Tagmondatokra bontás és NP-chunking függőségi alapon

Dömötör Andrea[1,2,3], Nemeskey Dávid[1,2]

[1]ELTE BTK TI Digitális Bölcsészet Tanszék
1088 Budapest Múzeum krt. 6-8.
[2]Digitális Örökség Nemzeti Laboratórium
[3]PPKE BTK Nyelvtudományi Doktori Iskola
1088 Budapest, Mikszáth Kálmán tér 1.
{domotor.andrea,nemeskey.david}@btk.elte.hu

Kivonat Ebben a cikkben a tagmondatokat és a köztük lévő kapcsolat típusát a függőségi elemzés mintázataiból kíséreljük meg meghatározni. Mivel ennek a feladatnak a teszteléséhez még nincs gold sztenderd adatunk, a módszerünket kipróbáltuk egy másik feladaton, az NP-chunkingon is. Ez utóbbi kiértékelésénél nehézséget okozott, hogy az elvben gold sztenderd korpuszok több hibát is tartalmaztak, mind a függőségi elemzésben, mind az NP-chunkingban. Mindezekkel együtt 89%-os f-score-t értünk el, ami ugyan elmarad a state-ot-the-arttól, de abból a szempontból mégis ígéretes, hogy ezt az eredményt egy egyszerű szabályrendszerrel értük el. Ez alapján a függőségi elemzés mintaillesztése további kutatásra érdemes módszer lehet a hasonló feladatokban. **Kulcsszavak:** NP-chunking, függőségi elemzés, tagmondatok, mintaillesztés

1. Bevezetés

A legtöbb korpusz és nyelvfeldolgozó eszköz a szöveget mondatok, azon belül pedig tokenek összességeként kezeli. A mondat és a token szintje között lehetnek még többszavas kifejezések (chunkok), azonban a tagmondat mint nyelvi szint jellemzően nem jelenik meg a számítógépes szövegfeldolgozásban.

Az összetett mondatok tagmondatai teljes értékű mondatoknak tekinthetők, így elkülönítésük hasznos lehet olyan feladatok esetén, ahol fontosak a (tag)mondathatárok, ilyen lehet például az elváló igekötők és igéik összekapcsolása. Ezen kívül a tagmondatok közötti viszonyok meghatározóak a szövegértelmezés szempontjából: egy kötőszón múlhat, hogy egy szövegrész tartalma egy másikéból következik-e vagy fordítva.

Ebben a cikkben a tagmondatokat és a köztük lévő kapcsolat típusát a függőségi elemzés mintázataiból kíséreljük meg meghatározni. Mivel ennek a feladatnak a teszteléséhez még nincs gold sztenderd adatunk, a módszerünket kipróbáltuk egy másik feladaton, a főnévi csoportok felderítésén (*NP-chunkingon*) is. Bár itt természetesen két eltérő feladatról van szó, látni fogjuk, hogy mindkettő megoldható függőségi mintaillesztéssel.

Magasabb szintű nyelvi feladatok megoldása függőségi, vagy frázisstruktúra-nyelvtan segítségével természetesen nem új gondolat. Csak példaképp, Recski és mtsai (2009) a Szeged Treebank (Csendes és mtsai, 2005) elemzési fáiból nyerték ki az NP chunking korpuszt; Metheniti és mtsai (2019) pedig függőségi elemzés alapján azonosítottak nyelvtani szabályokat. Tudomásunk szerint azonban az ebben a cikkben leírt feladatokat magyar nyelvre még nem próbálták korábban függőségi elemzés alapján megoldani.

2. Mintaillesztés függőségi fákon

Szabály alapú felismerési feladatokban — akár az összetett mondatokról, akár a (maximális) főnévi csoportok felismeréséről beszélünk — meghatározott nyelvtani konfigurációk meglétét ellenőrizzük a mondatokban. A függőségi nyelvtanok esetén ez egyszerűen átfogalmazható gráfelméleti alapokra.

A függőségi elemzés egy csúcs- és élcímkézett fát ad eredményül, ahol minden csúcs egy tokennek, minden él pedig egy nyelvtani függőségnek felel meg. Hasonlóan, a keresett nyelvtani minták is egy fával írhatók le, hiszen általában mind a csúcsok adatai (szó, lemma, szófaj, stb.), mind a közöttük lévő kapcsolatok fontosak (példa összetett mondatok esetén: két ige, amik egy kötőszón keresztül kapcsolódnak). A feladat ekkor a mintafával izomorf részgráf keresése az elemzési fában.

A mintakeresés fa gráfokban jól kutatott terület, nyelvi vonatkozásban is (Kilpeläinen és mtsai, 1992). Több rendszert is javasoltak függőségi fák mintaillesztéséhez (Wallis, 2008; Nichols és Reisert, 2014; Luotolahti és mtsai, 2015), azonban ezek többsége nem elérhető. Az utolsó az egyedüli, amelyik fellelhető, de csak offline keresésre használható, és nem specifikálja, hogy a minta hol fordul elő a mondatban. Ezért egy saját mintaillesztő írása mellett döntöttünk.

