TakeLab

Laboratorij za analizu teksta i inženjerstvo znanja
Text Analysis and Knowledge Engineering Lab
Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva
Unska 3, 10000 Zagreb, Hrvatska

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 4754

Otkrivanje pogrešaka leksičkog transfera u tekstovima učenika stranog jezika

Marin Kačan

Zagreb, lipanj 2016.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 11. ožujka 2016.

ZAVRŠNI ZADATAK br. 4754

Pristupnik: **Marin Kačan (0036476517)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Otkrivanje pogrešaka leksičkog transfera u tekstovima učenika stranog jezika**

Opis zadatka:

Leksički transfer odnosi se na prijenos znanja o riječima jednog jezika u drugi jezik. Učenici stranog jezika nerijetko čine pogreške leksičkog transfera, pogrešno prevodeći višeznačne riječi izvornog (materinjeg) jezika u ciljni (strani) jezik. U okviru paradigme računalno potpomognutog učenje jezika, od velike bi koristi bili postupci za automatsko otkrivanje i ispravljanje takvih semantičkih pogrešaka. Tema završnog rada jest automatsko otkrivanje pogrešaka leksičkog transfera uslijed višeznačnosti riječi u tekstovima izvornog jezika, s naglaskom na dvorječne pridjevsko-imeničke i glagolsko-objektne sintagme. Upoznati se s osnovnim koracima računalne obrade teksta i razviti modul za ekstrakciju dvorječnih sintagmi. Upoznati se s modelima za otkrivanje semantičke devijacije na temelju statističke analize korpusa. Razviti model za generiranje prijevodnih kandidata uporabom rječnika te model za njihovo ocjenjivanje uporabom statističke analize korpusa i nadziranog strojnog učenja. Ispitati rad sustava na tekstovima kineskih učenika engleskoga jezika koje ustupa sveučilište Xi'an Jiaotong-Liverpool ili na umjetno generiranom skupu podataka. Provesti eksperimentalno vrednovanje modela i statističku obradu rezultata. Radu priložiti izvorni i izvršni kod razvijenog sustava, označene skupove podataka i potrebnu dokumentaciju.

Zadatak uručen pristupniku: 18. ožujka 2016.
Rok za predaju rada: 17. lipnja 2016.

Mentor:

Doc. dr. sc. Jan Šnajder

Djelovođa:

Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za završni rad modula:

Prof. dr. sc. Siniša Srbljić

Zahvaljujem svojoj obitelji, djevojci i prijateljima na konstantnoj i bezuvjetnoj podršci svih ovih godina mojeg studija. Zahvaljujem mentoru Janu Šnajderu na mnogobrojnim savjetima, motivaciji i poticanju znanstveno-istraživačkog entuzijazma.

Sadržaj

1. Uvod

Opće je poznato da količina ljudskog znanja ubrzano raste te da je dnevna količina generiranih podataka svakim danom sve veća. Velik dio tog znanja (uključujući ovaj rad) pohranjen je u nestrukturiranom, tekstnom obliku. Obrađivati tolike količine podataka postaje sve teže, te da bi se s tim problemom uhvatili u koštac, ljudi moraju posegnuti za alatom čija moć također svakim danom ubrzano raste – računalom.

Obrada prirodnog jezika (engl. *Natural Language Processing*) grana je umjetne inteligencije čiji je glavni cilj omogućiti obosmjernu komunikaciju čovjeka i računala putem prirodnog jezika. Teži se tome da računalo bude sposobno razumjeti dani tekst koji je stvorio čovjek (razumijevanje prirodnog jezika, engl. *Natural Language Understanding*), ali također i samo stvoriti i korisniku prikazati smislen i jezično ispravan tekst (stvaranje prirodnog jezika, engl. *Natural Language Generation*).

Neki od najčešćih zadataka obrade prirodnog jezika su strojno prevođenje, automatizirano sažimanje teksta, analiza sentimenta i emocija, ekstrakcija informacija, odgovaranje na pitanja, označavanje vrsta riječi i ispravljanje pogrešaka.

Utjecajem globalizacije, engleski se jezik proširio cijelim svijetom, probio barijere raznih kultura, te je broj govornika engleskog jezika kojima je on drugi jezik (engl. *L2 speakers*) znatno veći u odnosu na sve ostale jezike.

Materinji jezik mnogih od tih spomenutih L2-govornika u mnogočemu se razlikuje od engleskog. Uz to, engleski se jezik velikim dijelom širi slobodnim medijem koji od govornika ne zahtijeva gramatičku ispravnost (internet). Uzimajući te dvije činjenice u obzir, jasno je da će engleski jezik kojim govori dio L2-govornika biti manjkav i gramatički neispravan.

Kako je za engleski jezik broj L2-govornika gotovo tri puta veći od broja L1-govornika (oni kojima je on materinji jezik), postoji opasnost opće degradacije kvalitete govorenog i pisanog engleskog jezika. Postoji opasnost da se ispravan engleski jezik izgubi u moru svojih iskrivljenih inačica koje nastaju miješanjem jezičnih pravila i izraza izvornog jezika nekog područja s engleskim jezikom.

Uz adekvatnu edukaciju govornika, automatizirano ispravljanje pogrešaka moglo bi imati ključnu ulogu u sprječavanju takvog razvoja događaja. U tome, između ostalog, leži važnost te grane obrade prirodnog jezika, te se time bavi i ovaj rad.

Kada se neka osoba nađe u procesu usvajanja drugog jezika (L2), velika je vjerojatnost da će na usvajanje tog jezika utjecati materinji jezik (L1) spomenute osobe. Moguće je da će takva osoba, dok još uči L2, u nedostatku izraza koji dobro opisuje ono što želi reći, doslovno prevesti odgovarajući izraz iz L1, koji je u tom kontekstu za jezik L1 ispravan, ali za L2 ne mora nužno biti, te često i nije. Ta se pojava zove leksički transfer, a podskup slučajeva u kojima je prenošenje izraza iz L1 u L2 rezultiralo rečenicom koja je gramatički neispravna u jeziku L2 zove se negativni leksički transfer. Tema je ovog rada upravo detekcija i ispravljanje negativnog leksičkog transfera za dane jezike L1 i L2. Podrobniji i oprimjeren opis ovog fenomena dan je u poglavlju 2.

U poglavlju 3 opisan je računalni model koji je osmišljen kao pokušaj rješavanja problema. Poglavlje 4 sadrži opis programske implementacije rješenja, dok su u poglavlju 5 prikazani rezultati vrednovanja modela. Poglavlje 6 sadrži komentar rezultata i zaključak rada.

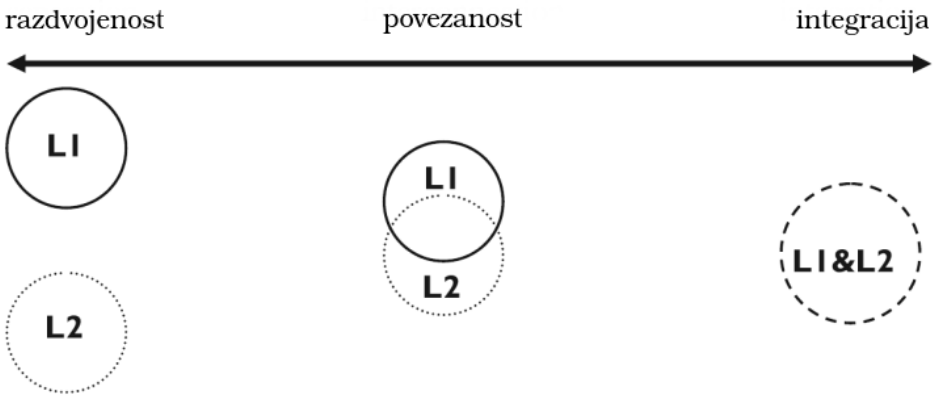
2. Problem

2.1. Leksički transfer

O'Malley i Chamot (1995) definiraju transfer kao "korištenje onog što je već poznato o jeziku kao pomoć razumijevanju i stvaranju".

Neka je jezik koji neka osoba uči L2, a neka je materinji jezik te osobe L1. U pravilu, učenici jezika L2 već su dobro usvojili konceptualna, semantička i leksička svojstva jezika L1, te ta znanja i navike utječu na to kako će usvojiti jezik L2. Kada su u pitanju svojstva jezika koja vrijede i za L2, tada ti učenici profitiraju od znanja koje imaju o L1. S druge strane, isto to znanje im otežava usvajanje koncepata koji se u L2 razlikuju od onih u L1.

Odnos između jezika L1 i L2 u umu jedne osobe ponekad se prikazuje integracijskim kontinuumom koji spominje Cook (2003):



Slika 2.1: Integracijski kontinuum, Cook (2003)

Dva jezika u umu L2-učenika kreću se od potpune odvojenosti do potpune integracije. Prema Cook (2014), nema sumnje da nitko ne može održati dva

jezika potpuno odvojenima, niti je itko potpuno nesposoban razlikovati ih – potpuna odvojenost i potpuna integracija su idealni oblici. L2-govornici nalaze se negdje između njih, duž integracijskog kontinuuma. Upravo zbog nemogućnosti potpunog odvajanja dva jezika događa se da se L2-govornici koriste jezikom koji sadržava svojstva i jezika L1 i jezika L2.

Kako je spomenuto u uvodu, osoba koja uči L2, u nemogućnosti da pronađe adekvatan izraz u jeziku L2 za koncept koji želi opisati, često će posegnuti za izrazom koji taj koncept opisuje u jeziku L1. Ako taj izraz jednostavno prevede riječ po riječ, a da ne vodi računa o tome odgovara li taj prevedeni izraz i značenja pojedinih riječi kontekstu unutar kojeg se taj izraz nalazi u jeziku L2, govorimo o leksičkom transferu.

2.2. Istraživanje nad kineskim studentima

Chengchen (2015) opisuje istraživanje nad kineskim studentima u vezi leksičkog transfera. U istraživanju su analizirani radovi na engleskom jeziku studenata čiji je L1 kineski jezik, a L2 engleski. Cilj istraživanja bio je utvrditi koliko se često i zbog čega javljaju pogreške leksičkog transfera.

U tom radu, navedena su tri tipa pogrešaka leksičkog transfera:

1. Višeznačnost (engl. *polysemy*) kineskih riječi
2. Izravno prevođenje kineskih kolokacija (engl. *collocations*)
3. Izravno prevođenje kineskih višerječnih izraza (engl. *multiword expressions*, *MWE*)

Valja napomenuti da je ovakva podjela između kolokacija i višerječnih izraza neuobičajena u obradi prirodnog jezika. Često se te dvije skupine tretiraju kao jedna.

Sve pronađene pogreške razvrstane su u jednu od tri navedene kategorije tako da se vidi koliki udio ima svaka i da se one mogu promatrati odvojeno.

