MT)*** feladata egy adott természetes nyelven elkészült szöveg automatikus átfordítása egy másik természetes nyelvre. A praktikus oka ennek a területnek az, hogy számos nyelven léteznek különféle dokumentumok (pl. az interneten), melyeket az emberek szeretnének elolvasni, de lehet, hogy nem ismerik azt a nyelvet amelyen az adott dokumentum készült. Egy megfelelő hatékonyságú gépi fordító rendszer gyorsan és jó minőségben szolgáltat megoldást erre a problémára.

Manapság a legjobb megoldást a ***statisztikai gépi fordító (Statistical Machine Translation, SMT)*** rendszerek adják. A *2005 SMT workshop* [Burbank05] célja egy olyan bárki számára hozzáférhető, nyílt forráskódú rendszer, a GenPar kifejlesztése volt, ami a mintaként megadott fordítási nyelvpárok mintájára lehetőséget biztosít bárki számára, hogy új nyelvpárt integráljon a rendszerbe.

A szerző által megvalósított kiegészítés [Hócza06b] eredményeként egy új, magyar-angol nyelvpár került a GenPar rendszerbe, azaz egy inputként beadott magyar szövegnek a rendszer outputjaként megkapjuk az angol nyelvű fordítását. Ennek a végrehajtása almodulok toolchain-szerű futtatásával történt melynek az angol nyelvű moduljai (tokenizáló, POS-tagger, szintaktikai elemző) a Penn Treebank [Marcus93] adatain felkészítve rendelkezésre álltak. A magyar nyelvű modulok a Szeged Treebank 2.0 [Csendes05] felhasználásával készültek. A Ratnaparkhi POS-tagger [Ratnaparkhi97] figyelemreméltóan jó eredményt ért el magyar nyelvű szövegeken MSD kódolással, például ismeretlen tulajdonnevek ragozott alakjának is jól meghatározta a kódját, és még morfológiai elemzőt sem igényelt a futtatása.

Az előzőekben ismertetett teljes szintaktikai elemzőnket a magyar nyelvű szófajilag egyértelműsített mondatokon alkalmaztuk. A GenPar fordító moduljának működése a nyelvpárok szintaktikai elemzéseiből vett azonos szerepű részfák izomorfizmusán alapul megadva az egyik részfából a másikba való átjárás transzformációs lépéseit. Ezen a téren gondot jelentett a magyar vonzatkeret ábrázolása (V_) és az angol VP közti különbség, ugyanis V_ nem tartalmazza a vonzatkeret elemeit, csak össze van velük indexelve. Ezért a szintaktikai elemzés végeztével a vonzatkeret elemeit átcsoportosítottuk egy újonnan bevezetett VP címke alá.

A GenPar betanításához szükség volt még párhuzamos mondatokra, azaz magyar nyelvű mondatokhoz rendelt angol fordításra. Ezeket a mondatpárokat a Hunglish Corpus [Varga05] adattárából választottuk ki, 5 ezer tanító és 500 teszt mondatpárt.

A gépi fordítás esetén a kiértékelés, azaz a fordítás jóságának megadása ún. ***BLEU mérték (BLEU score)*** [Papineni02] segítségével történik, ami a gépi fordítással előállított mondat és a referencia mondat egyező ***szó n-eseinek (n-grams)*** az arányát fejezi ki az összes lehetőséghez képest. Szokás több referencia mondatot is megadni, mivel egy adott mondatra általában több helyes fordítás is lehetséges. Az *n* értéke általában 1-től 4-ig terjed, a BLEU-4 ebből a 4 értékből képez egyet. A BLEU-4 mérték egy 0 és 1 közötti szám, ahol 1 jelenti a teljes egyezést. Általában már 0,4-től már jó minőségű fordításról beszélhetünk.

A GenPar rendszer lehetőségeinek a feltérképezésére 4 különböző összetételű prototípus készült a magyar-angol fordításhoz. A prototípusok moduljainak összeállításához ötleteket a már beintegrált nyelvpárok (francia-angol, arab-angol, angol-angol) megvalósításából lehetett nyerni. Az alább felsorolt prototípusokkal elért eredményeket az 5.6. táblázat tartalmazza:

Prototípus 1: A mondatokhoz nem készült szintaktikai elemzés és POS-tagging sem. Ez a francia-angol nyelvpáron egész jól működött. Az eredmények azt mutatják, hogy a felhasznált ~5500 mondat nem elég ahhoz a magyar nyelvre jellemző ragozott szóalakokat kezelni tudja a rendszer (nagyon sok szóalak csak egyszer fordult elő).

Prototípus 2: A szóalakok számát lemmatizálással lecsökkentettük, azaz a mondatokban a ragozott szóalakok helyett csak szótövek szerepeltek. Az eredmények észrevehetően javultak 1-hez képest.

Prototípus 3: A magyar szövegeken futtattuk a POS-taggert. A szópozíciókon szótő és szófaj szerepelt, azaz növekedett a szóalakok száma. Ez növelte a bizonytalanságot és így romlott az eredmény..

Prototípus 4: A 3-as prototípust kiegészítettük szintaktikai elemzéssel. A szintaktikai információk ugrásszerűen javítottak az eredményeken.

	BLEU-4	**1-gram**	**2-gram**	**3-gram**	**4-gram**
Prototípus 1	0.033	0.343	0.094	0.059	0.035
Prototípus 2	0.114	0.457	0.197	0.106	0.068
Prototípus 3	0.085	0.374	0.134	0.076	0.049
Prototípus 4	0.191	0.521	0.275	0.186	0.135

5.6. Táblázat: Különféle magyar-angol prototípusok eredményei.

5.5. Konklúzió

Ebben a fejezetben bemutattuk a szerző a szintaktikai elemzők különféle változataival elért eredményeit és azok alkalmazásait. A fejezet első részében átnéztük, hogy a magyar nyelv milyen specialitásokat, nehézségeket jelent szintaktikus elemzési szempontból és ehhez a Szeged Treebank a különböző munkafázisokban milyen információkat biztosított. Ezután bemutattuk a felszíni és teljes szintaktikai elemzéssel elért eredményeket és ezek alkalmazását információkinyerő és gépi fordító rendszerekben.