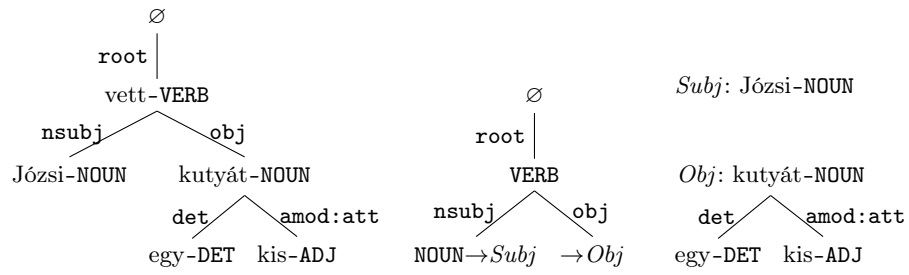
Az illesztés alapvetően a Kilpeläinen és mtsai (1992)-ben leírt *unordered path inclusion* (rendezetlen út tartalmazás) algoritmust követi. Két pontban bővítettük ki: egyrészt a csomópontok (esetünkben tokenek) több tulajdonságára (lemma, szófaj) is lehet keresni regex alapon, másrészt a minta élein megadott függőségi kapcsolatokkal is illesztünk. Az illesztés lehet pozitív, illetve negatív is; negatív feltétel teljesülése megakadályozza az illesztést.

Ha egy minta előfordul a mondatban, a rendszer kivágja az annak megfelelő részt a mondatfából. A minta csomópontjait meg lehet jelölni változókkal, amik megjelennek az eredményfában is. A jelölt szavak részfái, illetve a hozzájuk tartozó mondatrészek ezután egyszerűen lekérhetők.

Egy mintaillesztés példa látható az 1. ábrán. 1a. a *„Józsi vett egy kis kutyát”* mondat függőségi elemzése[1], 1b. pedig az illeszteni kívánt minta, ami igei (`VERB`) állítmánnyal (`root`) rendelkező mondatokban keresi a főnévi (`NOUN`) alanyt és a (bármilyen) tárgyat. Az előbbit *Subj*, az utóbbit *Obj* változóval jelöltük. 1c. mutatja a rendszer kimenetét: az egyes változókhoz tartozó részfákat.

Érdemes megemlíteni, hogy bár az algoritmus bármilyen függőségi címkekészlettel működik, a leírt rendszer a mára már sztenderdnek számító Universal

[1] A szemléltetés kedvéért a megszokottól eltérően „álló” fával.



(a) A mondat (b) A minta (c) Az eredmény

1. ábra. Példa mintaillesztésre

Dependencies (UD)[2] címkéit használja – ahogy a legtöbb magyar nyelvre elérhető szintaktikai elemző eszköz (Stanza, UDPipe, HuSpaCy) is. Hasonlóan, a módszer bizonyára adaptálható lehetne frázisstruktúra-nyelvtanok elemzési fáira is; ezek szerepét azonban a természetesnyelv-feldolgozásban szinte teljesen átvette a függőségi elemzés. A cikkben ezért mi is ez utóbbira koncentrálunk.

3. Tagmondatokra bontás

3.1. Kapcsolódó irodalom

A tagmondatkapcsolatok automatikus felismerésére Szemes (2021) tett kísérletet a közelmúltban. Módszere a mondatközi írásjelek, valamint a kötőszavak és vonatkozó névmások mintázatain alapul. Eszerint a tagmondathatárok egy írásjel és egy kötőszó vagy vonatkozó névmás szekvenciájával azonosíthatók, a tagmondatok közötti kapcsolat típusát pedig a kötőszók és vonatkozó névmások (kézi) klasszifikációjával határozta meg a szerző. A tanulmány elsősorban a mondattípusok gyakoriságának stilisztikai és irodalomtörténeti vonatkozásaira fókuszál, így magának a tagmondatkapcsolat-meghatározó algoritmusnak a kiértékelésére nem tér ki. Megemlíti azonban a módszer azon hiányosságát, miszerint az csak azokra az összetett mondatokra alkalmazható, ahol van mondatközi írásjel és kötőszó (vagy vonatkozó névmás) is. Ez azonban nem minden esetben igaz, a *hogy* kötőszó például gyakran elhagyható a tagmondathatárokról, és bár a helyesírás valóban megköveteli a tagmondatok közötti vesszőt, ezt a szabályt a szövegeket létrehozó nyelvhasználók gyakran nem tartják be. Másrészről a tagmondathatár-keresés azért sem mindig elégséges módja a tagmondatok meghatározásának, mert a tagmondatok nem feltétlenül folytonosak, lehetnek bennük megszakítások, közbeékelések. Ugyanakkor a kötőszók lexikális vizsgálata a mi módszerünknél jóval részletesebb tagmondatkapcsolat-kategorizálást tesz lehetővé.