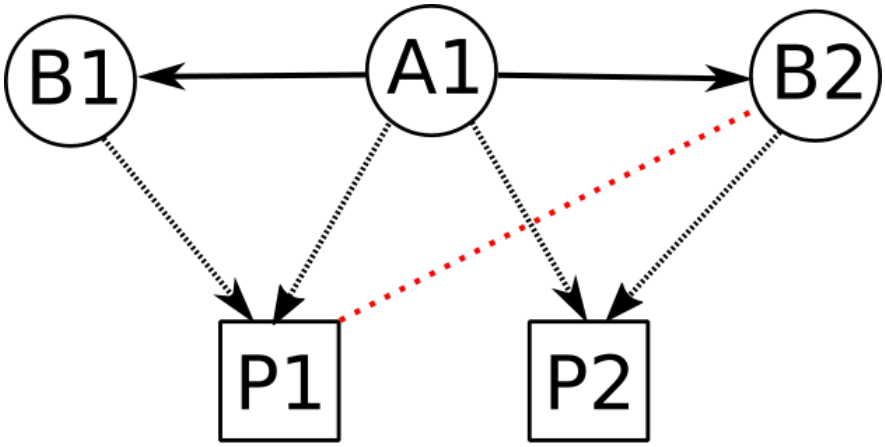
2.2.1. Višeznačnost

Uobičajeno je u prirodnim jezicima da se potpuno različite pojave označavaju istim riječima. Dakle, jedna riječ ima više značenja. Problem je u tome što svaki jezik ima zasebne slučajeve višeznačnosti koji se ne poklapaju s ostalim jezicima.

Pretpostavimo sljedeće:

1. Postoji pojava P1 koju u jeziku L1 označava riječ A1.
2. Govornik želi tu pojavu izreći u jeziku L2.
3. Riječ A1 u jeziku L1 je višeznačna i označava pojave P1 i P2.
4. Obje pojave P1 i P2 u jeziku L2 imaju vlastitu, različitu oznaku B1 i B2. Dakle B1 i B2 su mogući prijevodi riječi A1.
5. Budući da je L2 jezik koji još nije u potpunosti usvojio, govornik zna samo prijevod A1 $\rightarrow$ B2.

Pokuša li govornik u tom slučaju prijevodom A1 $\rightarrow$ B2 prevesti riječ A1 sa željom da na jeziku L2 opiše pojavu P1, napravit će grešku leksičkog transfera jer će zapravo riječju B2 opisati pojavu P2.



Slika 2.2: Odnos riječi i pojava koje opisuju

Primjeri takvih slučajeva za L1 – kineski, L2 – hrvatski su:

L1 riječ	L2 riječi
看 (kan)	gledati, vidjeti, čitati, posjetiti
地方 (di fang)	mjesto, područje, svojstvo, dio

Primjeri za L1 – hrvatski, L2 – engleski su:

L1 riječ	L2 riječi
plava	blue, blonde
prilog	adverb, side dish, contribution
luk	bow, arc

Izvorni bi govornik hrvatskog jezika zbog ovih višeznančosti mogao učiniti pogrešku leksičkog transfera. Npr. propitkujući svoje engleske prijatelje o receptima za priloge glavnim jelima, mogao bi doći u napast da kaže: *Do you know of any delicious* ***adverb*** *recipes?* Riječ *adverb* na engleskom jeziku označava prilog kao vrstu riječi, a ne prilog glavnom jelu. U ovom je kontekstu ispravan prijevod riječi *prilog* riječ *side dish*. Ne želi li ostati gladan, naš govornik trebat će svoje prijatelje upitati: *Do you know of any delicious* ***side dish*** *recipes?*

Pretpostavimo da istom govorniku, prilikom odabira partnerice, značajnu ulogu igra njena boja kose, te da preferira plavokose žene. Ako to bude želio dati do znanja svom društvu engleza, moguće je da će mu se ponovo omaknuti greška leksičkog transfera, te će kazati: *Call me superficial all you want, but I prefer girls with* ***blue*** *hair!* Iako se u hrvatskom jeziku plavom kosom naziva ona koja očigledno nije plave boje, u ostalim jezicima to nije tako. U engleskom jeziku sintagmom *blue hair* označava se kosa stvarno plave boje, dok se boja kose koju mi zovemo plava zapravo naziva *blonde hair*. Upravo je taj izraz naš govornik trebao iskoristiti, te reći: *Call me superficial all you want, but I prefer girls with* ***blonde*** *hair!*

Cilj je sustava koji će biti opisan da detektira i ispravi upravo takve pogreške.

2.2.2. Fraze i višerječni izrazi

Fraza je jezična jedinica kojoj je oblik ustaljen stalnom upotrebom, skup riječi sa ustaljenim značenjem koje je različito od zbroja značenja njegovih članova.[1]

Problem nastaje zbog toga što taj skup riječi, kada se svaka njegova riječ prevede zasebno, u ostalim jezicima u pravilu nema jednako (niti ikakvo) značenje. Ono što nekoj frazi jezika L1 daje smisao i značenje je činjenica da su ju govornici jezika L1 učestalo koristili da bi opisali određenu pojavu. Ako je zbroj značenja pojedinih riječi u frazi različit od značenja fraze, tj. značenje pojedine riječi može biti samo po sebi semantički udaljeno od značenja fraze, malo je vjerojatno da su govornici ostalih jezika odabrali baš te iste pojedine riječi da bi opisali istu pojavu.

Ne prepozna li da se radi o frazi svojstvenoj samo za jezik L1, L2-govornik će izravnim prijevodom te fraze (zasebnim prijevodom riječi od kojih se fraza sastoji) na jezik L2 napraviti pogrešku leksičkog transfera.

Primjeri[2] takvih slučajeva za L1 – kineski, L2 – engleski su:

L1 izvorna fraza	L2 prijevod riječ-za-riječ	L2 ispravan prijevod
学习知识 (xue xi zhi shi)	learn knowledge	gain knowledge
工具书 (gong ju shu)	tool books	reference books

Prilikom prevođenja ovakvih relacija oblika glagol-objekt (gain knowledge) i oblika pridjev-imenica (reference books) često se događa leksički transfer.

Primjeri za L1 – hrvatski, L2 – engleski su:

L1 izvorna fraza	L2 prijevod riječ-za-riječ	L2 ispravan prijevod
s druge strane	on the other side	on the other hand
gas do daske	gas to the plank	pedal to the metal

[1] http://hjp.znanje.hr/index.php?show=search_by_id&id=fFlvWBU%3D

[2] Primjeri su uzeti iz Chengchen (2015) gdje su također navedeni da oprimjere pogreške leksičkog transfera.

2.2.3. Rezultati istraživanja

Prilikom analize tekstova kineskih studenata urađenoj u istraživanju opisanom u Chengchen (2015), pronađeno je najviše grešaka leksičkog transfera prvog tipa (višeznačnost). Takvih je bilo oko 50%. Višerječni izrazi sačinjavali su 28% pogrešaka, a kolokacije 22%. Dakle, velik dio grešaka trebao bi se dati ispraviti koristeći samo višeznačnost jedne riječi, što je lakše od ispravljanja i konstruiranja čitavih točnih fraza.

3. Model

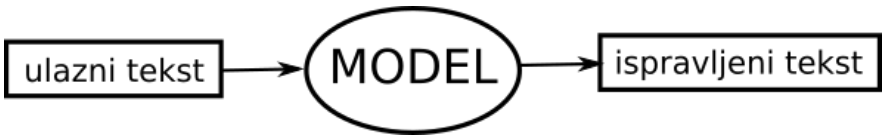
3.1. Ograničenje rješenja

Kako je problem leksičkog transfera složen i za potpuno rješenje trebalo bi pokriti mnogo različitih slučajeva, prilikom smišljanja modela u prvoj iteraciji odlučeno je da se pokušamo ograničiti na ispravljanje samo određenih jezičnih konstrukata u tekstovima.

Riječ je o dva tipa dvorječnih relacija: pridjevsko-imeničke i glagolsko-objektne sintagme. Ideja je da one pokrivaju velik dio slučajeva leksičkog transfera gdje se pridjev odnosno glagol zbog prethodno opisanog problema višeznačnosti netočno prevodi.

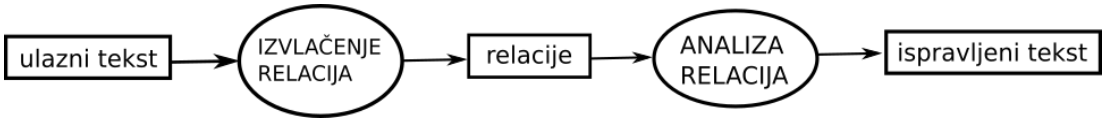
3.2. Nacrt rješenja

Najopćenitije gledano, rješenje problema čini sustav koji na ulazu ima tekst napisan jezikom L2, a na izlazu daje taj isti tekst s detektiranim i ispravljenim pogreškama leksičkog transfera.



Slika 3.1: Početni nacrt modela

Kako smo se ograničili na analizu određenih relacija, sustav mora dobiveni tekst obraditi i iz njega izvući tražene relacije. Tako da se model treba sastojati od modula za ekstrakciju relacija i modula za analizu ekstrahiranih relacija.



Slika 3.2: Nacrt modela s modulom za izvlačenje relacija

3.3. Pronalazak relacija

Da bismo iz teksta koji je dobiven na ulazu uspješno izvukli tražene relacije AN i relacije VO, potrebna je uobičajena predobrada cijelog teksta.

Svaka rečenica se tokenizira, označe se vrste riječi u njoj, te se lematizira. Tokenizacija je postupak raščlambe rečenice na riječi, dok je lematizacija postupak pretvorbe dane riječi u njen kanonski oblik – *lemu*.

Nakon toga, da bi se dobili svi odnosi između riječi unutar rečenice, tu rečenicu treba parsati i dobiti njeno sintaktičko ovisnosno stablo (engl. *dependency tree*). Za taj dio posla zadužen je parser. Važno je da sačuvamo sve parsane rečenice jer će nam i ostali dijelovi osim relacija AN i relacija VO trebati kao kontekst relacije pri detektiranju i ispravljanju.

Jednom kada imamo izvučene sve odnose između riječi u rečenici, potrebno je još odabrati samo one koji su nama bitni (pridjevsko-imenički) i (glagolsko-objektni). Nakon što se ovaj proces obavi za svaku rečenicu teksta, pronašli smo sve relacije koje su nam važne za korak analize.

3.4. Generiranje kandidata

Ono čega se sada treba prisjetiti je višeznačnost (polisemija) o kojoj smo pričali u odjeljku 2.2.1. Neke riječi u nekim jezicima mogu imati više od jednog značenja. Svako od tih značenja u nekom drugom jeziku može imati drukčiji prijevod. To je jedan od razloga zašto dolazi do leksičkog transfera pri prijevodu u drugi jezik. Govornik čiji je materinji jezik L1, s namjerom da izrekne jedno značenje riječi iz jezika L1 na jeziku L2, iz nekog razloga iskoristi prijevod za neko od drugih značenja.