6. Konklúzió

Az értekezés témája a szintaktikai elemzés volt, melynek a gyakorlati megvalósítása és alkalmazásai magyar nyelvre történtek. A szerző módszereiben szabályalapú gépi tanulási technikákat alkalmazott. A bevezetésben áttekintettük a szerző által elért eredményeket részletezzük.

A 2. fejezetben elemeztük, hogy a természetes nyelvek szintaktikai elemzése milyen követelményeket jelent és milyen nehézségek, problémás jelenségek vannak a természetes nyelvekben és ezekre milyen irányzatok megoldási javaslatok születtek. A fejezet végén bemutatásra került egy a szerző által kidolgozott módszer, a famintákkal történő szintaktikai elemzés.

A 3. fejezetben a gépi tanulás fogalmának bevezetése után részleteztük azokat a technikákat, melyeket a szerző valamilyen szinten felhasznált a szabályalapú szintaktikus elemzési modellek építésében. Ebben a részben ismertetésre kerül egy a szerző által kifejlesztett mintatanuló algoritmus, az RGLearn, melyet faminták tanulására alkalmazhatunk.

A 4. fejezetben leírtuk a szerző által kidolgozott komplett szintaktikai elemző módszert: a modellépítés folyamatát, az elemző algoritmus megvalósítását, a kiértékelés technikáját, valamint az eredmények visszacsatolásával végzett modell-optimalizálást.

Az 5. fejezet a szintaktikai elemzés magyar nyelvre történő alkalmazásának részleteit foglalta össze. A fejezet a probléma leírásával kezdődött: milyen specialitások, nehézségek jellemzik a magyar nyelvet szintaktikus elemzési szempontból, és a rendelkezésre álló korpusz milyen információk felhasználását teszi lehetővé. A további részek a szerző szintaktikai elemzésben elért eredményeit mutatta be, valamint a felszíni és teljes szintaktikai elemzés megvalósított alkalmazásaira hozott példákat, információkinyerő illetve gépi fordító rendszerek részeként.

Függelék

Összefoglalás

Jelen értekezés témája a szintaktikai elemzés, melynek a gyakorlati megvalósítása és alkalmazása magyar nyelvre történt. A szerző módszereiben szabályalapú gépi tanulási technikákat alkalmazott, melyek segítségével egy elemzett korpuszból kinyerhető információk felhasználásával szintaktikai elemzésre alkalmazható modell építhető. A szabályalapú reprezentáció ember számára olvasható módon tárolja a megszerzett ismereteket így lehetőséget biztosít a tudásbázis karbantartására és így szakértői tudással történő kiegészítésre.

A természetes nyelvek szintaktikai elemzése

A természetes nyelvek jelenségeinek ábrázolása kihívást jelent a számítógépes nyelvészet számára, ezen belül a magyar nyelv a szabad szórend és a ragozott szóalakok nagy száma miatt a nehezebben elemezhető nyelvek közé sorolható. A generatív nyelvtanok alkalmazásai megjelenésük idején ígéretes lehetőségnek tűntek, mivel ezeket hatékony algoritmusokkal lehet elemezni, különösen a reguláris és a környezetfüggő nyelvtanok esetén. Azonban hamarosan megjelentek cáfolatok, ellenpéldák, melyek azt mutatták, hogy ezek a nyelvosztályok nem alkalmasak a természetes nyelvek bizonyos jelenségeinek ábrázolása. Ilyen ellenpélda az önbeágyazás, melynek leírására nem alkalmas a reguláris nyelvtan, a keresztező függőségek leírását pedig a környezetfüggő nyelvtanokkal nem lehet megoldani.

Napjainkra a generatív megközelítés háttérbe szorult, ezeket felváltották olyan nyelvelméletek és formalizmusok, melyekben a nyelvi jelenségek minél pontosabb leírása került előtérbe a nyelv generálása helyett. Az egyeztetés és alkategórizálás, például azért jelent problémát, mert környezetfüggetlen nyelvtan alkalmazása esetén csak nagyon sok szabály bevezetésével lehetne leírni a jelenséget, ez a megoldás viszont a nyelvtan méretét a többszörösére növelné. Szintén ilyen jellegű problémát jelent a szabad szórend kezelése. A függőségek ábrázolása, különösen a távoli függőségeké nem oldható meg környezetfüggetlen nyelvtanokkal, mert ezek szabályai csak összefüggő szócsoportokra alkalmazható. A szabályok alkalmazásának statisztikai előfordulások alapján becsült valószínűségét is figyelembe kell vennünk, ha olyan modellt szeretnénk készíteni, ami alapján választani tudunk a többértelműségek miatt kialakult elemzési erdőből. Végezetül a lexikális és strukturális függőségek figyelmen kívül hagyása olyan

elemzési szerkezeteket eredményezhet a gépi elemzésben, melyeket az annotált korpusz nem tartalmaz, mert ezt a szöveg értelmezése alapján kizárhatjuk.

A szerző által bevezetett faminta formalizmus többszintű részfákat ismer fel a leveleire adott reguláris kifejezésekkel leírt minták segítségével. Ha szintaktikai elemzésre többszintű szerkezeteket alkalmazunk a modell várhatóan több elemet (szabályt) fog tartalmazni egy ugyanolyan korpuszon felkészített környezetfüggetlen nyelvtanhoz képest. A faminta formalizmus ezt a növekedést azzal kompenzálja, hogy a famináták a levelek leírása révén, képesek egymáshoz hasonló szerkezetek egy csoportját összefoglalni egyetlen mintában. A famintákban szereplő leírás rugalmassága lehetővé teszi más formalizmusok nyújtotta technikák alkalmazását, amivel kezelni lehet a strukturális függőségeken túl más problémás nyelvi jelenségeket is. A szövegek szintaktikai elemzését a ***chart parser*** algoritmus ([Kaplan73], [Kay86]) famintákra adaptált változata valósítja meg.