[2] `https://universaldependencies.org/`

A tagmondatok meghatározására függőségi elemzésre alapozó módszert használ Gyulai (2021), azonban a miénkkel ellentétben bottom-up megközelítéssel. Ennek lényege, hogy olyan összefüggő kifejezéseket keres, aminek az első és utolsó eleméből rekurzívan el lehet jutni a tagmondat fejéig, egy V POS taggel ellátott összetevőig vagy kopuláig. Ez a módszer jól alkalmazható a Szeged Treebanken, ahol valóban minden (tag)mondat feje igei elem, hiszen az elliptált igék és zéró kopulák is megjelennek a függőségi elemzésben üres igei csomópontokként. Más szövegekben azonban nem feltétlenül tartalmaz minden tagmondat igét. Ezen kívül, bár a módszernek kétségtelenül előnye, hogy nem függ a függőségi elemzés címkekészletétől, az elemzési sémától azonban így is függ: például a UD elvei szerint a kopulás mondatok esetén nem a kopula a mondat feje, hanem a predikatív névszó, így ezeknél a mondatoknál ezzel a módszerrel nem jutunk el a kopuláig. Végül, Gyulai (2021) módszere csak a tagmondatokat állapítja meg, a miénk azonban kísérletet tesz a tagmondatok közötti kapcsolatok típusának meghatározására is.

3.2. Tagmondatkapcsolat-típusok

A Magyar grammatika (Lengyel, 2000) alapján megkülönböztetünk alárendelő és mellérendelő összetett mondatokat. Utóbbi alcsoportjai (kapcsolatos, ellentétes, megengedő, magayrázó, következtető) között leginkább csak a kötőszó alapján lehet különbséget tenni, szerkezeti eltérés nincs közöttük, ezért a mellérendelő összetételek altípusaival nem foglalkoztunk. Az alárendelő monatoknál a mellékmondat által kifejtett mondatrész szintaktikai szerepe alapján határoztuk meg az altípusokat (akárcsak a Magyar grammatika). Külön csoportba soroltuk a vonatkozó mellékmondatokat, ahol a kötőszó szerepét egy vonatkozó névmás tölti be.

A tagmondatkapcsolatok típusait az 1. táblázat foglalja össze. A példák forrása a Szeged Treebank.

Főtípus	**Altípus**	**Példa**
Mellérendelő		*Már 5.05 volt, azonban a Zoli bácsi és a felesége még seholsem volt.*
Alárendelő	Alanyi	*A könyvben részletesen le volt írva, hogy ez a szelidítés hogyan történt.*
	Tárgyi	*Mindenki nagyon izgatottan várta, hogy elinduljon a busz.*
	Határozós	*S úgy láttam, mintha kezdene megnyugodni.*
	Jelzős	*Akkora pizzát adtak, hogy el kellett hoznunk.*
	Vonatkozó	*Zoli bácsi hozott fényképeket, amelyeket a felesége készített.*

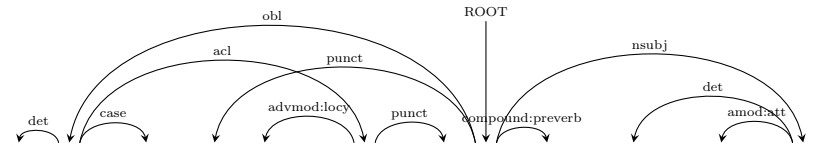
1. táblázat. Az összetett mondatok típusai

3.3. Tagmondatok meghatározása függőségi elemzés alapján

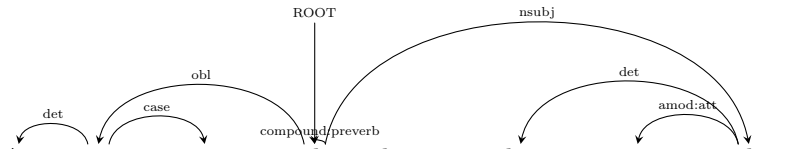
A tagmondatok meghatározásánál a függőségi elemzésből indultunk ki. A tagmondatokra úgy tekintünk, mint a teljes mondat függőségi elemzési fájának rész-

fái, így a tagmondatok meghatározása lényegében a részfák gyökércsomópontjainak meghatározását jelenti. Ez a megközelítés előnyösebb a szekvenciális (pl. IOB) címkézésnél, a tagmondatok ugyanis nem feltétlenül folytonosak. A gyökércsomópontok meghatározásával azonban a szétszakított tagmondatok részei is "egymásra találnak" a függőségi elemzésben (amennyiben persze az helyes). az 1–3. példák egy nem folytonos tagmondatot tartalmazó mondat függőségi elemzés szerinti felbontását mutatják. A mondat gyökere (ROOT) a főmondat gyökércsomópontja, a vonatkozó mellékmondat (3) gyökerét pedig az `acl` függőség alapján határozzuk meg. A gyökércsomópontokból kiindulva a függőségekből megkapjuk a tagmondatokat, azok sorrendjétől és elhelyezkedésétől (és általában a szórendtől) függetlenül, azaz a módszerünk nem érzékeny arra, hogy az *ahol sétáltunk* vonatkozó mellékmondat beékelődik-e a főmondatba, vagy követi azt.

