UNIVERZITET U NIŠU
ELEKTRONSKI FAKULTET

Chunker glagolskih fraza u korpusu *CoNLL-2000*

Mentor:
Prof. dr Suzana Stojković

Student:
Svetlana Mančić, 1423

Niš, septembar 2022. god.

Sadržaj

1. Uvod

Cilj ovog rada biće razvoj plitkog analizatora(chunkera) baziranog na pravilima u korpusu *CoNLL-2000*, korišćenjem python programskog jezika i nltk biblioteke.

U sledećem poglavlju biće predstavljena implementacija VP chunkera u korpusu *CoNLL-2000*. Pre svega biće reči o vrstama reči koje čine glagolske fraze, a zatim će biti opisan način implementacije. Na kraju trećeg poglavlja biće priložena evaluacija razvijenog chunkera i biće diskusije o najčešćim problemima do kojih dolazi.

2. Razvoj chunkera u korpusu *CoNLL-2000*

Zadatak je bio razvoj chunkera za jedan od chunk tipova u korpusu *CoNLL-2000* korišćenjem *RegexpChunkParser*-a uz korišćenje proizvoljne kombinacije pravila za *chunking*, *chinking*, *merging* i *splitting*. Razvijen chunker izdvaja glagolske fraze (*verb phrase – VP*).

2.1. Vrste reči u glagolskim frazama

Pre početka same implementacije, bilo je neophodno podsetiti se gramatike engleskog jezika, tačnije kako se grade vremena u engleskom jeziku i pozicije priloga kada su ugnježdeni između glagola. Nakon podsećanja i analize korpusa, došla sam do zaključka da glagolske fraze mogu biti jednostavne i da sadrže jednu reč (slika 1), a da sa druge strane, mogu biti i veoma složene kao one koje sadrže u sebi više glagola, prilog i modalni glagol (slika 2).

Economists/ NNS [$_{VP}$ suggested/VBD] that/IN if/IN the/DT pound/NN [$_{VP}$ falls/VBZ] much/JJ below/IN...

Slika 1. Prosta glagolska fraza

The/DT government/NN [VP will/MD be/VB forced/VBN to/TO increase/VB] rates/NNS ...

Slika 2. Složena glagolska fraza

Takođe, primetila sam da, ukoliko je prilog ili priloška fraza ugnježđena unutar glagolske fraze i nalazi se ispred glavnog glagola, postaje deo glagolske fraze (slika 3). Na slici 3 se odlično vidi kako se prilog (tag RB) ugnježđen unutar glagolske fraze svrstava u istu, dok prilog koji se nalazi van fraze se ne chunkuje.

This/DT month/NN [VP could/MD be/VB particularly/RB challenging/VBG] because/IN almost/RB every/DT problem/NN...

Slika 3. Prilog ugnježđen u VP

Dodatno, kada se koristi negacija, reč *not,* ili njen kraći oblik *n't*, tagovana je na isti način kao i prilog (not/RB), tako da pozicija RB taga u *ChunkString*-u može biti promenljiva u zavisnosti da li se koristi negacija, prilog ili oba, na šta takođe treba obratiti pažnju (slika 4).

that/WDT [VP can/MD not/RB be/VB entirely/RB solved/VBN] in/IN the/DT...

Slika 4. VP sa negacijom i prilogom

S druge strane, pridevi koji prate glavni glagol ne pripadaju glagolskoj frazi. Kod rečenice na slici 5 vidi se da pridev koji prati glagol ne pripada glagolskoj frazi.

… White/NNP House/NNP and/CC Congress/NNP over/IN who/WP [VP is/VBZ] responsible/JJ for/IN...

Slika 5. Pridev nakon VP

U engleskom jeziku, kada se nađu dva glagola u nekom od složenih vremena, umesto ponavljanja pomoćnih glagola vrši se nadovezivanje glavnih glagola pomoću veznika. U tom slučaju, pomoćni glagoli i glavni glagoli zajedno sa veznikom čine jednu glagolsku frazu (slika 6).

GE/NNP ‘s/POS Power/NNP Generation/NNP subsidiary/NN [$_{VP}$ will/MD operate/VB and/CC maintain/VB] the/DT plant/NN...