Gépi tanulási technikák alkalmazása nyelvtani modellek készítésére

A gépi tanulási módszerek egyik fontos alkalmazási területe a természetes nyelvi problémák, különösen akkor, ha erre a célra rendelkezésre áll egy annotált korpusz, melyből példákat gyűjthetünk egy adott jelenségre. A példák halmaza olyan (x_i, y_i) értékpárokból áll, melyekben az x_i értékek valamilyen objektum vagy esemény leírására szolgálnak, az y_i értékek pedig a következtetést adják meg. Diszkrét y_i értékek esetén ***osztályozásról*** beszélhetünk. Azt az esetet ***felügyelt tanulásnak*** nevezzük, amikor az (x_i, y_i) értékpárok halmaza ismert (például az annotált korpuszból kigyűjthető) és a tanulóprogram feladata egy olyan f függvény megkeresése, melyre $f(x_i) = y_i$ teljesül. Ebben az esetben azt is feltételezzük, hogy f függvény alkalmas lesz előre nem látott x értékek esetén is az y értékek helyes meghatározására. Ezt az elvet ***induktív tanulásnak*** nevezzük. Amikor a cél egy logikai értékű osztályozás tanulása, ezt ***fogalom tanulásának*** hívjuk, ebben az esetben pozitív és negatív példáink vannak attól függően, hogy igaz vagy hamis érték van hozzájuk rendelve.

A szerző által kidolgozott ***RGLearn*** mintatanuló algoritmus bemenete az annotált korpuszból kigyűjtött mintákból képzett pozitív és negatív példák, az alapján, hogy helyes, vagy hibás fedésről van-e szó. A kimenet egy olyan általánosított mintahalmaz, melynek együttes pontossága maximális, azaz a lehető legtöbb pozitív és a lehető legkevesebb negatív példát fedi le. Ez az algoritmus alkalmazva volt szófaji egyértelműsítés szabályalapú modelljének [Kuba04], valamint szintaktikai elemzésre használt famintáknak ([Hócza04a], [Hócza06a]) a tanulására is.

A gépi tanulási módszereknek további alkalmazási lehetőségei is vannak a szintaktikai elemzésre alkalmazható modellek építése során. A szófaji

egyértelműsítésnél használt ***címkéző algoritmus (tagger)*** szócsoportok határainak jóslására is alkalmazható, melynek felkészítése különféle gépi tanulási módszerekkel történhet. Ennek az eredményét felhasználhatjuk a ***felszíni elemzés*** **(*Shallow Parsing*)** során a mondatok szócsoportokra való szegmentálására vagy az alapvető szócsoportok (például ***base-NP***, ***top-NP***) kijelölésére.

A minták halmazára készíthetünk egy valószínűségi modellt, melyet annotált korpusz esetén a ***relatív gyakoriságok*** alapján számíthatunk ki, annotált korpusz hiányában pedig az ***Inside-Outside algoritmus*** [Baker79] segítségével közelíthetjük. A modell mintáinak az összetétele is változtatható ha a komplex modellkészítési folyamatot paraméterezhetővé tesszük és az elemzési pontosságra maximalizáljuk. Egy erre alkalmazható optimalizáló algoritmus a ***szimulált hűtés (Simulated Annealing)*** [Aarts89]. Különféle osztályozó módszerek kombinációjával feljavíthatjuk az egyedi módszerek eredményeit. További javulás érhető el az eredményekben, ha a módszerekhez súlyokat rendelünk és ezeket a példák egy részén a kiértékelés alapján optimalizáljuk.

A komplex szintaktikai elemző módszer alkalmazása magyar nyelvre

A szerző az automatikus szintaktikai elemzés megvalósítására kidolgozott egy komplex szintaktikai elemző módszert, mely a feladat részproblémáit összefoglalta egy összefüggő, paraméterezhető rendszerbe. A modell építése a kiinduló faminta halmaz korpuszból való kigyűjtésével indul, mely faalak típusok felhasználásával történik. A kiinduló példákból az RGLearn algoritmus segítségével történik a faminták tanulása. Az így kapott faminta halmazt a ***chart parser*** ([Kaplan73], [Kay86]) módosított változatával alkalmazzuk a szintaktikai elemzés során. A kiértékelési módszer bevezetése lehetőséget biztosít a szerzett tapasztalatok visszacsatolására. A szimulált hűtés algoritmusának egy módosított változatával úgy optimalizáljuk a modellkészítés paramétereit, hogy a felismerési pontosság maximális legyen.

A szerző felkészítette és kiértékelte a szintaktikai elemzők többféle változatát magyar nyelvű szövegeken a Szeged Treebank [Csendes05] adatait felhasználva, valamint alkalmazta különféle természetes nyelvvel kapcsolatos feladatokra készült összetett rendszerekben. A Szeged Treebank a különböző munkafázisokban készült el és az adott állapot információtartalma meghatározta az ez alapján készült szintaktikai elemző felhasználási lehetőségeit.

A ***felszíni szintaktikai elemzés (Shallow Parsing)*** során nem törekszünk arra, hogy feltárjuk a teljes szintaxist és ez olyan egyszerűsítésekre ad lehetőséget mely által az elemzési fázis felgyorsítható és a felismerés pontossága is javítható. Ilyen leegyszerűsített feladat a ***legbelső/legkülső főnévi csoportok*** (***base-NP/top-NP***

chunking) határainak meghatározása. A szerző által megvalósított felszíni elemző [Hócza04a] általános és üzleti szövegeken volt felkészítve és kiértékelve.

Számos olyan alkalmazásról van, ahol elegendő a szövegek felszíni szintaktikai elemzése. Ilyen például az ***automatikus információkinyerés (Information Extraction)*** vagy a ***szöveg kivonatolás (Text Summarisation)*** is. A szerző és társai által készített információ kinyerő rendszer [Hócza03b] a szövegek feldolgozásának különféle fázisait megvalósító moduljait láncszerűen összekapcsolva (***toolchain***) működik. A rendszer bemeneteként kapott egyszerű szövegfájlból az egymásra épülő részelemzések automatikusan végrehajtásával előállítja a kinyert információkat tartalmazó strukturált adatbázis, eközben a rendszernek a következő részfeladatokat kell megoldania: mondat- és szószegmentálás, nyílt tokenosztályok és tulajdonnevek felismerése, morfológiai elemzés, szófaji egyértelműsítés, felszíni szintaktikai elemzés, szemantikus keretek illesztése és a felismert információk átírása strukturált adatbázisba. A rendszert üzleti híreket tartalmazó szövegekre alkalmaztunk.