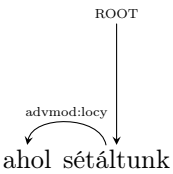
(1)



(2)

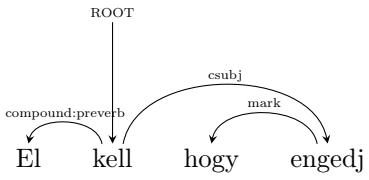


(3)

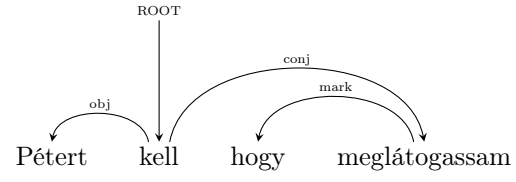


Problémásak azonban az úgynevezett mondatátszövődések (*El kell hogy engedj., Pétert kell hogy meglátogassam.*), ezeknél ugyanis a függőségi elemző számára nehéz az elváló igekötő vagy igei bővítmény fejének megtalálása. (4) és (5) példamondatokat Stanzával (Qi és mtsai, 2020) elemeztük, és mindkét esetben hibás elemzést kaptunk: (4) igekötője és (5) tárgya is a főmondathoz (*kell*) kapcsolódik a mellékmondatok megfelelői igéi helyett.

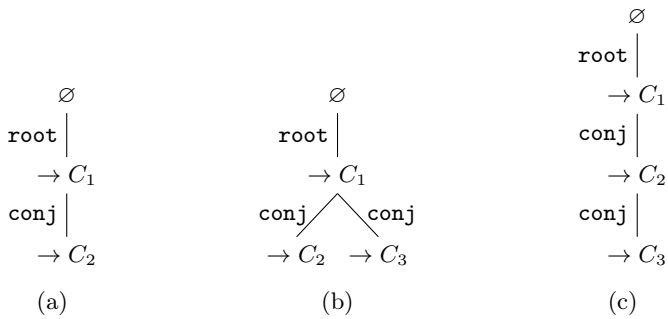
(4)



(5)



A tagmondatok gyökércsomópontjaira illeszkedő mintázatok meghatározásánál értelemszerűen az ideális esetből, azaz a helyes függőségi elemzésből indultunk ki. A mintázatok leírásához a Universal Dependencies címkekészletet használtuk, mert a legtöbb magyar nyelvre elérhető szintaktikai elemző eszköz (Stanza, UDPipe, HuSpaCy) ezt a sémát használja. A tagmondatokat meghatározó függőségi kapcsolatok a következők:

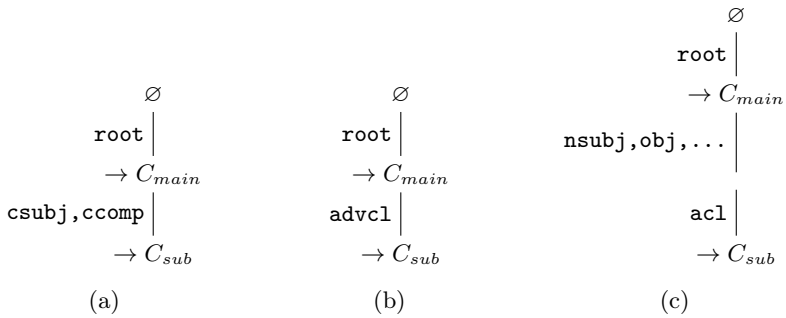


2. ábra. Minták mellérendelt tagmondatok azonosításához

– `root`: a főmondat, vagy mellérendelés esetén az első tagmondat gyökere
– `conj` (conjunct): mellérendelt tagmondat gyökere
– `csubj` (clausal subject): alanyi mellékmondat gyökere
– `ccomp:obj` (clausal complement: object): tárgyi mellékmondat gyökere
– `advcl` (adverbial close modifier): határozói mellékmondat gyökere
– `acl` (clausal modifier of noun)
  - jelzői mellékmondat, ha a fej egy másik tagmondat gyökércsomópontja
  - vonatkozó mellékmondat, ha a fej egy névszó

A vonatkozó mellékmondatok kivételével a tagmondatok gyökerének feje egy másik tagmondat (vagy a teljes mondat) gyökere. Ezért a függőségi elemzésben jellemzően láncszerű mintákat keresünk. A 2. ábrán néhány mellérendelő összetett mondat függőségi mintázatai láthatók. Többszörös összetételeknél a tagmondatok bizonyos esetekben láncszerűen kapcsolódnak egymáshoz (2c), más esetekben viszont egy tagmondat több mellérendelt tagmondat gyökereként van elemezve a függőségi elemzésben (2b).

Az alárendelő összetett mondatok esetén (3. ábra) a függőségi viszonyok tükrözik a tagmondatkapcsolatok struktúráját.