Naš je cilj s ovim modelom ispravljati takve pogreške. Ono što nam je potrebno je da za svaku riječ koja se pojavi u tekstu, a pogrešna je u tom kontekstu,

možemo pronaći skup riječi iz istog jezika koje su povezane s izvornom riječi preko višeznačnosti. Taj skup zapravo će biti skup kandidata – riječi koje su se mogle pojaviti na tom mjestu od kojih će jedna biti ispravna.

3.4.1. PS-okolina

Pojasnimo ideju podrobnije. Pretpostavke su sljedeće:

1. L2-govornik čiji je materinji jezik L1 piše tekst na jeziku L2;
2. Imamo rječnik iz L1 u L2 i iz L2 u L1. Taj rječnik je 1-na-n, tj. za svaku danu riječ vraća sve njene moguće prijevode (ako je riječ višeznačna, vjerojatno će imati više od jednog prijevoda).

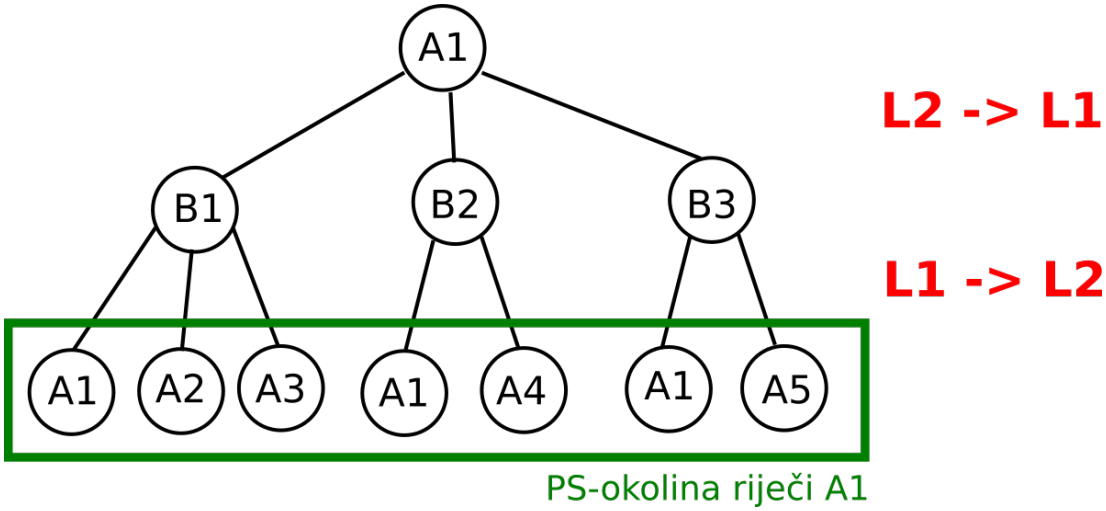
Govornik dok piše tekst u glavi ima riječ A1 jezika L1 te ju želi prevesti i napisati u tekst. Neka su u jeziku L2 mogući prijevodi te riječi B1, B2 i B3. Neka je u danom kontekstu ispravno napisati B1, ali govornik greškom napiše B2. Ako mi tada riječ B2 pošaljemo u rječnik L2->L1, on će nam vratiti skup prijevoda te riječi u jeziku L1 među kojima će biti i riječ A1 (na koju je govornik mislio). Pošaljemo li sve te dobivene riječi jezika L1 u rječnik L1->L2, za svaku od tih riječi dobit ćemo skup njenih prijevoda u jeziku L2. Unija tih skupova sigurno sadrži riječ B1 (jer je to jedan od prijevoda riječi A1 koju smo dobili nakon prvog prevođenja). Tu je riječ govornik trebao napisati umjesto one pogrešne.

Spomenutu uniju skupova koju smo dobili nakon dva prevođenja zvat ćemo *PS-okolina* (polisemija). Važno je naglasiti da, pod pretpostavkom dobrog dvosmjernog rječnika, PS-okolina riječi koja je netočno napisana zbog višeznačnosti sigurno sadrži i točnu riječ, onu koja je trebala biti napisana.

Navedimo dva primjera postupka dobivanja PS-okoline.

Težak Hrvatska riječ *težak* na engleski može biti prevedena kao: *trying*, *difficult*, *hard*, *heavy*, *rough*, *weighty*. Svaka od tih riječi ima jedan ili više prijevoda na hrvatski. Svi ti hrvatski prijevodi svih navedenih engleskih prijevoda čine PS-okolinu riječi *težak*. Konkretno, to su riječi: *težak*, *naporan*, *zamoran*, *tvrd*, *čvrst*, *surov*, *marljiv*, *snažan*, *kršan*, *hrapav*, *grub*, *približan*.

Posao Hrvatska riječ *posao* ima sljedeće engleske prijevode: *job*, *work*, *chore*, *business*, *affair*, *transaction*. Kada se dobiju svi hrvatski prijevodi tih engleskih riječi, dobije se sljedeća PS-okolina: *posao*, *zadatak*, *rad*, *zanimanje*, *pljačka*, *trgovina*, *zaposlenost*, *afera*, *radno mjesto*.



Slika 3.3: Generiranje PS-okoline riječi A1

3.4.2. Rječnik

Iz prethodnog odjeljka jasno je da će jedan od modula u našem rješenju trebati biti i rječnik. Idealno to treba biti dvosmjerni L1/L2 rječnik za koji sigurno vrijedi:

1. Ako se riječ A1 iz L1 prevede u skup riječi B1, B2, itd. jezika L2, onda za svaku od riječi iz prevedenog skupa mora vrijediti da, kada se ona prevede iz jezika L2 u jezik L1, njen skup prevedenih riječi sadrži, između ostalih, i riječ A1;

2. Ako se riječ B1 iz L2 prevede u skup riječi A1, A2, itd. jezika L1, onda za svaku od riječi iz prevedenog skupa mora vrijediti da, kada se ona prevede iz jezika L1 u jezik L2, njen skup prevedenih riječi sadrži, između ostalih, i riječ B1.

Valja napomenuti da je korak pronalaska PS-okoline pomoću L1/L2 rječnika jedini korak u kojem jezik L1, materinji jezik govornika, igra ulogu. Riječi jezika L1 služe samo kao spona između riječi jezika L2 – nigdje se ne spremaju niti se kasnije koriste.

3.4.3. Relacije – kandidati

Kako sada imamo modul koji nam za svaku danu riječ može pronaći njenu PS-okolinu, možemo to iskoristiti pri analizi relacija izvađenih iz teksta. Naime, ideja je da za svaku relaciju dobijemo skup PS-okoline prve riječi i skup PS-okoline druge riječi, te kombinirajući riječi iz ta dva skupa na neki način dobijemo skup relacija. Taj skup predstavlja parove riječi od kojih je jedan par ispravna relacija.

Ako je govornik u tekstu prilikom pisanja neke pridjevsko-imeničke ili glagolsko-objektne sintagme napravio pogrešku leksičkog transfera zbog višeznačnosti, tada će u skupu relacija dobivenih kombinirajući riječi iz PS-okolina riječi napisane relacije biti i ispravna relacija.

Svaka od tih dobivenih relacija (*PS-relacije*) bit će kandidat za ispravnu relaciju kada se bude išlo u ocjenjivanje relacija. Postavlja se pitanje na koji način kombinirati riječi PS-okoline prve riječi (skup PS1) i riječi PS-okoline druge riječi (skup PS2). Razmotriti ćemo dva načina uparivanja: bipartitno uparivanje i kartezijev produkt.

Bipartitno uparivanje

Prvi način generiranja skupova PS-relacija pretpostavlja da je jedna od riječi u napisanoj relaciji ispravna, tj. da govornik nije u obje riječi pogriješio. Na ovaj se način PS-relacije generiraju tako što se jednu od dvije riječi relacije drži nepromijenjenom, a na mjesto druge se redom stavlja sve riječi iz skupa PS-okoline druge riječi.

Na taj način dobiju se dva skupa jer se nepromijenjenom može držati prva riječ dok se druga mijenja po skupu PS2, i obrnuto, da se prva riječ mijenja po skupu PS1 dok se druga drži nepromijenjenom. Nazvali smo ovaj način bipartitno uparivanje.

Ako broj elemenata skupa PS1 iznosi M, a broj elemenata skupa PS2 iznosi N, tada broj elemenata skupa ovako dobivenih PS-relacija iznosi M+N.

Generiranje kandidata na ovakav način oprimjerit ćemo koristeći relaciju *težak posao* za čije smo obje riječi, u odjeljku 3.4.1, pronašli PS-okoline. Neki od generiranih kandidata za tu relaciju bili bi *težak zadatak*, *teško zanimanje*, *težak*

rad, *teška trgovina*, *naporan posao*, *surov posao*, *čvrst posao*, *grub posao*, *snažan posao* i *marljiv posao*. Ono što se može primijetiti je da je u svim primjerima prisutna jedna riječ iz početne relacije, u nekima primjerima riječ *težak*, a u drugim primjerima riječ *posao*.

Kartezijev produkt

Drugi način generiranja skupova PS-relacija pretpostavlja da je govornik mogao pogriješiti u obje riječi pa tako razmatra sve moguće parove riječi iz PS-okolina dviju riječi. Dakle, radi se kartezijev produkt dva skupa.

Ovaj pristup je slobodniji, ali postoji mogućnost da se previše udalji od izvorne relacije, tj. da se dobije smislena relacija koja je potpuno semantički različita od ispravne.

Ako broj elemenata skupa PS1 iznosi M, a broj elemenata skupa PS2 iznosi N, tada broj elemenata skupa ovako dobivenih PS-relacija iznosi M*N.

Generiranje kandidata na ovakav način također ćemo oprimjeriti koristeći relaciju *težak posao*. Neki od generiranih kandidata za tu relaciju bili bi *grub zadatak*, *marljivo zanimanje*, *približna afera*, *tvrda trgovina*, *naporna zaposlenost*, *surova pljačka*, *čvrst rad*, *teška trgovina*, i *marljiv posao*. Ono što se može primijetiti je da se u primjerima pojavljuju i kandidati koji su imaju jednu riječ zajedničku početnoj relaciji (kandidati koji nastaju i bipartitnim uparivanjem) i kandidati kojima su obje riječi drukčije od onih u početnoj relaciji. Već na ovim se primjerima vidi da se ovakvim uparivanjem dobivaju relacije koje su semantički znatno udaljene od početne relacije, npr. relacija *približna afera*.

3.5. Rangiranje kandidata

Sada kad imamo način za dobivanje mogućih kandidata za svaku relaciju, želimo pronaći najboljeg kandidata. To želimo zato što se na tome temelji detektiranje i ispravljanje relacije.

Pretpostavka za pronalazak najboljeg kandidata je da imamo neki način za ocjenjivanje svakog od kandidata. Ocjena bi u ovom slučaju predstavljala ispravnost kandidatne relacije.

Pod uvjetom da smo obavili ocjenjivanje svih kandidata, uključujući izvornu relaciju (budući da je svaka riječ član svoje PS-okoline, tako je i svaka relacija u skupu svojih PS-relacija, neovisno koji od dva načina generiranja kandidata koristimo), možemo potražiti relaciju koja ima najveću ocjenu. Ako to nije izvorna relacija, detektirali smo pogrešku. Pogrešku ispravljamo tako da kao točnu relaciju uzimamo onu s najvećom ocjenom.