A teljes szintaktikai elemzés több szempontból nehezebb probléma mint a felszíni elemzés, mivel sokféle szócsoport van és ezek mélyebb, összetettebb szerkezeteket alkotnak, emiatt a tanulás több mintát állít, valamint (a felszíni elemzéssel ellentétben) teljes szintaxisfát kell építeni. De a legnagyobb problémát magyar nyelv esetén az igei vonzatkeret modellezése jelenti, mivel a vonzatkeret elemek szabadok átrendezhetőek és nem feltétlenül összefüggősége mondatrészt alkotnak, emiatt ezt a jelenséget generatív jelegű szabályokkal nem lehet ábrázolni.

A szerző az általa kifejlesztett famintákon alapuló teljes szintaktikai elemzőjét a Szeged Treebank 2.0 adataiból vett általános szövegek és üzleti híreken készítette fel és értékelte ki [Hócza06a]. A szerző és társai a megtanult faminták felismerési pontosságának javítására alkalmazták a Boosting algoritmust [Hócza05a]. A [Hócza05b] cikkben a szerző és társai beszámolnak arról, hogy kialakítottak a Szeged Treebank 2.0 állományaiból egy mintaadatbázist és javasolták, hogy az eddig elkészült és az ezután kifejlesztett magyar nyelvű szintaktikai elemzők a pontos összehasonlíthatóság érdekében ezen legyenek felkészítve és kiértékelve.

A ***gépi fordítás (Machine Translation)*** feladata egy adott természetes nyelven elkészült szöveg automatikus átfordítása egy másik természetes nyelvre. Manapság a legjobb megoldást a ***statisztikai gépi fordító (Statistical Machine Translation)*** rendszerek adják. A szerző megvalósította egy ilyen rendszer, GenPar kiegészítését úgy, hogy beépítette magyar-angol nyelvpárt [Hócza06b], azaz egy inputként beadott magyar szövegnek a rendszer outputjaként megkapjuk az angol nyelvű fordítását. A rendszer tulajdonságainak feltérképezése céljából több prototípus is készült. A rendszerben szereplő magyar szövegek elemzéséért felelős modulok (szófaji egyértelműsítő és teljes

szintaktikai elemző) a Szeged Treebank annotált szövegein voltak felkészítve, az angol nyelvért felelős rész pedig a rendszerrel adott mintaprototípusok részeként adottak voltak. A GenPar betanításához és a kiértékeléshez szükség volt még párhuzamos mondatokra, azaz magyar nyelvű mondatokhoz rendelt angol fordításra. Ezeket a mondatpárokat a Hunglish Corpus [Varga05] adattárából választottuk ki, 5 ezer tanító és 500 teszt mondatpárt.

A disszertáció tézisei

A szerző értekezésben beszámolt az elmúlt években elért tudományos eredményeiről. Ezek két csoportba oszthatók, egyrészt beszélhetünk elméleti konstrukciókról és gyakorlati alkalmazásokról. Az első csoportba sorolhatóak a következő elméleti eredmények:

I/1. 2.7.2. fejezet: A szerző kidolgozott egy új formalizmust, melyet famintáknak nevezett el [Hócza04a]. A faminták mondatokon belül nagyobb, több szintű szintaktikai egységeket különítenek el, ugyanakkor hasonló szerkezetek összevonására is lehetőséget biztosítanak, így hatékony eszközt adnak az olyan ragozó és szabad szórendű nyelvek elemzéséhez, mint például a magyar nyelv.

I/2. 3.2.3. fejezet: A szerző kifejlesztett egy általános mintatanuló algoritmust, mely az RGLearn nevet kapta [Hócza04a]. Az algoritmus megkeresi a minták általánosítása és specializálása közötti optimális arányt, így famintákra alkalmazva azt a faminta halmazt, amely a maximális pontosságú szintaktikai elemzést adja.

I/3. 2.4.4. fejezet: A szerző elkészítette a chart parser szintaktikai elemző algoritmus famintákra alkalmazható változatát, mellyel bottom-up elemzés végezhető [Hócza04a].

I/4. 4. fejezet: A szerző egy komplex faminta alapú szintaktikai elemző módszerbe foglalta össze az egyedi lépéseket: kiinduló mintahalmaz gyűjtése, tanulás, elemzés, kiértékelés, modell optimalizálás [Hócza04a].

A gyakorlati alkalmazások az alábbi pontokba foglalhatók össze:

II/1. 3.2.4. fejezet: A szerző elkészített egy szövegkörnyezeti mintákon alapuló szófaji egyértelműsítőt, melynek alkalmazható mintáit az RGLearn algoritmussal állította elő. A módszer összehasonlításra került a szerző társai által kidolgozott módszerekkel [Kuba04].

II/2. 5.3.1. fejezet: A szerző alkalmazta a komplex faminta alapú módszert magyar nyelvű szövegek főnévi csoportjainak tanulására és felismerésére

[Hócza04a]. Az szerző által elért eredmények jelentős javulást mutattak a magyar nyelvre ezt megelőzően közölt eredményekhez viszonyítva.

II/3. 5.3.2. fejezet: A főnévi csoportokra alkalmazott felszíni elemzés beépítésre került a szerző és társai által készített információkinyerő rendszerbe [Hócza03b], mely magyar nyelvű gazdasági rövidhíreken volt felkészítve és kiértékelve.

II/4. 5.4.2. fejezet: A komplex faminta módszert a szerző alkalmazta magyar nyelvű szövegek teljes szintaktikai elemzésére [Hócza05b], [Hócza06a].

II/5. 5.4.3. fejezet: A szerző és társai a teljes szintaktikai elemzés faminta tanuló modelljét a Boosting algoritmussal optimalizálták [Hócza05a].

II/6. 5.4.5. fejezet: A szerző a teljes szintaktikai elemzőt beépítette a GenPar gépi fordító rendszerbe és létrehozott egy új, magyar-angol fordításra alkalmas kiegészítést [Hócza06b].

Summary in English

This dissertation concentrates on syntactic parsing of Hungarian texts. The author applied rule-based machine learning methods using an annotated corpus to build models for syntactic parsers. The rule-based approach stores the collected information in readable format for human readers, and allow experts to extend the syntactic database with their knowledge.