3. ábra. Minták alárendelt tagmondatok azonosításához

3.4. A tagmondatkapcsolatok jelölése CoNLL-ben

A tagmondatok közötti viszonyok nem reprezentálhatók fa struktúrában, egy tagmondat ugyanis több másik tagmondathoz is kapcsolódhat: lehet egyszerre egy tagmondat alárendeltje és egy másik mellérendeltje. Ezért a tagmondatkapcsolatok CoNLL-ben való megjelenítéséhez a CoNLL-U formátum továbbfejlesztett függőségi gráf (DEPS) mezőjéből indultunk ki. A továbbfejlesztett függőségi gráf lehetővé teszi a mellérendelések megjelenítését azáltal, hogy az elsődleges függőségi viszonyok mellett megadhatók további fej-reláció párok is (4. ábra).

1	They	they	PRON	PRP	Case=Nom|Number=Plur	2	nsubj	2:nsubj|4:nsubj
2	buy	buy	VERB	VBP	Number=Plur|Person=3|Tense=Pres	0	root	0:root
3	and	and	CONJ	CC	_	4	cc	4:cc
4	sell	sell	VERB	VBP	Number=Plur|Person=3|Tense=Pres	2	conj	0:root|2:conj
5	books	book	NOUN	NNS	Number=Plur	2	obj	2:obj|4:obj
6	.	.	PUNCT	.	_	2	punct	2:punct

4. ábra. Példa az UD továbbfejlesztett függőségi gráfjára

A tagmondatkapcsolatok jelöléséhez a CoNLL-U+ formátumot kiegészítettük egy `COMP:RELS` mezővel, ahol a tagmondatok gyökércsomópontjai kapnak

annotációt (a többi token esetén a mező _ jelet kap). A mondat gyökéreleme minden esetben `0:root` címkét kap, a belőle származó tagmondatot tekintjük a főmondatnak[3]. A többi tagmondat gyökerének annotációjában megadjuk annak a tagmondatnak a gyökerét (token id-ját), amellyel a tagmondat viszonyban áll, majd kettősponttal elválasztva meghatározzuk a tagmondatkapcsolat típusát. Ha egy tagmondat több másik tagmondattal is viszonyban áll, akkor mindegyiket jelöljük | jellel elválasztva.

Az egyes tagmondattípusok jelölései a következők:

- Mellérendelés: `coord`
- Alanyi alárendelés: `subj`
- Tárgyi alárendelés: `obj`
- Határozói alárendelés: `obl`
- Jelzői alárendelés: `att`
- Vonatkozó mellékmondat: **rel**

```
# global.columns = FORM WSAFTER FEATS UPOS XPOS LEMMA ID DEPREL HEAD COMP:RELS

Úgy	" "	PronType=Dem	ADV	_	úgy	1	advmod:mode	2	_	_
volt	" "	Definite=Ind|Mood=Ind|Number=Sing|Person=3|Tense=Past|VerbForm=Fin|Voice=Act	VERB	_	van	2	root	0	0:root
megbeszélve	" "	VerbForm=Conv	ADV	_	megbeszélve	3	advmod:mode	2	_
,	" "	_	PUNCT	_	,	4	punct	2	_
hogy	" "	_	SCONJ	_	hogy	5	mark	8	_
hajnali	" "	Case=Nom|Degree=Pos|Number=Sing|Number[psed]=None|Number[psor]=None|Person[psor]=None	ADJ	_	hajnali	6	amod:att	7	_
ötkor	" "	_	ADV	_	ötkor	7	advmod:tlocy	8	_
indulunk	" "	Definite=Ind|Mood=Ind|Number=Plur|Person=1|Tense=Pres|VerbForm=Fin|Voice=Act	VERB	_	indul	8	advcl	2	2:obl
és	" "	_	CCONJ	_	és	9	cc	10	_
találkozunk	" "	Definite=Ind|Mood=Ind|Number=Plur|Person=1|Tense=Pres|VerbForm=Fin|Voice=Act	VERB	_	találkozik	10	conj	8	8:coord|2:obl
a	" "	Definite=Def|PronType=Art	DET	_	a	11	det	12	_
Postánál	" "	Case=Ins|Number=Sing|Number[psed]=None|Number[psor]=None|Person[psor]=None	PROPN	_	Posta	12	nmod:obl	10	_
.	"\n"	_	PUNCT	_	.	13	punct	2	_
```

5. ábra. Példa egy többszörösen összetett mondat CoNLL-U+ annotációjára

Az 5. ábrán látható egy példamondat tagmondatkapcsolatokat is tartalmazó CoNLL-U+ annotációja. Jelenleg zajlik egy gold standard tesztkorpusz annotációja, amely alkalmas lesz majd a tagmondat-azonosító módszerünk kiértékelésére.

4. A módszer tesztelése egy másik feladaton: NP-chunking

4.1. NP-chunking

Az NP-chunking feladat célja a főnévi (névszói) csoportok (*noun phrase*-ek, *NP*-k) azonosítása a mondatban. A főnévi csoport egy olyan szerkezet (frázis), amely főnévi funkciót tölt be.