Za sve varijante rješenja ovaj dio je zajednički, razlikovat će se samo načini ocjenjivanja pojedine relacije.

3.6. Ocjenjivanje kandidata

3.6.1. Prebrojavanje pojavljivanja

Prvi pokušaj ocjene kandidata bio je pogledati u veliku količinu ispravnog teksta na jeziku L2 i vidjeti koliko se tamo često pojavljuje taj kandidat. Ocjena je upravo broj pojavljivanja. Kandidat koji se najčešće pojavljivao u velikoj količini teksta bit će proglašen ispravnom relacijom. Dakle, za ovaj način potreban je ispravan korpus s velikim brojem riječi na jeziku L2.

Rezultati dobiveni na ovaj način predstavljaju referentnu metodu s kojom možemo uspoređivati naprednije metode i vidjeti jesu li i koliko uspješnije.

3.6.2. Strojno učenje i model vektorskog prostora

Napredniji pokušaj ocjene kandidatnih relacija bio je da se napravi model nadziranog strojnog učenja tipa regresije, koji se može istrenirati na točnim i netočnim relacijama.

Ideja je bila iz ispravnog teksta koji se nalazi u prije spomenutom korpusu izvući određeni broj točnih relacija. Uz svaku relaciju zapamtiti treba i rečenicu u kojoj se ona nalazi. Za svaku relaciju treba generirati njene PS-relacije po prethodno opisanom postupku (odjeljak 3.4.3.). U ovom ćemo slučaju PS-relacije generirati samo postupkom bipartitnog uparivanja.

Razlog zanemarivanja postupka generiranja PS-relacija kartezijevim produktom je njegova neučinkovitost koja se pokazala prilikom testiranja na referentnom

algoritmu. Vrijeme koje smo imali na raspolaganju više smo željeli iskoristiti tako da se posvetimo ocjenjivanju kandidata i testiranju većeg broja modela stroja potpornih vektora, pa smo za tu metodu kao postupak generiranja kandidata koristili samo onu koja se pokazala efikasnijom na referentnom algoritmu – bipartitno uparivanje. Ipak, kako su generiranje kandidata i ocjenjivanje kandidata potpuno razdvojeni, ortogonalni pojmovi, smislenije bi bilo testiranje provesti nad sve 4 kombinacije generiranja i ocjenjivanja kandidata. Svakako bi u budućonsti bilo dobro da se to provede.

Sve PS-relacije osim izvorne su netočne, te ćemo ih zato tako tretirati. Za danu relaciju uzimamo rečenicu u kojoj se ona nalazi (bez same relacije). To predstavlja okolinu, kontekst relacije. Zatim treniramo model tako da u tu okolinu umjesto izvorne relacije stavljamo generirane, netočne relacije i to dajemo kao primjer netočne relacije modelu (Y vrijednost koju dajemo modelu za takve primjere je 0). Okolinu u koju smo umetnuli izvornu (i ispravnu) relaciju (to je zapravo cijela rečenica u kojoj se relacija pojavila u tekstu) dajemo kao primjer točne relacije modelu (Y vrijednost koju dajemo modelu za takve primjere je 1).

Da bismo modelu strojnog učenja mogli predati neku rečenicu, ta rečenica treba biti predstavljena kao vektor brojeva. Da bismo rečenicu mogli predstaviti kao vektor brojeva, iskoristit ćemo model vektorskog prostora koji su opisali Turney i Pantel (2010). Svaka riječ koja se može pojaviti ima svoju dimenziju u vektorskom prostoru. Vektorski prostor ima onoliko dimenzija koliko ima različitih riječi (lema). Svaki dio teksta, pa tako i rečenica, tada se može predstaviti kao vektor, koji na svakom mjestu ima broj koji govori koliko se puta riječ koju to mjesto predstavlja pojavila u rečenici.

Prilikom detektiranja i ispravljanja na ovaj način, opet će se prvo pronaći PS-relacije za zadanu relaciju, svakoj će model dati određenu ocjenu (*score*), te će se kao ispravna uzeti ona s najvećom ocjenom. Ako je zadana relacija ona s najvećom ocjenom, onda nema pogreške, a ako je relacija s najvećom ocjenom neka od generiranih kandidata, tada se zadana relacija detektira kao greška i ispravlja u relaciju s najvećom ocjenom.

3.6.3. Stroj potpornih vektora

Metoda stroja potpornih vektora (engl. *support vector machine*) nastala je kao metoda klasifikacije. Trenutni uobičajeni oblik (koji koristi meku marginu) nastao je 1995. godine (Cores i Vapnik, 1995.). Te iste godine, metoda je proširena na regresiju i nastala je metoda regresije potpornim vektorima.

Kada govorimo o klasifikaciji, stroj s potpornim vektorima kao ulaz prima n-dimenzionalne vektore (u našem slučaju, n će biti jednak ukupnom broju različitih lema) i za svaki vektor prima jednu od dvije klase kojoj taj vektor pripada. Zatim, u tom n-dimenzionalnom prostoru stroj pokušava pronaći hiperravninu koja najbolje razdvaja vektore (točke u n-dimenzionalnom prostoru) jedne klase od vektora druge klase.

Pretpostavimo najprije da su ulazni podaci takvi da su vektori jedne klase potpuno linearno razdvojivi od vektora druge klase. To znači da se u prostoru može konstruirati ravnina koja s jedne svoje strane ima isključivo vektore jedne klase, a s druge strane isključivo vektore druge klase. Svaka od zadanih točaka na nekoj je udaljenosti d od konstruirane ravnine. Gledajući udaljenosti točaka samo jedne od dviju klasa od ravnine, najmanja od svih tih udaljenosti predstavlja udaljenost ravnine od skupa točaka te klase. Isto vrijedi i za udaljenosti točaka druge klase od ravnine – najmanja od svih tih udaljenosti predstavlja udaljenost ravnine od skupa točaka druge klase. Ova dva broja – najmanja udaljenost neke točke prve klase od ravnine i najmanja udaljenost neke točke druge klase od iste ravnine – brojevi su čiji zbroj daje marginu. Dakle, margina je zbrojena najmanja udaljenost skupa točaka svake od klasa od ravnine.

Ravnina koja razdvaja dvije klase točaka ne mora biti samo jedna, nego ih može biti mnogo različitih. Svaka od njih ima svoj iznos margine. Zadatak je stroja s potpornim vektorima da pronađe ravninu s najvećom marginom. Ravnina koja je najudaljenija od skupova točaka obje klase najbolje razdvaja te skupove. Postoji matematički dokaz[1] da je margina obrnuto proporcionalna normi vektora koeficijenata ravnine, pa zapravo stroj s potpornim vektorima teži minimizaciji kvadrata norme vektora koeficijenata ravnine.

[1] `http://ce.sharif.ir/courses/85-86/2/ce725/resources/root/LECTURES/SVM.pdf`

Meka margina

Ako je skup podataka velikim dijelom linearno razdvojiv, ali postoje neke rijetke točke koje su se našle sa suprotne strane ravnine od one na kojoj je većina njihove klase, tada bismo još uvijek željeli da takva ravnina bude konstruirana, te se uvodi pogreška kod klasifikacije. Za sve točno klasificirane točke, pogreška je 0, a za netočno klasificirane točke je pogreška jednaka udaljenosti od ravnine (uvijek pozitivna vrijednost). Tada se zbroj tih pogrešaka za danu ravninu pomnožen nekim regulacijskim faktorom dodaje prethodno izračunatom kvadratu norme vektora koeficijenata ravnine, te se taj izraz nastoji minimizirati. Ova metoda zove se metoda meke margine.

Transformacija jezgrenom funkcijom

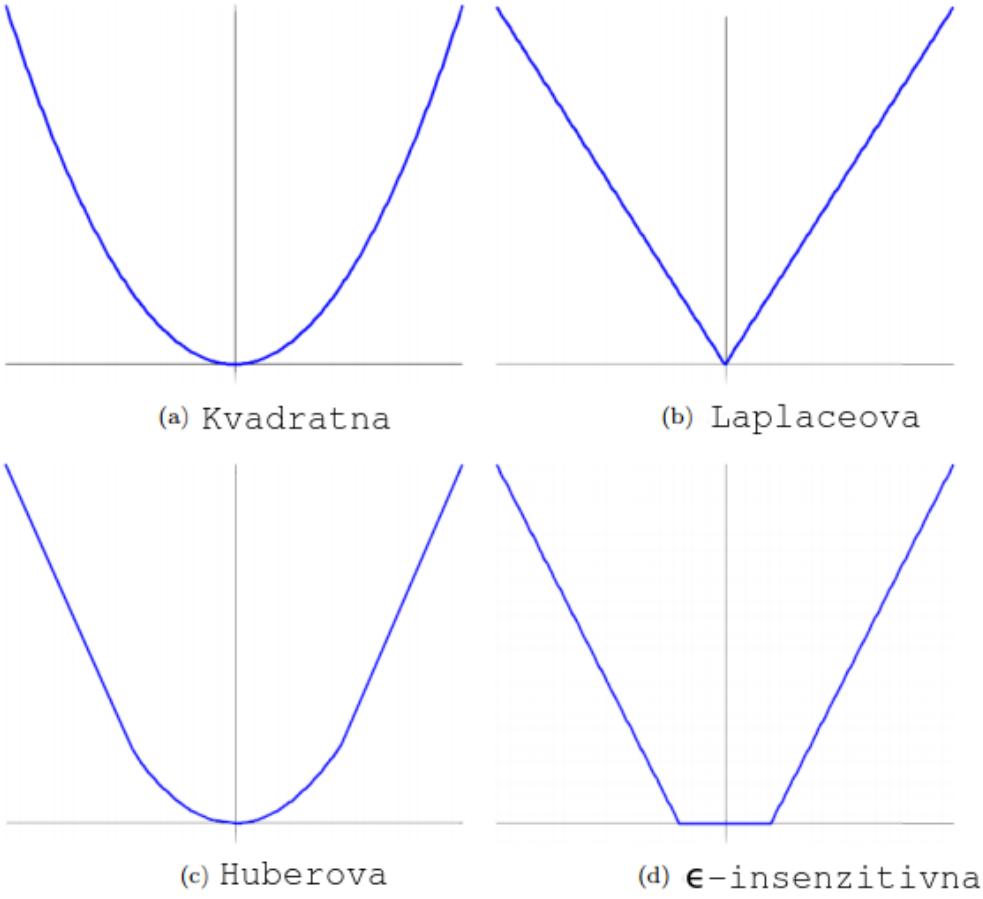
Ako je su točke obiju klasa toliko nerazdvojive da linearna granica (ravnina) nema smisla, nego je potrebna nekakva nelinearna ploha da točno razdvoji klase, tada se treba koristiti trik koji inače služi kada želimo da linearni algoritam učenja nauči nelinearnu granicu. Sve vektore pomnoži se nekom transformacijskom (jezgrenom) funkcijom koja podigne te vektore u neku višu dimenziju. Velika je vjerojatnost da će u toj višoj dimenziji vektori biti linearno razdvojivi, tj. moći će se konstruirati ravnina koja točno razdvaja vektore.