Syntactic parsing of natural languages

Describing a phenomenon of natural languages is a great challenge for computational linguistics, especially the Hungarian language, that is customarily defined as an agglutinative, free word order language with a rich morphology. These properties make a full analysis of it difficult, compared to Indo-European languages. The generative grammars seemed promising possibilities at the time of their introduction, because these grammars can be parsed with effective algorithms, especially in case of applying regular and context free grammars. However counterexamples appeared soon showing that this grammar classes are not suitable to describe certain phenomenon of natural languages. Such counterexample the self-embed structures, that can not be described by regular grammars, and the description of cross-dependencies can not be solved by context free grammars.

Today the generative approach have been changed such a linguistic theories and formalisms which concentrate on more precise description of a natural language phenomenon instead of generating languages. If we apply context free grammars to handle agreement and subcategorization we have to insert a lot of new rule and at the end of this process we get a very big grammar. Similar problem the handling of free word order. Describing dependency, especially far dependency, which can not be solved by context free grammars, because the rules of these grammars can cover only neighboring words. If we want to choose the best of ambiguous syntactic structures we have to extend rules with probability the values and estimate probability of syntax trees. Ignoring lexical and structural dependencies during the automatic parsing can results such an unused or unlikely structures which we can exclude according to sense of text.

With tree patterns introduced by the author we can recognize complex syntactic structures with help of description given by regular expressions. If we apply complex structures in syntactic parsing, this model will expectedly contains more items (rules) than a context free which grammar produced with the same corpus. The tree pattern description compensates this growing with containing similar structures in one pattern.

The flexibility of description allow building in elements of other formalisms, so various phenomenon of natural languages can be handled. We apply a modified version of ***chart parser*** ([Kaplan73], [Kay86]) to syntactic parsing of text with tree patterns.

Applying machine learning to create grammar

An efficient solution for natural language problems might be the application of machine learning methods, but it requires a large number of training and test examples of annotated phrases. If we have an annotated corpus we can collect the examples of a certain phenomenon. The set of examples contain (x_i, y_i) pairs, where x_i is the description of an object or event and y_i is the category or decision. In a ***classification*** problems the examples have discrete y_i values. It is a ***supervised training*** if the y_i values are known for each x_i (e.g. it can be extracted from an annotated corpus), and the task of machine learning to seek a suitable f function, where $f(x_i) = y_i$. In this case we suppose that the f function will be suitable to determine y values for unseen cases, this principle is called ***inductive learning***. When the classification problem has two classes (true or false) we call it ***concept learning***, in this case we have positive or negative examples depending on the decision of examples true or false.

The author developed the ***RGLearn*** pattern learning algorithm. The input of algorithm is a set of positive and negative examples collected from annotated corpus depend on the example has proper or improper coverage. The output of algorithm is a set of generalized patterns which collective precision is maximized, consequently it covers the most positive and the least negative examples. This algorithm was applied to learn disambiguation model for a rule-based POS-tagger [Kuba04], and to learn syntactic tree patterns ([Hócza04a], [Hócza06a]).

There are other possibilities of applying machine learning methods in building models for syntactic parsing. The POS tagger used for disambiguation of Part-of-Speech can be applied for predicting boundaries of word groups as well. We can utilize a word group boundary prediction method in various ***shallow parsing*** tasks, for example segmentation of sentence to smaller parts, or recognizing basic phrases (base-NP, top-NP). In order to create a probability model we can assign probabilities to patterns. If we have annotated corpus, we can estimate a pattern probability from its normalized occurrence. I was proved [Prescher03], that this way gives the ***maximum likelihood*** estimation of the given annotated corpus. Without annotated corpus we can use the ***Inside-Outside algorithm*** [Baker79] for probability estimation.

The content of pattern set can be different if we introduce parameters for the complex process of model making. We can search the best parameter settings evaluating models in a smaller part of the annotated corpus as well. An applicable optimization

algorithm for this task the ***simulated annealing*** [Aarts89]. We can improve our results with system combination of different methods. If we use weights for these methods we can also optimize setting on the annotated corpus.

Applying a complex method for syntactic parsing of Hungarian texts

The author developed an automatic syntactic parsing method collecting the solutions of problem parts in a complex parameter-driven system. The model building start with collecting syntactic structures examples from the annotated corpus with help of tree shape types. Typical tree shape types are the ***self-embed*** and ***string*** structures that give constraints for properties of syntactic structures collected from annotated corpus. Since we can collect a huge set of tree parts from entire corpus, it may cause a serious technical problem processing these examples together, therefore the examples are grouped together according to their most general forms. The tree pattern learning is performed with the RGLearn algorithm group by group.

The learned set of tree patterns is used by a modified version of ***chart parser*** ([Kaplan73], [Kay86]). This mean only some small changes in the original algorithm. The application of an evaluation method allow us to feedback our experiences and improving results. In order to make optimization we added parameters to our complex tree pattern learning methods. We made a frame procedure from the simulated annealing algorithm and we optimize the parameters of model generating on precision of parse trees.

The author prepared and evaluated various type of syntactic parsers on Hungarian texts using annotated texts of Szeged Treebank [Csendes05], and applied his syntactic parsers in various natural language tasks. The first version of Szeged Treebank allowed us to build models for NP recognition parsers. NP recognition is the process of determining whether sequences of words can be grouped together with nouns, and as a part of the field of *Shallow Parsing* is rich enough to support a number of large-scale natural language processing applications including *Information Extraction*, *Information Retrieval*, *Text Summarisation*, and a variety of text-mining operations. The author developed a shallow parser [Hócza04a] and it was applied and evaluated on general texts and short business news.

The author apply his shallow parsing method in an *Information Extraction* (IE) system [Hócza03b]. The IE system that was made by the author and his colleagues connects their results of various NLP tasks that had been developed as a toolchain. The input of this IE system is a plane text and the output is a structured database containing the extracted information. During the IE process the syntactic and semantic features of a sentence are determined with a pipeline of NLP modules, and this consist of

tokenization, sentence segmentation, morpho-syntactic analysis, part-of-speech tagging, shallow syntactic parsing, recognizing semantic frames, storing extracted information in a structured database. The IE process was applied on short business news taken from Szeged Treebank.