A főnévi csoportok „rekurzívak”, mivel tartalmazhatnak más főnévi csoportokat is: a „Piroska könyve” kifejezés egy NP, de a „Piroska” szó maga is az. Emiatt

[3] Bár nyelvészeti szempontból a főmondat fogalma nem értelmezhető mellérendelés esetén, a jelölésrendszerünkben akkor is meghatározunk egy "főmondatot", ha csak mellérendelő tagmondatkapcsolatok fordulnak elő a mondatban. Főmondat alatt itt azt a tagmondatot értjük, amelyikből a gráf kiindul. Ez a tagmondatkapcsolatok meghatározásának viszonyítási alapja.

olyan lineáris címkézést, ami az összes csoportot jelöli, nem lehet – de legalábbis nem praktikus – készíteni. Az NP chunking feladat során ezért tipikusan a két végletet címkézik (Ramshaw és Marcus (1995) base NP-definíciója alapján):

- a *minimális NP* nem tartalmaz más NP-ket;
- a *maximális NP* épp ellenkezőleg, nem része magasabb szintű főnévi csoportnak.

A főnévi csoport a frázisstruktúra-nyelvtan eleme, és nem képezhető le triviálisan függőségi nyelvtani terminológiára. Az NP-k jól megfoghatók bottom-up konstrukciós szabályokkal (Recski, 2014), a minimális NP-k pedig akár véges nyelvtannal is leírhatók. Azonban mindkét konstrukció jelentős mértékben támaszkodik a szavak sorrendiségére, ami a függőségi nyelvtanban nem jelenik meg. Hasonló bottom-up megközelítés ezért tisztán a függőségi gráf alapján nem lehetséges.

A függőségi elemzésnek azonban megvan az az előnye a frázisstruktúrával szemben, hogy a különböző nyelvtani szerepek (mint például igei argumentumok: `nsubj`, `obj`, `nmod:obl`) megjelennek függőségként a fában. Ezt használva egy top-down algoritmus megkeresheti a főnévi csoportok fejeit a mondatbeli szerepeik alapján. A teljes NP ezek után kinyerhető a fejtől függő szavak felsorolásával.

A fentiek miatt az algoritmus számára az NP belseje egy fekete doboz, így csak maximális NP-k megtalálására alkalmazható. A továbbiakban ezért a maximális főnévi csoportokra koncentrálunk.

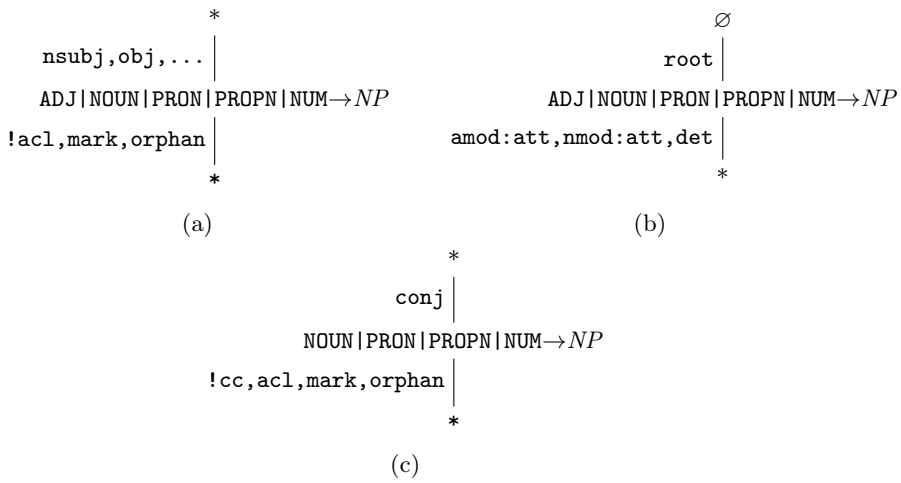
4.2. Az adat

A kísérlet során amennyire tudtunk, gold sztenderd adatokat használtunk (de lásd 4.5). Ennek megfelelően az annotációkat a Szeged korpusz különböző változataiból vettük: a morfológiai és függőségi elemzést a Szeged UD treebankből[4], az NP-ket pedig az NP-chunking korpuszból (Recski és mtsai, 2009). Mivel előbbi a Szeged korpusznak csak egy apró szeletét tartalmazza, egy egyszerű heurisztika alapján kiválogattuk az összes olyan mondatot, ami mindkettőben megtalálható. Ez 812–107–98 mondatot eredményezett a train–valid–test halmazokban.

4.3. A minták

A tanítókorpusz alapján három fő mintát azonosítottunk, ezeket a 6. ábra mutatja.