Regresija potpornim vektorima

Regresija potpornim vektorima ima neke sličnosti s prethodno opisanom klasifikacijom. Cilj regresije s potpornim vektorima je također pronaći optimalnu ravninu. Optimalna ravnina u ovom slučaju je ona koja što bliže prolazi svim zadanim točkama. Kao i kod klasifikacije, opet se nastoji minimizirati kvadrat norme vektora koeficijenata ravnine, ali uz uvjet da udaljenost svih vektora od ravnine bude unutar neke dozvoljene pogreške.

Kako se kod klasifikacije kažnjava postojanje vektore koji su s krive strane ravnine, tako se i kod regresije kažnjava vektore koji su izvan dozvoljene pogreške. Tome služi funkcija gubitka (*loss function*) koja raste kako se udaljenost od ravnine povećava. Funkcije gubitka mogu biti različite. Na slici 3.4[2] prikazane su 4 moguće takve funkcije (funkcija pod d) upravo je ova prethodno opisana, koja ne kažnjava vektore unutar nekog intervala).

[2]Preuzeto sa: `http://ce.sharif.ir/courses/85-86/2/ce725/resources/root/LECTURES/SVM.pdf`



Slika 3.4: Primjeri funkcija gubitka

Također, kao i kod klasifikacije, ako se ulazni podaci ne mogu dobro aproksimirati linearnom plohom, koristi se isti trik – vektori se množi jezgrenom (kernel) funkcijom koji ih podiže u višu dimenziju gdje postoji ravnina koja dobro aproksimira sve vektore.

3.6.4. Predviđanje

Jednom kada je stroj s potpornim vektorima pronašao zadovoljavajuću hiperravninu, predviđanje izlaza za nove ulazne podatke radi se tako da se ulazni vektor ubaci u formulu hiperravnine i izračuna rezultat.

4. Implementacija

Kada je model bio osmišljen, trebalo je napisati odgovarajuće programsko rješenje. Implementacija je napisana u programskom jeziku Python 2.7. Python je odabran zbog velikog broja gotovih paketa koji sadrže module potrebne za implementaciju našeg rješenja i zbog jednostavnosti uporabe.

4.1. Korišteni alati

Za različite korake u implementaciji korišteni su različiti moduli. Ovo je popis korištenih modula i biblioteka:

- WordNetLemmatizer: lematizacija[1]
- Stanford CoreNLP: parsanje rečenica i vađenje relacija[2]
- BingTranslator Python API: implementacija L1/L2 rječnika[3]
- NLTK CorpusReader: pristup korpusu[4]
- scikit-learn Support Vector Regression: strojno učenje[5]
- pickle: pohranjivanje objekata (rječnika, istreniranih modela, itd.)[6]

4.2. Stanford CoreNLP i WordNetLemmatizer

Ključan korak u rješavanju problema je prelazak iz ulaznog neobrađenog teksta u parsane rečenice u kojima su označene sve relacije ovisnosti između riječi u svakoj rečenici. Upravo za taj zadatak zadužen je Stanford CoreNLP – skupa alata za obradu jezika među kojma se nalazi i ovisnosni parser. CoreNLP napisan

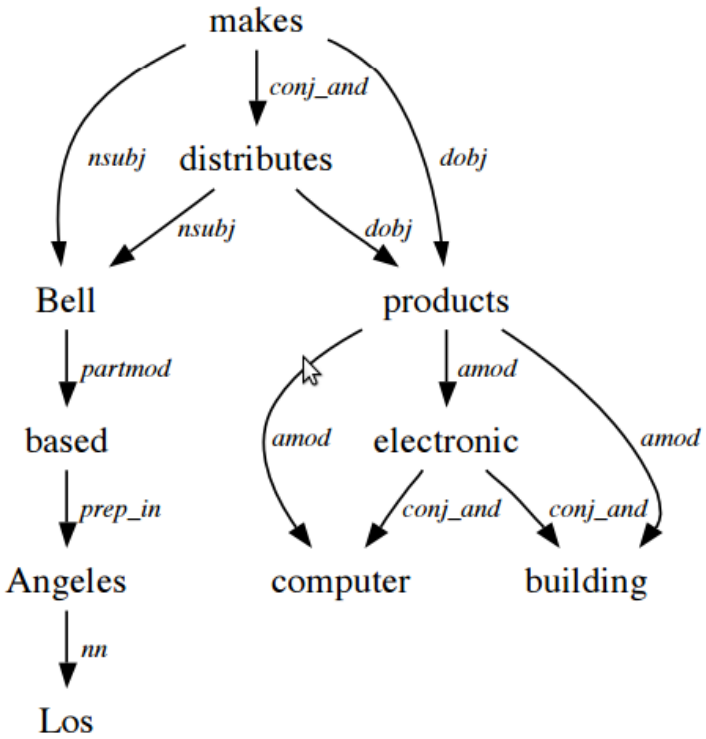
[1] `www.nltk.org/_modules/nltk/stem/wordnet.html`
[2] `http://stanfordnlp.github.io/CoreNLP/`
[3] `https://pypi.python.org/pypi/BingTranslator/0.1`
[4] `http://www.nltk.org/_modules/nltk/corpus/reader/bnc.html`
[5] `http://scikit-learn.org/stable/modules/svm.html#regression`
[6] `https://docs.python.org/2/library/pickle.html`

je u u programskom jeziku Java, ali postoji Python omotač pod nazivom corenlp-python, koji omogućuje pokretanje parsera unutar skripti u Pythonu.

Gramatička struktura rečenice parsane ovisnosnim parserom prikazana je kao stablo ovisnosti koje se širi od jedne riječi (korijena) prema ostalim riječima do listova. Svaka riječ u stablu povezana je s nekom drugom riječi relacijom ovisnosti. U svakoj relaciji ovisnosti jedna je riječ glavna (engl. *head*), a druga ovisna (engl. *dependent*) o glavnoj.

Postoji mnogo različitih tipova relacija ovisnosti koje Stanfordov parser može prepoznati, ali su nas za ovaj rad zanimala samo dva tipa: pridjevsko-imenički (parser koristi kraticu *amod*) i glagolsko-objektni (parser koristi kraticu *dobj*).

Za primjer, na slici je prikazano stablo ovisnosti nakon parsanja rečenice: *Bell, based in Los Angeles, makes and distributes electronic, computer and building products.*



Slika 4.1: Stablo ovisnosti, izvor: Stanford typed dependencies manual (de Marneffe i Manning)

Vidimo da se rečenica sastoji i od ovisnosti tipa *dobj* ("distributes products") i od ovisnosti tipa *amod* ("electronic product"). Ovo su primjeri relacija koje smo željeli izvući iz tekstova i analizirati.

4.2.1. Format CoNLL

Parserom se stvorena stabla ovisnosti mogu prikazati u formatu CoNLL. To je format zapisa koji ima onoliko redaka koliko ima riječi u rečenici i nekoliko stupaca (ovisno o verziji formata). Format CoNLL korišten u ovom radu ima 10 stupaca. Svaka riječ ima svoj redak u ispisu, a oni idu redoslijedom kojim idu riječi u rečenici. Stupci su sljedeći:

- ID je redni broj riječi u rečenici;
- FORM je sama riječ;
- LEMMA je lematizirani oblik riječi;
- CPOSTAG je gruba oznaka vrste riječi;
- POSTAG je preciznija oznaka vrste riječi;
- FEATS označava skup sintaksnih ili morfoloških svojstava riječi;
- HEAD je ID riječi prema o kojoj je ova riječ ovisna. HEAD je za riječ koja je korijen stabla ovisnosti jednak 0 (jer ta riječ nije ovisna ni o jednoj riječi u rečenici);
- DEPREL je tip relacije ovisnosti ove riječi i riječi koji ima ID jednak HEAD. DEPREL je za riječ koja je korijen stabla ovisnosti jednak 'ROOT' (jer ta riječ nije ovisna ni o jednoj riječi u rečenici);
- PHEAD Projektivna glava trenutne riječi. Uvijek prazno mjesto u ispisu Stanford Parsera;
- PDEPREL Relacija ovisnosti prema riječi PHEAD trenutne riječi. Uvijek prazno mjesto u ispisu Stanford Parsera.

Jednom kada se rečenica parsa u format CoNLL, jednostavno je napisati algoritam koji će iz nje izvući dane tipove relacija (u našem slučaju *dobj* i *amod*).

4.2.2. Razred `Relation`

Razred koji predstavlja jednu relaciju iz teksta nazvan je `Relation`. Svaki objekt tipa `Relation` u sebi sadrži lematiziranu glavnu riječ, lematiziranu ovisnu

riječ, tip relacije, ID-eve glavne i ovisne riječi u rečenici, broj rečenice iz koje je relacija izvučena u dokumentu i ime dokumenta u kojem se rečenica nalazi. Tako je jednostavno iz relacije koja nam je zanimljiva i čiji nam je kontekst potreban dobiti cijelu rečenicu u kojoj se relacija pojavila.

4.2.3. WordNetLemmatizer

Da bi se moglo kvalitetno pretvoriti riječi iz teksta u leme koristi se WordNetLemmatizer koji je dio Natural Language Toolkita – skupa alata za obradu prirodnog jezika. Metoda lemmatize prima riječ i jednostavnu oznaku vrste riječi, te pomoću toga pronalazi i vraća lemu za danu riječ.

4.3. BingTranslator i implementacija rječnika

Kada se dođe do koraka analize izvučenih relacija, važan korak je pronalazak PS-okoline (skupa riječi jezika L2 koji su preko jezika L1 povezani s danom riječi iz jezika L2) za dane riječi. U poglavlju 3 već je rečeno da je za to potreban L1/L2 dvosmjerni 1-na-n rječnik.

Microsoft omogućava uporabu svog Translator API-ja besplatno do 2 milijuna prevedenih znakova mjesečno. Kako za Python postoji biblioteka BingTranslator koja nudi funkcije prevođenja riječi i kako u njoj postoji funkcija koja vraća sve dostupne prijevode za danu riječ, upravo smo tu biblioteku i taj Translator API koristili kao rječnik. L1 bio je kineski, a L2 engleski jezik.

Uz Translator, koristili smo i jedan popis engleskih riječi povezanih preko višeznačnosti kineskih riječi.[7]

Konačni generator PS-okoline implementiran je kao struktura podataka rječnik koji za danu riječ (ključ) vraća skup riječi (value) koje čine PS-okolinu dane riječi.

4.4. NLTK Corpus Reader i BNC

Kako smo kao jezik L2 odabrali engleski jezik, za implementaciju zamišljenog modela potreban je bio korpus gramatički ispravnih tekstova na engleskom

[7]Popis riječi ustupila je prof. Marina Dodigović sa Sveučilišta Xi'an Jiaotong-Liverpool University (XJTLU) u Kini.

jeziku. Odlučili smo se za British National Corpus koji sadržava kolekciju engleskih tekstova veličine oko 100 milijuna riječi. Da bi se dobio pristup tom korpusu, potrebno je prijaviti se za preuzimanje korpusa na stranici Sveučilišta u Oxfordu.