In many aspects the full syntactic parsing is a harder task than shallow parsing. There are more syntactic labels and syntactic structures are more deeper and complex than structures of shallow parsing. There are additional problems in full syntactic parsing like the VP of Hungarian language. Due to free word order the components of a verb group can be rearranged to a lot of order, and in addition the sentence part of a verb group is not always continuous. Therefore it is not possible to describe this phenomenon of Hungarian language with context free rules.

The author developed a full tree pattern based syntactic parser [Hócza06a] evaluated his method on general texts and short business news taken from Szeged Treebank 2.0. The author and his colleagues effectively improve the recognition accuracy of the syntactic parser using the Boosting algorithm [Hócza05a]. In [Hócza05b] author and his colleagues reported their effort in making a database from Szeged Treebank 2.0 to evaluate syntactic parsers, and they proposed using this database to compare Hungarian syntactic parsers that had been made so far and new parsers in the future.

Machine Translation (MT) is the application of computers to the translation of texts from one natural language to another. Today's state of the art in MT has been defined by *Statistical Machine Translation* (SMT) systems. The author extended an existing syntax-driven SMT system, the GenPar, building in the Hungarian-English as a new machine translation language pair [Hócza06b]. In order to examine effects of various preprocessing steps (POS-tagging, lemmatization and syntactic parsing) on system performance more prototypes had been made. The manually POS-tagged and syntactically parsed Hungarian and English texts needed for preprocessing was derived from the Szeged Treebank 2.0. The author used his tree pattern based method to parse Hungarian sentences, and the preprocessor modules for English texts was given in the GenPar original prototypes. Parallel sentences were selected from the Hunglish Corpus [Varga05]. The evaluation was performed on 5k training and 500 test sentence pairs selected from the Hunglish Corpus.

Results

The present thesis summarizes the results obtained by the author in the past couple of years. The results can be separated into two different groups, we can read about theoretical constructions and practical applications. The theoretical results are the following:

I/1. The author developed a new formalism named tree patterns [Hócza04a], that identify larger syntactic structures inside a sentence. With a single tree pattern we can describe several similar structures as a variation of a syntactic object. This formalism give us a powerful tool to parsing inflective and free word order languages like Hungarian.

I/2. The author implemented the RGLearn general pattern learning algorithm [Hócza04a]. The algorithm searches the optimal pattern between generalized and specialized form, for example in case of tree patterns the algorithm looks for the set of tree patterns that achieve the best result evaluating it on a test corpus.

I/3. The author implemented a tree pattern based chart parser that can be applied with bottom-up building strategy [Hócza04a].

I/4. The author worked out the complex method of tree pattern based syntactic parsing building together the individual modules: extracting syntactic structures from annotated corpus, learning tree patterns, parsing with tree patterns, evaluating results of the parser, feedback the information of performance to optimize the model [Hócza04a].

The practical applications from the second group of results, these are:

II/1. The author implemented a rule-based POS-tagger applying the RGLearn algorithm to learn context sensitive patterns for disambiguation. This method was compared with other methods made by colleagues of the author [Kuba04].

II/2. The author applied his complex tree pattern based method to learn and recognize noun phrases of Hungarian texts significantly outperforming previous results [Hócza04a].

II/3. The shallow parser for noun phrase recognition was built in an information extraction toolchain made by the author and his colleagues [Hócza03b]. This IE system was applied on Hungarian short business news.

II/4. The complex tree pattern based method was applied on full syntactic parsing of Hungarian texts [Hócza05b], [Hócza06a].

II/5. The results of full syntactic parsing method was improved by the author and his colleagues using the Boosting algorithm [Hócza05a].

II/6. The full syntactic parser was built in a statistical machine translation system named GenPar as a part of a new Hungarian-English extension [Hócza06b].

Irodalomjegyzék

[Aarts89] E. H. L. Aarts, E., Korst, J. (1989): Simulated Annealing and Boltzmann Machines, *John Wiley & Sons*, New York

[Abney91] Abney S. (1991): Parsing by chunks, in Principle-Based Parsing, *Kluwer Academic Publishers*.

[Abney96] Abney S. (1996): Partial Parsing via Finite-State Cascades, in *Proceedings of ESSLLI'96 Robust Parsing Workshop*, pp. 1-8.

[Alberti02] Alberti Gábor, Medve Anna (2002): Generatív grammatikai gyakorlókönyv I-II., *Janus/Books*, Budapest.

[Argamon98] Argamon, S., Dagan, I., and Krymolowski, Y. (1998): A memory-based approach to learning shallow natural language patterns, in *Proceedings of 36th Annual Meeting of the Association for Computational Linguistics (ACL)*, Montreal, pp. 67-73.

[Alexin03] Alexin, Z., Csirik, J., Gyimóthy, T., Bibok, K., Hatvani, Cs., Prószéky, G., Tihanyi, L. (2003): Manually Annotated Hungarian Corpus, in *Proceedings of the Research Note Sessions of the 10th Conference of the European Chapter of the Association for Computational Linguistics EACL03*, Budapest, Hungary, pp. 53–56.

[Baker79] Baker, James K. (1979): Trainable grammars for speech recognition, in *Proceedings of the Spring Conference of the Acoustical Society of America*, pp. 547–550.

[Barta05] Barta, Cs., Csendes, D., Csirik, J., Hócza, A., Kocsor, A., Kovács, K. (2005): Learning Syntactic Tree Patterns From A Balanced Hungarian Natural Language Database, The Szeged Treebank, in *Proceedings of the IEEE International Conference on Natural Language Processing and Knowledge Engineering (IEEE NLP-KE'05)*, Wuhan, China

[Baum72] Baum, L. E. (1972): An inequality and associated maximization techniquein statistical estimation of probabilistic functions of a markov process, *Inequalities*, (3).

[Bishop95] Bishop, C.M. (1995): Neural Networks for Pattern Recognition, *Oxford University Press*.