[4] `https://universaldependencies.org/treebanks/hu_szeged/index.html`



6. ábra. Minták főnévi csoportok azonosításához. A `!` a negatív feltételt jelöli

Az első minta olyan, névszói paradigmához tartozó szavakat keres, amelyek a `nsubj,obj,obj:lvc,iobj,nmod,nmod:obl,obl,obl:lvc,orphan` függőségei valaminek. A második a névszói állítmányokat fedi le, míg a harmadik a konjunkcióban álló főnévi csoportokat. Utóbbiban az `ADJ` szófaj hiánya nem tévedés: az `ADJP`-k tévesen `NP`-ként való címkézését kerüljük el vele.

Mivel a főnévi csoportok és a nyelvtani függőségek nem fedik pontosan egymást, az `NP` alatti részfára is kellett megkötéseket tennünk. Mindegyik mintánál tiltjuk a `acl,mark,orphan` függőségeket, az 6c-ben pedig a `cc`-t is. Ezen felül a visszaadott részfából kivesszük az összes, vonzat típusú élet (`nsubj,obj,...`), hiszen azok külön NP-t képeznek.

Mivel a mintaillesztő algoritmusunk nem veszi figyelembe a szavak sorrendjét, az alaprendszert ki kellett egészítenünk még egy NP-specifikus lépéssel: a fejtől jobbra minden `case,advmod,cc,punct` token törlünk, illetve a csoport elejéről az esetlegesen odakerült vesszőket is.

4.4. Eredmények

A chunkert lefuttattuk a teljes korpuszon, és a `seqeval`[5] csomaggal értékeltük ki. Az eredményeket a 2. táblázat listázza.

Mint látható, az eredmény elmarad a „state of the art”-tól: a 89%-os f-score nem mérhető a `emBERT` 97 százalékához (Nemeskey, 2021); gyakorlatilag a `hunchunk` szintjét hozza. Viszont a módszer erejét mutatja, hogy ezt három egyszerű szabály alkalmazásával értük el. A rendszer szabályalapúságának megfelelően a pontszámok függetlenek attól, hogy a korpusz melyik részén futtatjuk le.

[5] `https://github.com/chakki-works/seqeval`

	Accuracy	f-score
train	93,53%	0.89
valid	93,84%	0.88
test	93,86%	0.89

2. táblázat. NP-chunking kiértékelése

4.5. Értékelés és hibaelemzés

A hibásan címkézett szavak kézi átnézése során három hibaosztályt sikerült elkülöníteni. Ezeknek csak egyike származik a rendszerből; a másik kettő a felhasznált adatok problémáira mutat rá.

A Szeged UD Treebank sajnos számos szisztematikus hibás elemzést tartalmaz. A legfeltűnőbb a melléknévből képzett határozószók (*tisztán, remekül*, stb.) `ADJ`-ként címkézése. Ez magát a függőségi elemzőt is megtéveszthette, ami sokszor `obl`-ot társított ezekhez a szavakhoz, így ezeket mi is tévesen `NP`-nek jelöltük. Egyéb hibaként előfordultak `nmod:att` függőségek igei fejjel, rosszul kötött `conj`-ok, stb.[6]

Az NP-chunking korpusz sem hibamentes. Több esetben is előfordul, hogy a birtokos NP-összetételeket (pl. *a kormány egészének ellenőrzését*) nem vonja össze egy magasabb szintű főnévi csoportba, annak ellenére, hogy mind Kornai (1985), mind Recski (2014) listázza ezt a szabályt. Hasonló inkonzisztencia figyelhető meg az adpozíciók címkézésében: *Budapestre, a csepeli szabadkikötőbe* két külön NP, míg *a fővárosban a XVIII. kerületben* egy. Végül, vannak határozói „NP"-k, mint a *még egyszer* (`ADV ADV`).

Ezek a hibák sajnos megnehezítik a rendszer teljesítményének pontos kiértékelését, illetve továbbfejlesztését. Jó hír, hogy a fentiek fényében a 89%-os f-score egy konzervatív becslésnek tűnik; a „nettó" teljesítmény valószínűleg így 90% felett lehet. Viszont ahhoz, hogy a rendszert javítsuk, és újabb mintákat tudjunk hozzáadni, először a tanítókorpuszt kell kijavítani, vagy egy jobb minőségű tanítókorpuszt előállítani. A szerzők ezért a jövőben először erre terveznek koncentrálni.

Egy másik kísérlet a megközelítés más gyengeségére is rámutatott. Eleinte egy, a Szeged UD Treebanken feltanított Stanza modellel (Qi és mtsai, 2020) elemeztük a mondatokat. A függőségi elemzés rossz minősége miatt azonban az eredmények nagyon elmaradtak a fent közölttől. Ez két okra vezethető vissza: egyrészt a Szeged UD Treebank nagyon kicsi, használható modellt tanítani rajta nem igazán lehet. Másrészt egy szabályalapú megközelítés igényli a pontos bemenetet, ezért rendszerünk nem lesz olyan robosztus, mint egy gépi tanuláson alapuló modell. Azonban ez a hátrány csökkenthető lenne egy megfelelő méretű és minőségű függőségi nyelvtan tanítókorpusszal.