Korpus se sastoji od velike količine XML-datoteka koje za svaku riječ teksta dodatno sadržavaju i njenu lematiziranu verziju i oznaku vrste riječi. Jednostavan način za pristupanje korpusu kroz Python je putem modula BNCCorpusReader, koji je dio Natural Language Toolkita. To je sučelje za pristup XML-verziji British National Corpusa. Objekt BNCCorpusReader se inicijalizira putanjom do direktorija u kojem se nalaze tekstovi korpusa, te regularnim izrazom koji određuje koje će od svih mogućih datoteka korpusa taj reader čitati.

Kako je za ovaj zadatak bilo potrebno brzo pretraživanje korpusa (za pronalazak svih pojavljivanja relacija u korpusu), napisan je razred CorpusInterface koji se koristi BNCCorpusReaderom i iz korpusa uzima sve parove riječi, te ih sortira po prvoj i po drugoj riječi da omogući binarno pretraživanje po parovima riječi, te time ubrza pronalaženje pojavljivanja relacija.

4.5. scikit-learn i Support Vector Regression

Paket scikit-learn skup je Python alata za analizu podataka. U njemu se nalaze implementacije mnogih različitih metoda nadziranog i nenadziranog strojnog učenja. Kod implementiranja drugog modela detekcije i ispravljanja netočnih relacija bilo nam je potrebno nadzirano strojno učenje tipa regresije. To je stoga što kao izlaz za svaku relaciju koju damo modelu želimo nekakav broj (ocjenu) koji možemo uspoređivati s ocjenama ostalih relacija i tražiti onu s najvišom ocjenom.

Scikit-learn nudi razne algoritme regresije, a za ovu implementaciju korištena je metoda stroja s potpornim vektorima za slučaj regresije. Support Vector Regression u biblioteci scikit-learn ima tri različite implementacije (SVR, NuSVR, LinearSVR), a za naše rješenje odabran je razred SVR.

4.6. Pickle

Pickle je alat koji služi za serijalizaciju i deserijalizaciju objekata. U ovom programskom rješenju korišten je za spremanje napunjenih rječnika, listi relacija,

istreniranih modela strojnog učenja i objekata ostalih tipova.

4.7. Ostali razredi

Ovdje su navedeni i ukratko opisani ostali bitni razredi koji do sada nisu bili spomenuti.

`RelationCorrection` Razred zadužen za ispravljanje relacije naivnim referentnim algoritmom, s oba načina generiranja PS-relacija za danu relaciju. Iz dane relacije i objekta tipa CorpusInterface metoda correctRelation vraća ispravljenu relaciju (ili danu relaciju, ako je ista ocijenjena ispravnom).

`MLCorrection` Razred zadužen za ispravljanje relacija algoritmom strojnog učenja. Metoda correct_relation za danu relaciju i kontekst (ostatak rečenice) vraća ispravljenu relaciju (ili danu relaciju, ako je ista ocijenjena ispravnom).

`POSLookup` Razred zadužen za određivanje vrste riječi u slučajevima kada se dogodi da ona nije poznata. Koristi korpus (koji ima označene vrste riječi) i gleda koja je najčešća oznaka za danu riječ i tu oznaku uzima kao točnu.

`utils.py` Nije razred, nego skupina često korištenih metoda. Tu su prisutne metode za otvaranje datoteka za pisanje ili čitanje, serijalizaciju i deserijalizaciju objekata, stvaranje rječnika, slanje zahtjeva prevođenja BingTranslatoru i slično.

5. Vrednovanje

Testiranje svih modela provodi se nad jednim dijelom teksta iz Britanskog nacionalnog korpusa (BNC). Taj dio nije dio korpusa koji prvi model s referentnim algoritmom koristi za pretraživanje pojavljivanja relacija. Referentni algoritam za pretraživanje koristi prvih 500 dijelova teksta korpusa koji se sastoji od oko 10 milijuna riječi. Svi tekstovi su iz dijela označenog slovom A. Tekst nad kojim se provodi testiranje nalazi se u dijelu korpusa označenom slovom K, pod nazivom K1B.xml. Dio koji je korišten za testiranje sastoji se od od oko 15000 riječi.

U tom dijelu teksta postoji 1000 nama zanimljivih relacija. Među njima ima 673 *amod* relacija, te 327 *dobj* relacija.

Relacije ekstrahirane iz korpusa smatraju se ispravnima. Neispravne ćemo relacije generirati koristeći PS-okolinu i dobivanje skupa PS-relacija neke zadane relacije. Postupak dobivanja PS-relacija opisan je u odjeljku 3.4. Za neku zadanu relaciju, sve relacije koje dobijemo tim postupkom, a koje nisu jednake zadanoj relaciji, tretiramo kao neispravne.

Generiranjem neispravnih relacija putem PS-okoline simulira se neispravan tekst L2-govornika. Motivacija za ovakav način dobivanja neispravnih relacija je i to da se tada ne treba ručno označavati (ispravljati) tekstove.

Cilj je svih modela da sve točne relacije prepoznaju kao točne, te ih ostave nepromijenjenima, a da sve netočne relacije isprave u točne. Točnost modela ocjenjuje se vrlo strogo. Za svaku od 1000 relacija gleda se ona sama i njen skup PS-relacija. Model treba funkciju ispravljanja primijeniti na točnu relaciju i na sve relacije iz skupa PS (netočne relacije). Da bi bio ocijenjen točnim, model mora točnu relaciju ostaviti nepromijenjenom i sve netočne relacije ispraviti baš u točnu relaciju. Pogriješi li s ispravljanjem i na jednoj od netočnih relacija

(kojih ima u prosjeku 20 puta više od točnih) model se ocjenjuje netočnim na tom primjeru. Također, ako model odluči da je točna relacija netočna, te ju promijeniti u neku drugu, opet se ocjenjuje netočnim.

Ako je R broj točnih relacija (od kojih svaka ima svoj skup PS-relacija) koje se analiziraju, a N broj relacija koje su točno ispravljene, tada je točnost T jednaka:

$$T = \frac{N}{R} \cdot 100\%$$

U sklopu ovog rada, vrednovana su tri različita modela:

1. Referentni algoritam s bipartitnim generiranjem kandidata,
2. Referentni algoritam s generiranjem kandidata putem kartezijevog produkta,
3. Algoritam strojnog učenja koji koristi stroj potpornih vektora.

5.1. Referentni algoritam

Ovaj algoritam radi tako da za zadanu relaciju u korpusu prebrojava njena pojavljivanja i pojavljivanja njenih PS-relacija, te ako se zadana relacija ne pojavljuje najviše puta, ispravlja tu relaciju u onu koja se pojavljuje najviše puta.

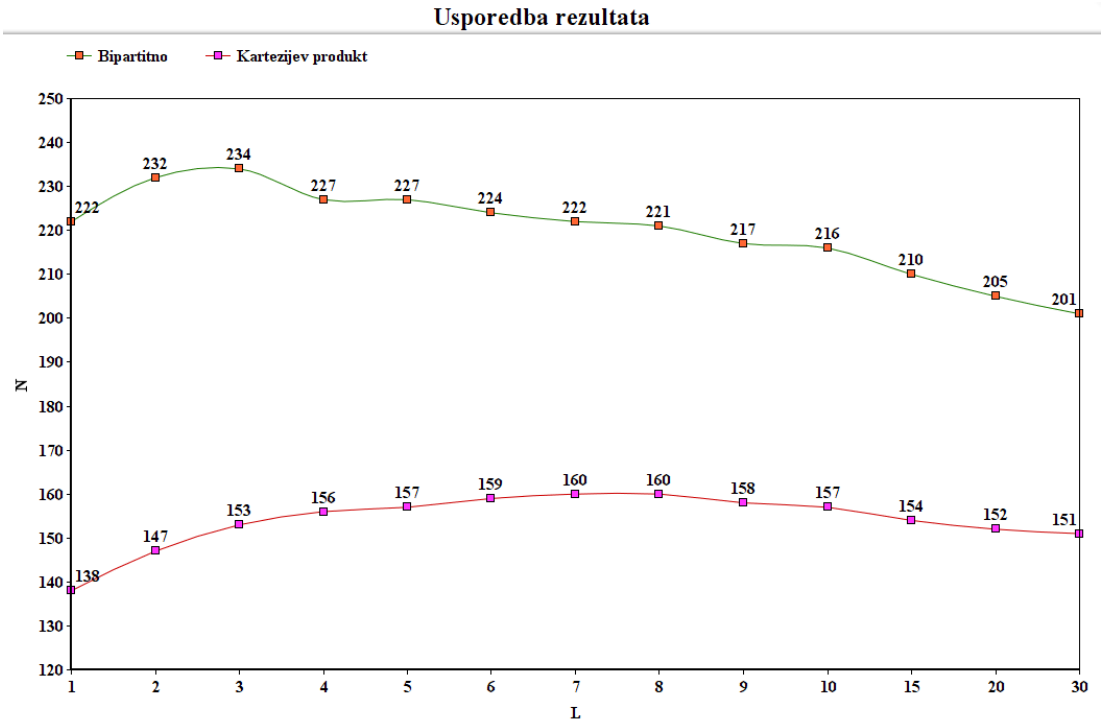
Ipak, u slučajevima kada je broj pojavljivanja svih tih relacija jako malen pa je i broj pojavljivanja čak i one relacije koja se najviše pojavljuje malena brojka, možda ne bismo htjeli ispravljati relaciju. Zato uvodimo parametar L kojim konfiguriramo model. Ako je nakon provedenog prebrojavanja pojavljivanja relacija broj pojavljivanja relacije koja se najčešće pojavljuje manji od broja L, relacija se ostavlja nepromijenjenom. Vrednovanje je provedeno nad raznim vrijednostima parametra L.

Rezultati za referentni algoritam s bipartitnim generiranjem kandidata, te za referentni algoritam s generiranjem kandidata putem kartezijevog produkta prikazani su u tablici 5.1. Tbi označava točnost prvog, a Tkp točnost drugog algoritma u postocima.

L	1	2	3	4	5	6	7	8	9	10	15	20	30
Tbi	22.2	23.2	23.4	22.7	22.7	22.4	22.2	22.1	21.7	21.6	21	20.5	20.1
Tkp	13.8	14.7	15.3	15.6	15.7	15.9	16	16	15.8	15.7	15.4	15.2	15.1

Tablica 5.1: Rezultati vrednovanja

Na slici 5.1. nalazi se graf na kojem su prikazani rezultati vrednovanja referentnog algoritma s oba načina generiranja kandidatnih relacija za razne vrijednosti parametra L.