[Booth69] Booth, T. (1969): Probabilistic representation of formal languages. In *Tenth Annual IEEE Symposium on Switching and Automata Theory*, pp. 74-81

[Brants00] Brants, T. (2000): TnT -- a statistical part-of-speech tagger. In: *Proceedings of the Sixth Applied Natural Language Processing, ANLP-2000*, Seattle, WA

[Brill95] Brill, E. (1995): Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics* 21, pp. 543-565

[Black91] E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini and T. Strzalkowski (1991): A procedure for quantitatively comparing the syntactic coverage of English grammars, in *Proceedings of the DARPA Speech and Natural Language Workshop*, pp. 306-311.

[Burbank05] Burbank, A., Carpuat, M., Clark, S., Dreyer, M., Fox, P., Groves, D., Hall, K., Hearne, M., Melamed, I. D., Shen, Y., Way, A., Wellington, B., Wu, D., (2005): Final Report of the 2005 Language Engineering Workshop on Statistical Machine Translation by Parsing

[Charniak93] Charniak, E (1993): Statistical Language Learning, *MIT Press*, Cambridge, Massachusetts

[Charniak97] Eugene Charniak. (1997): Statistical parsing with a context-free grammar and word statistics: in *Proceedings of the 14th National Conference on Artificial Intelligence*, Providence, RI. AAAI Press/MIT Press.

[Chomsky57] Chomsky, Noam (1957): Syntactic Structures, *Mouton, The Hague*. (Magyarul: Chomsky, Noam (1995): Mondattani szerkezetek, *Nyelv és elme. Osiris – Századvég*, Budapest)

[Csendes05] Csendes, D., Csirik, J., Gyimóthy, T., Kocsor, A. (2005): The Szeged Treebank, in *Proceedings of the 8th International Conference on Text, Speech and Dialogue, TSD 2005*, Karlovy Vary, pp. 123-131

[Dempster77] Dempster, A. P., N. M. Laird, and D. B. Rubin. (1977): Maximum likelihood from incomplete data via the EM algorithm, in *Journal of the RoyalStatistical Society*, Series B, 39:1–38.

[Earley70] Jay Earley (1970): An efficient context-free parsing algorithm, in *Communications of the ACM*, 6:451–55

[Erjavec97] Erjavec, T. and Monachini, M., ed. (1997): Specification and Notation for Lexicon Encoding, *Copernicus project 106 "MULTEXT-EAST"*, Work Package WP1 – Task 1.1 Deliverable D1.1F.

[Fellbaum98] Fellbaum, C. (ed.). (1998): WordNet: an electronic lexical database, *MIT Press*, Cambridge, MA

[Fodor80] Fodor, Janet D. and Frazier, Lyn. (1980): Is the human sentence processing mechanism an ATN?, in *Cognition*, 8, 417-459.

[Hobbs96] Jerry Hobbs, Douglas Appelt, John Bear, David Israel, Megumi Kameyama, Mark Stickel, and Mabry Tyson (1996): FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural Language Text, in *Emmanuel Roche and Yves*

[Hócza02] Hócza, A., Szilágyi, Gy., Gyimóthy, T. (2002): LL Frame System of Learning Methods, in *Proceedings of the CS2 2002*, Szeged, Hungary pp. 46.

[Hócza03a] Hócza, A., Iván, Sz. (2003): Főnévi csoportok tanulása és felismerése, *MSZNY 2003 konferencia kiadványa*, Szeged, 72-78 oldal

[Hócza03b] Hócza, A., Alexin, Z., Csendes, D., Csirik, J., Gyimóthy, T. (2003): Application of ILP methods in different natural language processing phases for information extraction from Hungarian texts, in *Proceedings of the Kalmár Workshop on Logic and Computer Science*, Szeged, pp. 107-116.

[Hócza04a] Hócza, A. (2004): Noun Phrase Recognition with Tree Patterns, in *Acta Cybernetica*, Szeged, Volume 16, Issue 4, pp. 611-623

[Hócza04b] Hócza, A. (2004): Shallow Parsing for Information Extraction, in *Proceedings of the CS2 2004*, Szeged, pp. 54.

[Hócza04c] Hócza, A. (2004): Teljes mondatszintaxis tanulása és felismerése, *MSZNY 2004 konferencia kiadványa*, Szeged, 127-135 oldal

[Hócza05a] Hócza, A., Felföldi, L., Kocsor, A. (2005): Learning Syntactic Patterns Using Boosting and Other Classifier Combination Schemas, in *V. Matousek et al. (Eds.): Proceedings of the 8th International Conference on Text, Speech and Dialogue, TSD 2005*, Karlovy Vary, Czech Republic, LNAI 3658, pp. 69-76

[Hócza05b] Hócza, A., Kovács, K., Kocsor, A. (2005): Szintaktikai elemzők eredményeinek összehasonlítása, *MSZNY 2005 konferenciakiadványa*, Szeged, 277-284 oldal

[Hócza06a] Hócza, A. (2006): Learning Tree Patterns for Syntactic Parsing, in *Acta Cybernetica*, Szeged, Volume 17, Issue 3, pp. 647 - 659

[Hócza06b] Hócza, A., Kocsor, A. (2006): Hungarian-English machine translation using GenPar, in *Proceedings of the 9th International Conference on Text, Speech and Dialogue, TSD 2006*, Brno, Czech Republic, September 11-15, pp. 87-94

[Jain00] Jain, A. K. (2000): Statistical Pattern Recognition: A Review, in *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 22. No. 1

[Joshi85] Aravind K. Joshi (1985): Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In: *D. R. Dowty, L. Karttunen and A. M. Zwicky (eds.), Natural Language Parsing. Psychological, Computational, and Theoretical Perspectives*, pp. 206-250. Cambridge University Press, New York, NY.

[Joshi92] Aravind K. Joshi and Yves Schabes (1992): Tree-adjoining grammars and lexicalized grammars, in *Maurice Nivat and Andreas Podelski, editors, Tree Automata and Languages*. Elsevier Science.

[Joshi91] Aravind K. Joshi, K. Vijay-Shanker and David J. Weir. (1991): The convergence of mildly context-sensitive grammar formalisms, in *Natural Language Processing*, Cambridge, MA: MIT Press, pp. 31–81

[Jurafsky00] Jurafsky, Daniel, Martin, James H. (2000): Speech and Language Processing: An Introduction to Natural Language Processing, *Speech Recognition, and Computational Linguistics*. Prentice-Hall.