Végezetül, az NP-chunking feladat rámutatott a felhasznált gráfos leírás elégtelen voltára: ahhoz, hogy a le tudjuk vágni a felesleges függőségeket az NP körül,

[6] A szerzők ezeket a hibákat a korpusz Github repozitóriumában is jelentették.

illetve, hogy minimális NP-ket le tudjunk írni vele, sorrendezett reprezentációra van szükség. A jövőben ebbe az irányba szeretnénk továbbfejleszteni a rendszert.

5. Összegzés

Kutatásunk célja a függőségi elemzés mintaillesztésének alkalmazása volt a tagmondatra bontás és az NP-chunking feladatokban. Bár előbbihez még nem áll rendelkezésre tesztelésre használható gold sztenderd adat, az előzetes tapasztalataink alapján elmondható, hogy a tagmondatok és a köztük lévő viszonyok elvileg jól meghatározhatók a függőségi elemzés alapján. Kérdéses azonban, hogy mennyire számíthatunk a függőségi elemzés helyességére.

Az NP-chunking módszerünk kiértékelésénél is nehézséget okozott, hogy az elvben gold sztenderd korpuszok több hibát is tartalmaztak, mind a függőségi elemzésben, mind az NP-chunkingban. Mindezekkel együtt 89%-os f-score-t értünk el, ami ugyan elmarad a state-ot-the-arttól, de abból a szempontból mégis ígéretes, hogy ezt az eredményt egy egyszerű szabályrendszerrel értük el. Ez alapján a függőségi elemzés mintaillesztése további kutatásra érdemes módszer lehet a hasonló feladatokban.

Hivatkozások

Csendes, D., Csirik, J., Gyimóthy, T., Kocsor, A.: The szeged treebank. In: Matoušek, V., Mautner, P., Pavelka, T. (szerk.) Text, Speech and Dialogue. pp. 123–131. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)

Gyulai, L.: Az igekötők legjellemzőbb argumentumszerkezetváltoztató hatásainak korpuszalapú vizsgálata. In: Gráczi, T.E., Ludányi, Z. (szerk.) Alknyelvdok15. Doktoranduszok tanulmányai az alkalmazott nyelvészet köréből. pp. 176–198. Nyelvtudományi Kutatóközpont (2021)

Kilpeläinen, P., és mtsai: Tree matching problems with applications to structured text databases. Ph.D.-értekezés, Department of Computer Science, University of Helsinki (11 1992)

Kornai, A.: The internal structure of noun phrases. In: István, K. (szerk.) Approaches to Hungarian, pp. 79–92. JATE (1985), `https://eprints.sztaki.hu/7848/`

Lengyel, K.: Magyar grammatika. Nemzeti Tankönyvkiadó, Budapest (2000)

Luotolahti, J., Kanerva, J., Pyysalo, S., Ginter, F.: SETS: Scalable and efficient tree search in dependency graphs. In: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations. pp. 51–55. Association for Computational Linguistics, Denver, Colorado (Jun 2015), `https://aclanthology.org/N15-3011`

Metheniti, E., Park, P., Kolesova, K., Neumann, G.: Identifying grammar rules for language education with dependency parsing in German. In: Proceedings of the Fifth International Conference on Dependency Linguistics (Depling, SyntaxFest 2019). pp. 100–111. Association for Computational Linguistics, Paris, France (Aug 2019), `https://aclanthology.org/W19-7712`

Nemeskey, D.M.: Introducing huBERT. In: XVII. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY2021). p. TBA. Szeged (2021)

Nichols, E., Reisert, P.: Dgrep: A pattern-matching tool for dependency trees. In: Proceedings of the Twentieth Annual Meeting of the Association for Natural Language Processing.(to appear) (2014)

Qi, P., Zhang, Y., Zhang, Y., Bolton, J., Manning, C.D.: Stanza: A Python natural language processing toolkit for many human languages. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations (2020), https://nlp.stanford.edu/pubs/qi2020stanza.pdf

Ramshaw, L., Marcus, M.: Text chunking using transformation-based learning. In: Third Workshop on Very Large Corpora (1995), https://aclanthology.org/W95-0107

Recski, G.: Hungarian noun phrase extraction using rule-based and hybrid methods. Acta Cybernetica 21(3), 461–479 (2014)

Recski, G., Varga, D., Zséder, A., Kornai, A.: Főnévi csoportok azonosítása magyar-angol párhuzamos korpuszban [Identifying noun phrases in a parallel corpus of English and Hungarian]. VI. Magyar Számítógépes Nyelvészeti Konferencia [6th Hungarian Conference on Computational Linguistics] (2009)

Szemes, B.: Kidolgozott viszonyok. a tagmondatkapcsolatok automatikus azonosításának hasznosíthatósága a stilisztikában és az irodalomtörténet-írásban. Digitális Bölcsészet 2021(4), 31–77 (2021)

Wallis, S.: Searching treebanks and other structured corpora. In: Corpus linguistics: An international handbook, pp. 738–759. Handbücher zur Sprache und Kommunikationswissenschaft, De Gruyter Mouton, Berlin (2008)