Slika 5.1: Graf usporedbe prva dva algoritma

5.2. Algoritam sa strojnim učenjem

Da bi se model strojnog učenja mogao evaluirati, treba ga najprije istrenirati. Iz tog razloga, uobičajeno je jedan (veći) dio skupa podataka iskoristiti za učenje modela, a ostatak za testiranje. U našem slučaju, za učenje modela uzeli smo 70% od ukupnih ulaznih podataka (700 točnih relacija i njihove netočne PS-relacije), a na preostalih 30% smo model testirali.

U implementaciji koristimo razred SVR koji se nalazi u modulu sklearn.svm. Slobodni argumenti konstruktora razreda SVR koji mogu varirati su C i ϵ. C predstavlja regulacijski faktor uz izraz koji označava pogrešku, dok ϵ određuje širinu intervala [-ϵ, ϵ] u kojem se greška ne dodjeljuje. Ako ih ne zadamo, pretpostavljene vrijednosti parametra C i ϵ su 1, odnosno 0.1. Kao što se u rezultatima vrednovanja vidi, varijacijom parametra C u odnosu na pretpostavljenu vrijednost postigli su se značajno bolji rezultati. Variranje parametra ϵ nije ni u jednom slučaju izmijenilo rezultat.

5.2.1. Preciznost, odziv, točnost. F1-mjera.

Osim točnosti (kako smo prije opisali), prilikom evaluacije možemo pratiti i neke druge brojke. Budemo li za svaku od relacija pratili je li nakon testiranja bila ispravljena ili nije i uzmemo li u obzir je li ta relacija točna ili nije, dobit ćemo 4 moguća slučaja koja se mogu dogoditi:

1. **True positive (TP)** Relacija je točna i model ju je prepoznao kao točnu (nije ju mijenjao);
2. **True negative (TN)** Relacija je netočna, model ju je prepoznao kao netočnu i ispravio ju u točnu;
3. **False positive (FP)** Relacija je netočna, ali ju je model prepoznao kao točnu (nije ju mijenjao);
4. **False negative (FN)** Relacija je točna, ali ju je model prepoznao kao netočnu i promijenio ju.

Za svaku evaluaciju modela pratimo koliko ima primjera za koji slučaj, te pomoću tih brojki možemo izračunati neke zanimljive statističke značajke koje nam onda mogu poslužiti u uspoređivanju modela.

Preciznost Preciznost (engl. *precision*) nam govori koliki je dio primjera koji su odabrani kao točni stvarno točan. Formula za preciznost glasi:

$$P = \frac{TP}{TP + FP}$$

Odziv Odziv (engl. *recall*) nam govori koliki je dio primjera koji je stvarno točan odabran kao točan. Formula za odziv glasi:

$$R = \frac{TP}{TP + FN}$$

Točnost Točnost (engl. *accuracy*) nam govori koliki je omjer broja svih istinitih rezultata (TP i TN) i ukupnog broja testnih primjeraka. Ovo ne treba miješati s našom prethodno definiranom točnosti koju smo označavali s T. Formula za točnost glasi:

$$A = \frac{TP + TN}{TP + TN + FP + FN}$$

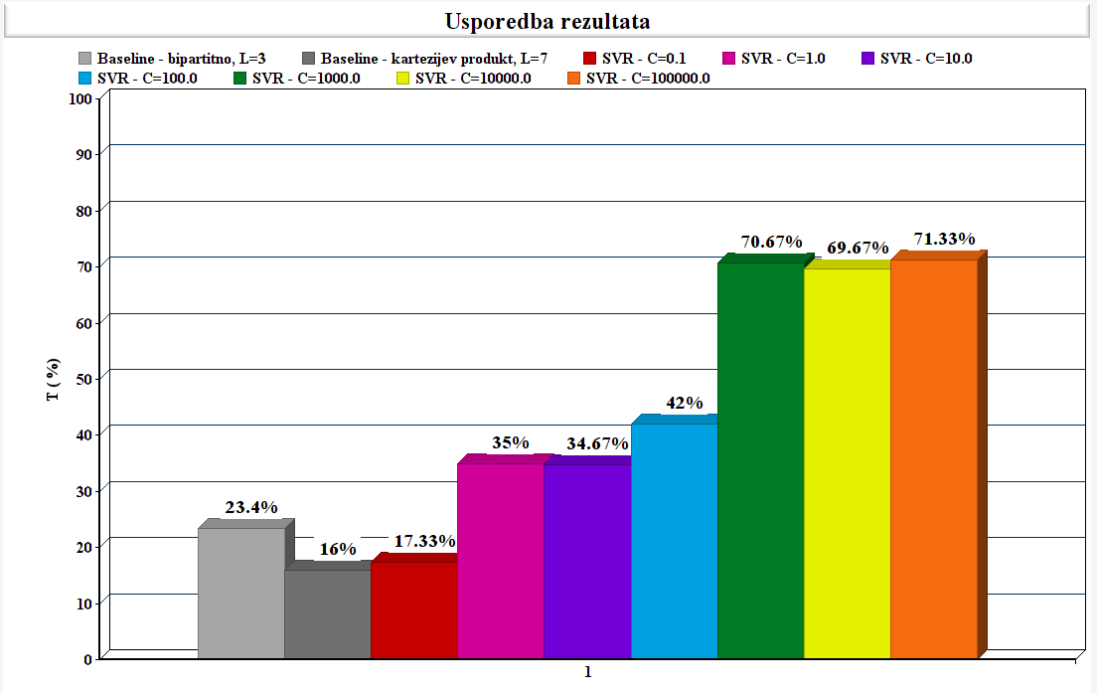
F1-mjera F1-mjera (engl. *F1-score*) još je jedna statistička mjera točnosti nekog testa. Formula za računanje ocjene F1 score glasi:

$$F1 = 2 \cdot \frac{P \cdot R}{P + R}$$

U tablici 5.2. prikazani su rezultati za različite modele stroja s potpornim vektorima koji se razlikuju u regulacijskom faktoru (parametru C). Kao i prije, slovo N označava broj točnih relacija (od ukupnih 300), T označava postotak točnosti kako je na početku poglavlja opisano. Navedeni su i podaci o prethodno pojašnjenim pojmovima (TP, TN, FP, FN, P, R, A, F1).

C	0.1	1	10	100	1000	10000	100000
N	52	105	104	126	212	209	214
T	17.33	35	34.67	42	70.67	69.67	71.33
TP	149	242	250	255	283	281	283
TN	2888	4635	4846	4991	5604	5593	5607
FP	2912	1165	954	809	196	207	193
FN	151	58	50	45	17	19	17
P	0.0487	0.172	0.2076	0.2397	0.591	0.5758	0.5945
R	0.4967	0.8067	0.833	0.85	0.9433	0.9367	0.9433
A	0.4979	0.7995	0.8354	0.86	0.9651	0.963	0.9656
F1	0.0887	0.2835	0.3324	0.3739	0.7267	0.7132	0.7293

Tablica 5.2: Rezultati vrednovanja trećeg algoritma



Slika 5.2: Točnosti modela za različite parametre na prvom skupu podataka

5.3. Konačna usporedba

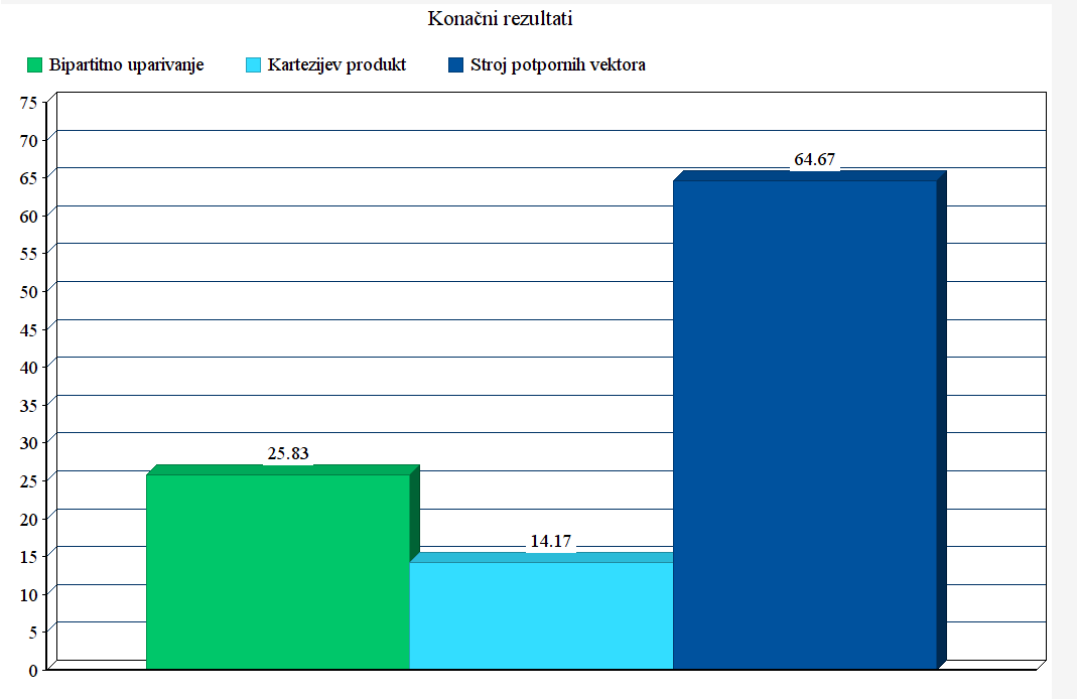
Za sva tri modela pronašli smo optimalne parametre. Za referentni algoritam s bipartitnim uparivanjem optimalna vrijednost parametra L jednaka je 3. Za referentni algoritam s uparivanjem putem kartezijevog produkta optimalna vrijednost parametra L jednaka je 7. Za algoritam koji koristi strojno učenje, optimalna vrijednost parametra C za regresiju potpornim vektorima je 100 000.

Ono što sad treba napraviti je usporediti ta tri modela na novom skupu relacija. Taj će skup relacija biti ekstrahiran iz dijela teksta datoteke korpusa pod nazivom K8S.xml. Ekstrahirano je 600 relacija, među kojima je njih 366 tipa *amod*, a 244 tipa *dobj*.

Rezultati su prikazani u tablici 5.3. Slovo T označava postotak točnosti koju smo mi definirali, kao i u prethodnim tablicama.

Metoda	Bipartitno	Kartezijev produkt	Stroj potpornih vektora
T	25.83	14.17	64.67
P	0.088	0.044	0.47
R	0.633	0.45	0.91
A	0.664	0.49	0.93
F1	0.155	0.08	0.62

Tablica 5.3: Usporedba sva tri modela



Slika 5.3: Usporedba sva tri modela

5.4. Primjer na stvarnom tekstu

Nakon što smo usporedili rezultate i zaključili da je metoda regresije potpornim vektorima najuspješnija, prikazat ćemo kako ta metoda radi na jednom odlomku teksta iz korpusa. Sve relacije u tekstu su ispravne tako da bi, u idealnom slučaju, nakon provedbe algoritma tekst trebao ostati nepromijenjen.

Tekst se sastoji od nekoliko gramatički ispravnih engleskih rečenica. Na prvoj je slici prikazan tekst prije pokretanja algoritma. U tekstu su bojama označene ekstrahirane relacije. Relacije tipa *dobj* označene su plavom bojom, dok su relacije tipa *amod* označene narančastom bojom.