Slika 6. Glagoli u istom vremenu spojeni veznikom

2.2. Chunker klasa

Nakon pažljivog analiziranja problema, zaključila sam da je najbolji pristup chunkovanje manjih delova, a zatim korišćenjem merge pravila spajanje u veće celine. Sve funkcije vezane za chunker objedinjene su u okviru klase *ChunkerClass*. Definisanje pravila izvršeno je još u konstruktoru i započela sam pre svega definisanjem *ChunkRule* pravila, koje chunkuje sve reči sa tagovima VB, VBZ, VBP, VBG, VBN, VBD, MD (slika 7).

```
Chunk_Parts = ChunkRule(r"<VB|VBZ|VBP|VBG|VBN|VBD|MD>", "Chunk smaller parts")
```

Slika 7. ChunkRule za izdvajanje manjih delova glagolskih fraza

Zatim su definisana nekoliko *ChunkRuleWithContext* pravila, koja za cilj imaju chunkovanje priloga i veznika u kontekstu glagolske fraze (slika 8). Cilj ovih pravila je chunkovanje samo priloga i veznika u kontekstu VP kako bi se izbeglo chunkovanje onih koji ne bi trebalo da se nađu u glagolskoj frazi.

```
adverb = ChunkRuleWithContext(r"<MD|TO|VB.*>+", r"<RB|RBS|RBR>+", r"<VB.*>+",
"Chunk adverb") #chunk adverb in context with verbs

cc = ChunkRuleWithContext(r"<VB.*>+", r"<CC>", r"<VB.*>+", "Chunk CC") #chunk
and connecting two verbs
```

Slika 8. ChunkRuleWithContext za izdvajanje priloga i veznika u kontekstu sa glagolom

Pošto prilog može da se nađe na različitim mestima u glagolskoj frazi, stripuje se kontekst pomoću *StripRule*, a sam prilog je pomoću više *MergeRule* pravila dovodi u vezu sa ostalim članovima glagolske fraze.

```
Strip_Context = StripRule(r"<MD|TO|VB.*>", "Strip context")
```

Slika 9. StripRule za uklanjanje konteksta

U ovoj fazi dobijeni su gradivni delovi glagolskih fraza i pomoću *MergeRule,* ukoliko je to potrebno se spajaju u složenije fraze. Pre svega, koristi se *MergeRule* za spajanje glavnog glagola i priloga, modalnog glagola i negacije i veznika i glavnog glagola (slika 10).

```
merge_not = MergeRule(r"<MD>", r"<RB|RBS|RBR>+", "Merge won't/wouldn't")
#example: won't

merge_cc_vb = MergeRule(r"<CC>", r"<VB.*>", "Merge CC and verb")

merge_rb_vb = MergeRule(r"<RB|RBS|RBR>+", r"<VB.*>", "Merge adverb with a
verb")
#example: partly restore
```

Slika 10. MergeRule za spajanje veznika, priloga i modalnog glagola sa glavnim glagolom