[Kaplan73] R. M. Kaplan (1973). A general syntactic processor. In Rustin, R. (Ed.), *Natural Language Processing*, pp. 193-241. Algorithmics Press, New York.

[Kay86] Martin Kay. (1986). Algorithm schemata and data structures in syntactic processing. In Readings in natural language processing, pp. 35-70. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Kiefer92] Kiefer Ferenc (1992): Strukturális magyar nyelvtan, I. Mondattan, Akadémiai Kiadó, Budapest.

[Kiss99] É. Kiss Katalin, Kiefer Ferenc, Siptár Péter (1999): Új magyar nyelvtan, Osiris Kiadó, Budapest.

[Kis03] Kis Balázs; Naszódi Mátyás; Prószéky Gábor (2003): Komplex (magyar) szintaktikai elemző rendszer mint beágyazott rendszer, *I. Magyar Számítógépes Nyelvészeti Konferencia előadásai (MSZNY)*, 145-150. SZTE, Szeged

[Knuth65] D. E. Knuth (1965): On the translation of languages from left to right, *Information and Control*, 8:607-639, 12

[Kuba03] Kuba A., Bakota T., Hócza A., Oravecz Cs. (2003): A magyar nyelv néhány szófaji elemzőjének összevetése, *MSZNY 2003 konferencia kiadványa*, Szeged, 16-24 oldal

[Kuba04] Kuba, A., Hócza, A., and Csirik, J. (2004): POS Tagging of Hungarian with Combined Statistical and Rule-Based Methods, in *Proceedings of the 7th International Conference on Text, Speech and Dialogue TSD 2004*, Brno, Czech Republic, September 8-11, pp. 113-120

[Marcus93] Marcus, M., Santorini, B., Marcinkiewicz, M. (1993): Building a large annotated corpus of English: the Penn Treebank. in *Computational Linguistics*, vol. 19

[Mitchel97] Tom M. Mitchel (1997): Machine Learning, The McGrew-Hill Companies Co. ISBN: 0-07-115467-1.

[Papineni02] Papineni, K., Roukos, S., Ward, T., Zhu, W. J. (2002): BLEU: a method for automatic evaluation of machine translation, in *Proceedings of the 40th Annual Meeting of the ACL*, pp. 311-318.

[Prescher03] Prescher, D. (2003): A Tutorial on the Expectation-Maximization Algorithm Including Maximum-Likelihood Estimation and EM Training of Probabilistic Context-Free Grammars, *Presented at the 15th European Summer School in Logic, Language, and Information (ESSLLI 2003)*.

[Prószéky99] Prószéky, G., Kis, B. (1999): A unification-based approach to morpho-syntactic parsing of agglutinative and other (highly) inflectional languages. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, College Park, Maryland, USA, pp. 261-268.

[Pullum84] Pullum, Geoffrey K. (1984): On Two Recent Attempts to Show that English is Not a CFL, in *Computational Linguistics*. 10(3-4), pp. 182-186.

[Quinlan86] Quinlan, J. R. (1986): Induction of Decision Trees, *Machine Learning*, Vol 1 No 1 pp. 81-106.

[Quinlan93] Quinlan, J. R. (1993): C4.5: Programs for Machine Learning, *Morgan Kaufmann Publisher*.

[Rabiner89] Rabiner, L. R. (1989): A tutorial on hidden Markov models and selected applications in speech recognition, in *Proceedings of IEEE*, 77 (2), 257-286.

[Ramshaw95] Ramshaw, L. A., and Marcus, M. P. (1995): Text Chunking Using Transformational-Based Learning, in *Proceedings of the Third ACL Workshop on Very Large Corpora*, Association for Computational Linguistics.

[Ratnaparkhi97] Ratnaparkhi, A., (1997): A linear observed time statistical parser based on maximum entropy models, in *Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Providence, Rhode Island

[Shapire90] R.E. Shapire (1990): The Strength of Weak Learnability, in *Machine Learnings*, Vol. 5, pp. 197-227

[Shieber85] Shieber, Stuart M. (1985): Evidence against Context-Freeness of Natural Language, in *Linguistics & Philosophy*. 8(3), pp.333–343.

[Shieber86] Shieber, Stuart M. (1986): An Introduction to Unification-based Approaches to Grammar, *Lecture notes, CSLI*, Leland StanfordJunior University, Stanford, California.

[Sleator91] Sleator, D., Temperley, D. (1991): Parsing English with a Link Grammar, in *Carnegie Mellon University Computer Science technical report* CMU-CS-91-196.

[Simmons92] Robert F. Simmons, Yeong-Ho Yu (1992): *The acquisition and use of context dependent grammars for English*. Computational Linguistics, 18:391-418

[Simov01] Simov K. (2001): CLaRK – an XML-based System for Corpora Development, in *Proceedings of the Corpus Linguistics 2001 Conference*, Lancaster, pp. 553-560.

[Tjong99] Tjong Kim Sang, E. F., and Veenstra, J. (1999): Representing text chunks, in *Proceedings of EACL '99*, Association for Computational Linguistics.

[Tjong00] Tjong Kim Sang, E. F. (2000): Noun Phrase Recognition by System Combination, in *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, Seattle, pp. 50-55.

[Tomita91] Masaru Tomita (1991): Generalized LR Parsing, Kluwer Academic Publishers,

[Varga05] Varga, D., Németh, L., Halácsy, P., Kornai, A., Trón, V., Nagy, V. (2005): Parallel corpora for medium density languages, in *Proceedings of the Recent Advances in Natural Language Processing 2005 Conference*, pages 590-596.

[Váradi03] Váradi T. (2003): Shallow Parsing of Hungarian Business News, in *Proceedings of the Corpus Linguistics 2003 Conference*, Lancaster, pp. 845-851.

[Viterbi67] Viterbi, A. J. (1967): Error bounds for convolutional codes and an asymptotically optimal decoding algorithm, in *IEEE Transactions on Information Processing*, 13:260-269.

[Younger67] D. H. Younger (1967): Recognition of context-free languages in time n 3, in *Inform. Control*, vol. 10, no. 2, pp. 189-208

[Wirén89] Mats Wirén (1989): Interactive incremental chart parsing, in *Proceedings of the fourth conference on European chapter of the Association for Computational Linguistics*, Manchester, England, pp. 241-248