Second bomb blast in Ulster.
The second bomb blast in Ulster in twenty-four hours has caused millions of pounds of damage and several minor injuries .
Lamont and Smith clash over taxes.
The Chancellor, Norman Lamont, has said the average taxpayer would have to pay an additional thousand pounds every year under a Labour government.
The Shadow Chancellor, John Smith, denied the Conservative claims , pledging to keep the rate at twenty-five pence.
Ousted Georgian President seeks asylum.
The ousted President of Georgia, Zvlad Gamsakhurdia, has fled to Azerbaijan as rebel leaders in Georgia plan a new regime.

Na tekstu smo proveli detekciju i ispravljanje odabranom metodom. Na drugoj je slici prikazan tekst nakon ispravljanja. Zelenom su bojom označene one relacije koje je metoda točno detektirala kao ispravne (pa ih nije promijenila). Crvenom su bojom označene one relacije za koje je metoda napravila pogrešku i utvrdila da su neispravne, te ih pokušala ispraviti.

Second bomb bomb in Ulster.
The second bomb bomb in Ulster in twenty-four hours has caused wan of pounds of damage and several minor injuries .
Lamont and Smith clash over taxes.
The Chancellor, Norman Lamont, has said the average taxpayer would have to pay an additional thousand pounds every year under a Labour government.
The Shadow Chancellor, John Smith, denied the keep claims , pledging to keep the rate at twenty-five pence.
Ousted Georgian President seeks asylum.
The ousted President of Georgia, Zvlad Gamsakhurdia, has fled to Azerbaijan as rebel leaders in Georgia plan a new regime.

6. Zaključak

Pogreške leksičkog transfera javljaju se najčešće kod govornika koji se služe jezikom koji im nije materinji i koji nisu potpuno usvojili (L2-govornici). Pravila, koncepti i izrazi dobro usvojenog materinjeg jezika u umovima L2-govornika miješaju se s jezikom koji uče. Često se zato događa da L2-govornici u drugi jezik izravno prenose izraze koji su točni u njihovom materinjem jeziku, ali u tom drugom jeziku nisu. Tako nastaju pogreške leksičkog transfera.

Iznimno velikom broju ljudi engleski jezik nije materinji pa bi kvalitetan sustav za detekciju i ispravljanje leksičkog transfera mogao biti važan u sprječavanju degradacije govorenog engleskog jezika diljem svijeta.

Ovaj rad opisuje sustav za automatizirano otkrivanje i ispravljanje grešaka leksičkog transfera, od opisa problema, preko osmišljenog modela do konkretne implementacije u jeziku Python i vrednovanja na skupu podataka.

Prije nego se krenulo sa strojnim učenjem, osmišljen je jednostavniji model koji se temelji na učestalosti pojavljivanja izraza u uzorku teksta. Kao što se vidi iz rezultata, rad takvog modela očekivano je loš. Taj model i služi kao nekakav jednostavni prototip u odnosu s kojim se uspoređuju naprednije metode.

Model koji koristi strojno učenje pomoću stroja s potpornim vektorima daje bolje rezultate, pogotovo kada se faktor regulacije utjecaja greške algoritma SVM poveća tisuću puta u odnosu na pretpostavljeni iznos. Evaluiranje modela s variranje ostalih parametara i pokrivanje veće količine slučajeva cilj su daljnjeg rada.

Ono što bi se u budućnosti također trebalo poboljšati je skup podataka koji se koriste. Naime, pri početku osmišljanja modela, ideja je bila raditi na tekstovima

studenata koji su te tekstove pisali na jeziku koji im nije materinji. U tim su tekstovima bile označene pogreške leksičkog transfera. Ipak, rano je ustanovljeno da je taj skup podataka premalen za ikakvu smislenu analizu. Uz to, često nije bio slučaj da su ti tekstovi ispravni s iznimkom leksičkog transfera nego su sadržavali i mnoge druge, veće greške. Te dvije činjenice razlog su zašto smo za skup podataka odabrali tekstove izvučene iz korpusa BNC. Naravno, kako su ti tekstovi većinom ispravni, primjere pogrešaka leksičkog transfera morali smo umjetno generirati na temelju višeznačnosti u jeziku L1. Puno bi bolje i smislenije bilo da je postojao veliki skup označenih primjera gdje se dogodio leksički transfer.

Druga važna komponenta koja bi se trebala poboljšati je rječnik. Naime, ono što je ključno za ovaj rad jest da rječnik ima mogućnost prevođenja jedne riječi u sve njene moguće prijevode i to u oba smjera. Kineski rječnik općenito nije lako pronaći, a i oni koji postoje obično nude samo mogućnost vraćanja jednog prijevoda riječi. BingTranslator, koji je u ovom radu korišten nije zadovoljavajući jer, iako nudi mogućnost vraćanja višestrukih prijevoda, često opet vraća samo jedan prijevod za riječi za koje sigurno ima više prijevoda. Uz to, taj rječnik nije pohranjen lokalno nego se za svaki novi prijevod treba se slati zahtjev serveru i čekati da ga on vrati što znatno usporava rad cijelog sustava. Uz kvalitetan lokalni rječnik cijeli bi sustav bolje funkcionirao.

Konačno, sustav opisan u ovom radu mogao bi se poboljšati uključivanjem koncepta sinonima. Naime, ovaj sustav sve riječi koje su u PS-okolini ispravne riječi smatra neispravnima. Ipak, u jeziku L2 mogu postojati sinonimi, te se tako može dogoditi da nekom mjestu u tekstu odgovara više različitih riječi, tj. da ima više jednako vrijednih rješenja. Kada bi postojao modul koji provjerava jesu li dvije riječi sinonimi, sustav bi se mogao poboljšati tako da ne tretira točno upotrijebljene sinonime kao neispravne.

Treba napomenuti da je popis parova engleskih riječi koje su vezane preko višeznačnosti u kineskom jeziku koji je ustupila prof. Marina Dodigović doprinio radu sustava. Treba se poraditi na tome da taj popis naraste i pokriva što više mogućih riječi.

Dobra je stvar kod ovog modela to što o jeziku L1 ovisi isključivo preko rječnika. U ovoj se implementaciji koristio kineski rječnik, ali da se koristio rječnik

bilo kojeg drugog jezika, koji ima spomenute karakteristike, sve ostalo u modelu moglo bi ostati nepromijenjeno. To čini ovaj model široko primjenjivim. Uz pretpostavku kvalitetnog rječnika, može se iskoristiti za svaki jezik bez ikakve izmjene.

Python se u procesu implementacije pokazao kao jezik koji korisniku olakšava da se usredotoči na znanstveno-istraživački aspekt rješenja. Zbog jednostavnosti jezika, razvoj je brz, a korisnik je u velikoj mjeri lišen briga o tome kako će neku komponentu modela implementirati.

Literatura

List of languages by total number of speakers. URL https://en.wikipedia.org/wiki/List_of_languages_by_total_number_of_speakers.

Performance measures for machine learning. URL http://www.cs.cornell.edu/courses/cs578/2003fa/performance_measures.pdf.

Mohammad Al-Khawalda i Ahmed Al-Oliemat. Linguistic transfer: Example from arabic users of english.

James Allen. *Natural Language Understanding (2nd ed.)*. Benjamin-Cummings Publishing Co., 1995.

Ma Chengchen. Lexical transfer from chinese to english in the written production of xjtlu students, 2015.

Gobinda G. Chowdhury. Natural language processing. *Annual Review of Information Science and Technology*, 37(1):51–89, 2003. doi: 10.1002/aris.1440370103.

Vivian Cook. *Effects of the Second Language on the First*. Multilingual Matters, 2003.

Vivian Cook i David Singleton. *Key Topics in Second Language Acquisition*. Multilingual Matters, 2014.

Michael A. Covington. A fundamental algorithm for dependency parsing. 2001. doi: 10.1.1.136.7335.

Marie-Catherine de Marneffe i Christopher D. Manning. Stanford typed dependencies manual. URL http://nlp.stanford.edu/software/dependencies_manual.pdf.

Steve R. Gunn. Support vector machines for classification and regression. URL http://ce.sharif.ir/courses/85-86/2/ce725/resources/root/LECTURES/SVM.pdf.

HJP. fraza. URL http://hjp.znanje.hr/index.php?show=search_by_id&id=fFlvWBU%3D.

J. Michael O'Malley i Anna Uhl Chamot. *Learning Strategies in Second Language Acquisition*. Cambridge University Press, 1990.

María Pilar i Agustín Llach. An overview of variables affecting lexical transfer in writing: A review study. *International Journal of Linguistics*, 2(1), 2010.

Alex J. Smola i Bernhard Scholkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004.

Seyed Hassan Talebi. Cross-linguistic transfer among iranian learners of english as a foreign language. *Issues in Educational Research*, 24(2):212–227, 2014.

Peter D. Turney i Patrick Pantel. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188, 2010.

Kristina Vučković. Model parsera za hrvatski jezik. 2009.

Max Welling. Support vector machines, a. URL http://www.ics.uci.edu/~welling/classnotes/papers_class/SVM.pdf.

Max Welling. Support vector regression, b. URL http://www.ics.uci.edu/~welling/classnotes/papers_class/SVregression.pdf.

Otkrivanje pogrešaka leksičkog transfera u tekstovima učenika stranog jezika

Sažetak

Automatizirano ispravljanje pogrešaka važan je zadatak obrade prirodnog jezika. Pogreške leksičkog transfera učestala su kod učenika stranih jezika. Najčešći je uzrok transfera višeznačnost riječi. Cilj ovog rada bio je osmisliti model koji otkriva i ispravlja takve pogreške za dva tipa jezičnih relacija: pridjevsko-imeničke i glagolsko-objektne. Implementirana su dva različita modela, od kojih drugi koristi metode nadziranog strojnog učenja, preciznije algoritam regresije pomoću stroja s potpornim vektorima. Jezik implementacije je Python. Ovaj projekt nastao je u suradnji s kineskim sveučilištem Xi'an Jiaotong-Liverpool University, te se oni ustupili skup podataka koji se jednim dijelom koristi u ovome radu.

Ključne riječi: obrada prirodnog jezika, automatizirano ispravljanje pogrešaka, leksički transfer, višeznačnost, relacije, stroj s potpornim vektorima, regresija

Detecting Lexical Transfer Errors of Second Language Learners

Abstract

Automated error correction is an important task of natural language processing. Lexical transfer errors are common with L2-learners. The biggest cause of transfer is word polysemy. The goal of this paper was to come up with a model that can detect and correct such errors for two language relation types: adjective-noun and verb-object. Two different models were implemented, the second of which uses supervised learning methods, more precisely the support vector regression algorithm. The model was implemented in Python. This project is carried out in cooperation with Xi'an Jiaotong-Liverpool University, China, who also make the dataset available, a part of which was used in this paper.

Keywords: natural language processing, automated error correction, lexical transfer, polysemy, relations, support vector machine, regression