Sledi spajanje manjih chunkovanih delova u složenije glagolske fraze (slika 11). Primenom *merge_pas_simple* pravila na *ChunkString* dobija se chunk koji odgovara negaciji glagola u *Past Simple Tense*-u. Za svako od definisanih pravila dat je i primer chunka koji se dobije primenom pravila. Pravila *merge_cc_pres_cont, merge_cc_past_part* i *merge_cc_rest* deluju redundantno, ali je jako bitno da glagoli koji se spajaju veznikom budu u istom vremenu, inače može da se dobije neželjeni rezultat.

```
#merge smaller chunks into more complex chunks
merge_past_simple = MergeRule(r"<VBD>", r"<RB|RBS|RBR><VB|VBP>", "Merge past
simple")
# example: didn't do
merge_pres = MergeRule(r"<VB|VBZ|VBP>", r"<RB|RBS|RBR>*<VB|VBZ|VBP|VBG>",
"Merge infinitive and present continuous")
# example: 'm waiting

merge_pres_perf = MergeRule(r"<VB|VBZ|VBP>", r"<RB|RBS|RBR>*<VBN|VBD>", "Merge
present perfect")
 # example: have worked
merge_past_perf_cont = MergeRule(r"<VBN|VBD>", r"<RB|RBS|RBR>*<VBN|VBD|VBG>",
"Merge past perfect and past continuous")
# example: had worked or was working
merge_pres_past_perf_cont = MergeRule(r"<VB|VBZ|VBP|VBN|VBD><RB|RBS|RBR>?<VBN|
VBD><RB|RBS|RBR>?", r"<VBG>","Merge present and past perfect continuous")
# example: have/had been working or hadn't been entirely honest

merge_going_to = MergeRule(r"<VB|VBZ|VBP><RB|RBS|RBR>*<VBG>", r"<TO><VB>",
"Merge going to-future")
 # example: am going to work
merge_fut_sim_cont = MergeRule(r"<MD><RB|RBS|RBR>*<VB|VBZ|VBP>", r"<VBD|VBN|
VBG>","Merge Future Perfect Simple and Continuous")
# example: will have worked/ will be working
merge_fut_cont = MergeRule(r"<MD><RB|RBS|RBR>*<VB|VBZ|VBP><VBD|VBN>",
r"<VBG>","Merge Future Perfect Continuous")
 # example: will have been working
merge_future = MergeRule(r"<MD><RB|RBS|RBR>*<VB|VBZ|VBP>?", r"<TO>?<VB|VBZ|
VBP>","Merge future")
# example: would help fill/to show
merge = MergeRule(r"<VBG>", r"<VBD|VBN>", "Merge")
 # example: being given

merge_with_inf = MergeRule(r"<.*>*<VB.*>", r"<TO><VBP|VB|VBZ>", "Merge with
infinitive")
# example: expected to show

merge_cc_pres_cont = MergeRule(r"<.*>*<VBG>", r"<CC><VBG>", "Merge two verbs")
 # example: waiting and watching
merge_cc_past_part = MergeRule(r"<.*>*<VBN>", r"<CC><VBN>", "Merge two verbs")
 # example: waited and saw
merge_cc_rest = MergeRule(r"<.*>*<VBP|VB|VBZ>", r"<CC><VB|VBP|VBZ>", "Merge
two verbs")  # example: wait and see
```

Slika 11. MergeRule za spajanje u složenije glagolske fraze

Na kraju su definisana *UnChunkRule* pravila, kojima se uklanjaju tagovi koji ne treba da budu chunkovani (slika 12).

```
unchunk_unused = UnChunkRule(r"<RB|RBS|RBR|CC>+", "Unchunk unused in VP")

unchunk_cc_vb = UnChunkRule(r"<CC><VB.*>", "Unchunk CC VB without VB before")
```

Slika 12. UnChunkRule

Sva prethodno definisana pravila dodaju se u listu, koja se prosleđuje prilikom kreiranja *RegexpChunkParser* objekta (slika 13).

```
#add all rules to one list
rules = [cc, adverb, strip_context, verb, infinitive, chunk_parts,
         merge_rb_vb, merge_not, merge_cc_vb, merge_past_simple,
         merge_with_inf, merge_pres, merge_pres_perf,
         merge_past_perf_cont, merge_pres_past_perf_cont, merge_going_to,
         merge_fut_sim_cont, merge_fut_cont, merge_future, merge,
         merge_cc_pres_cont, merge_cc_rest, merge_cc_past_part,
         unchunk_cc_vb, unchunk_unused, unchunk_adj]

# initialize chunker, use list of rules, label chunks VP
self.chunker = RegexpChunkParser(rules, chunk_label='VP')
self.chunk_score = ChunkScore()
```

Slika 13. Kreiranje RegexpChunkParser objekta

Za chunkovanje je definisana funkcija chunk (slika 14), koja se poziva sa dva parametra. Prvi je *gold_text*, koji predstavlja već chunkovan tekst, a drugi *tagged_text*, koji je potrebno chunkovati. Prolazi se kroz sve rečenice iz tagovanog teksta, svaka se parsira i poredi sa odgovarajućom chunkovanom rečenicom.

```
def chunk(self, gold_text, tagged_text):
    for i in range(len(tagged_text)):
        result = self.chunker.parse(tagged_text[i])
        self.chunk_score.score(gold_text[i], result)
```

Slika 14. Funkcija chunk

3.3. Evaluacija

Za evaluaciju chunkera iskorišćen je *ChunkScore*. Klasa *ChunkerClass* poseduje funkciju *evaluate* (slika 15), koja obračunava parametre u procentima i štampa na standardnom izlazu.

```
def evaluate(self):
    precision = round(self.chunk_score.precision()*100, 2)
    recall = round(self.chunk_score.recall()*100, 2)
    f_measure = round(self.chunk_score.f_measure()*100, 2)
    print("Precision: " + str(precision) + "%")
    print("Recall: " + str(recall) + "%")
    print("F-measure: " + str(f_measure) + "%")
```

Slika 15. Evaluacija chunkera

3.4. Glavni program

Na slici 16 je struktura glavnog programa. U glavnom programu, objedinjene su chunkovane rečenice iz *test* i *train* skupa iz korpusa C*oNLL-2000* i tagovane rečenice iz *test* i *train* skupa iz korpusa *CoNLL-2000*. Instanciran je objekat chunker i pozvana funkcija *chunk* iz klase *ChunkerClass*, kojoj su prosleđene chunkovane i tagovane rečenice. Na kraju, odstampana je evaluacija chunkera (slika 17).

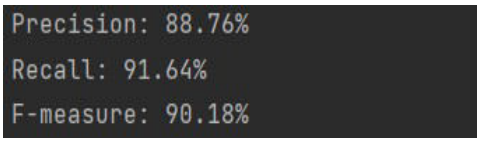
```
import nltk
from nltk.corpus import conll2000
from nltk.chunk.regexp import *
from nltk.chunk.util import *
from ChunkerClass import ChunkerClass

chunked_sents = conll2000.chunked_sents('test.txt', chunk_types=['VP']) +
conll2000.chunked_sents('train.txt', chunk_types=['VP'])

tagged_sents = conll2000.tagged_sents('test.txt') +
conll2000.tagged_sents('train.txt')

chunker = ChunkerClass()
chunker.chunk(chunked_sents, tagged_sents)
chunker.evaluate()
```

Slika 16. Glavni program



Slika 17. Evaluacija razvijenog chunkera

3.5. Problemi

Chunking se vrši nad tagovanom verzijom teksta iz korpusa, međutim, neka stabla su veoma kompleksna, a neke oznake su nisu tačne. Na primer, u nekim slučajevima se dešava da glavni glagol ima tag NN.* (slika 18).

The/DT labor-management/JJ buy-out/NN group/NN [$_{VP}$ plans/NNS to/TO keep/VB] its/PRP$ offer/NN

Slika 18. Glavni glagol sa tagom NNS

U tom slučaju može se dozvoliti da NNS bude član glagolske fraze ili se može izostaviti, kao što sam ja uradila. Do grešaka dolazi u bilo kom slučaju.

Takođe, pridevi koji se završavaju na -ed, -en, -ing ili nekim drugim nastavkom koji može imati glagol često se nađu sa tagom VB.* umesto JJ, kao i imenice koje imaju isti oblik kao glagol budu tagovane tagom VB.* umesto NN.*, zbog čega takođe dolazi do grešaka. Ono što sam ja uradila kako bih pokušala da minimizujem broj grešaka u tom slučaju je chunking celokupne fraze zajedno sa članom, imenicom ili drugim pridevima koji se mogu tu naći kako takav chunk ne bi zaostao na kraju procesa, a zatim na samom kraju je dodato pravilo koje vrši unchunking identičnog chunka (slika 19).

```
verb = ChunkRule(r"<DT|JJ|WRB|POS|CD><VB.*><JJ|CD|NN.*>", "Verb assumes
adjective function") #verb as an adjective
unchunk_adj = UnChunkRule(r"<DT|JJ|WRB|POS|CD><VB.*><JJ|CD|NN.*>", "Unchunk
verb mistaken for adjective")
```

Slika 19. Pridev tagovan kao glagol

Kako je rađeno sa tagovanim rečenicama, a ne sa *raw* tekstovima, znakovi interpunkcije nisu uklonjeni iz tagovanih rečenica, već su tagovani zajedno sa ostalim rečima. Dešava se da se znak interpunkcije nađe na neobičnom mestu, na primer između to i glagola u infinitivu. U tom slučaju, chunker ne detektuje to kao VP chunk.

Kod inverznih rečenica, pomoćni glagol ne treba da bude deo nijedne glagolske fraze. Međutim, do može biti i glagol, a kako je tagovan kao glagol samim tim se i chunkuje.

[$_{VP}$ Do/VBP] they/PRP [$_{VP}$ believe/VBP] that/IN ...

Slika 20. Pomoćni glagol